Jianying Zhou
Moti Yung (Eds.)

LNCS 6123

Applied Cryptography and Network Security

8th International Conference, ACNS 2010
Beijing, China, June 2010
Proceedings

Springer

Jianying Zhou   Moti Yung (Eds.)

Applied Cryptography and Network Security

8th International Conference, ACNS 2010
Beijing, China, June 22-25, 2010
Proceedings

Springer

Volume Editors

Jianying Zhou
Institute for Infocomm Research
1 Fusionopolis Way, Singapore, 138632, Singapore
E-mail: jyzhou@i2r.a-star.edu.sg

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

Library of Congress Control Number: 2010928335

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-642-13707-5 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-13707-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

ACNS 2010, the 8th International Conference on Applied Cryptography and Network Security, was held in Beijing, China, during June 22-25, 2010. ACNS 2010 brought together individuals from academia and industry involved in multiple research disciplines of cryptography and security to foster the exchange of ideas.

ACNS was initiated in 2003, and there has been a steady improvement in the quality of its program over the past 8 years: ACNS 2003 (Kunming, China), ACNS 2004 (Yellow Mountain, China), ACNS 2005 (New York, USA), ACNS 2006 (Singapore), ACNS 2007 (Zhuhai, China), ACNS 2008 (New York, USA), ACNS 2009 (Paris, France). The average acceptance rate has been kept at around 17%, and the average number of participants has been kept at around 100.

The conference received a total of 178 submissions from all over the world. Each submission was assigned to at least three committee members. Submissions co-authored by members of the Program Committee were assigned to at least four committee members. Due to the large number of high-quality submissions, the review process was challenging and we are deeply grateful to the committee members and the external reviewers for their outstanding work. After extensive discussions, the Program Committee selected 32 submissions for presentation in the academic track, and these are the articles that are included in this volume (LNCS 6123). Additionally, a few other submissions were selected for presentation in the non-archival industrial track. The prize for the best student paper was awarded to Mehdi Tibouchi for his paper "On the Broadcast and Validity-Checking Security of PKCS#1 v1.5 Encryption", co-authored with Aurelie Bauer, Jean-Sebastien Coron, David Naccache, and Damien Vergnaud.

We would like to thank General Chair Yongfei Han and the local organizing team from Beijing University of Technology and ONETS for their efforts in putting this conference together. Our special thanks are due to Ying Qiu for managing the Easy Chair system for paper submission and review. We would also like to thank all the authors who submitted papers and the participants from all over the world who chose to honor us with their attendance.

April 2010

Jianying Zhou
Moti Yung

ACNS 2010

8th International Conference on
Applied Cryptography and Network Security

Beijing, China
June 22–25, 2010

Organized and Sponsored by

Beijing University of Technology & ONETS, China

General Chair

Yongfei Han	BJUT & ONETS, China

Program Chairs

Jianying Zhou	Institute for Infocomm Research, Singapore
Moti Yung	Columbia University & Google, USA

Program Committee

Michel Abdalla	ENS, France
Ben Adida	Harvard University, USA
N. Asokan	Nokia, Finland
Joonsang Baek	I2R, Singapore
Lucas Ballard	Google, USA
Feng Bao	I2R, Singapore
Lujo Bauer	Carnegie Mellon University, USA
Alex Biryukov	Uni. of Luxembourg, Luxembourg
Alexandra Boldyreva	Georgia Tech, USA
Colin Boyd	QUT, Australia
Levente Buttyan	BME, Hungary
Liqun Chen	HP Laboratories, UK
Songqing Chen	George Mason University, USA
Debra Cook	Telcordia, USA
Cas Cremers	ETH Zurich, Switzerland
Sabrina De Capitani di Vimercati	UNIMI, Italy
Robert Deng	SMU, Singapore

Orr Dunkelman	Weizmann Institute, Israel
Dieter Gollmann	TU Hamburg-Harburg, Germany
Stefanos Gritzalis	University of the Aegean, Greece
Marc Joye	Technicolor, France
Charanjit Jutla	IBM, USA
Angelos Keromytis	Columbia University, USA
Xuejia Lai	Shanghai Jiao Tong University, China
Dong Hoon Lee	Korea University, Korea
Ninghui Li	Purdue University, USA
Yingjiu Li	SMU, Singapore
Benoit Libert	UCL, Belgium
Dongdai Lin	Institute of Software, China
Peng Liu	Pennsylvania State University, USA
Javier Lopez	University of Malaga, Spain
Mark Manulis	TU Darmstadt, Germany
Fabio Martinelli	CNR, Italy
Atefeh Mashatan	EPFL, Switzerland
Paolo Milani	Technical University of Vienna, Austria
Chris Mitchell	RHUL, UK
Atsuko Miyaji	JAIST, Japan
Tatsuaki Okamoto	NTT, Japan
Alina Oprea	RSA Laboratories, USA
Elisabeth Oswald	University of Bristol, UK
Benny Pinkas	University of Haifa, Israel
Pandu Rangan	Indian Institute of Technology, India
Vincent Rijmen	TU Graz, Austria
Mark Ryan	University of Birmingham, UK
Ahmad-Reza Sadeghi	Ruhr-Uni. Bochum, Germany
Reihaneh Safavi-Naini	University of Calgary, Canada
Palash Sarkar	Indian Statistical Institute, India
Nitesh Saxena	Poly Institute of New York Uni., USA
Radu Sion	Stony Brook University, USA
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	FUN, Japan
Duncan Wong	City University of Hong Kong, China

Organizing Chairs

Jian Li	Beijing University of Technology, China
Yu Wang	ONETS, China

Publicity Chairs

Javier Lopez	University of Malaga, Spain
Tsuyoshi Takagi	FUN, Japan
Sijin Li	ONETS, China

Steering Committee

Yongfei Han	BJUT & ONETS, China
Moti Yung	Columbia University & Google, USA
Jianying Zhou	Institute for Infocomm Research, Singapore

External Reviewers

Gergely Acs
Isaac Agudo
Efthimia Aivaloglou
Mansoor Alicherry
Elli Androulaki
Myrto Arapinis
Frederik Armknecht
Tomoyuki Asano
Elias Athanasopoulos
Man Ho Au
Jean-Philippe Aumasson
Sumeet Bajaj
Collard Baudoin
Bruno Blanchet
Marina Blanton
Shaoying Cai
Tianjie Cao
Bogdan Carbunar
Julien Cathalo
Sambuddho Chakravarty
Shiping Chen
Wei Cheng
Kyu Young Choi
Tom Chothia
Cheng-Kang Chu
Ji Young Chun
Gabriele Costa
Gabriela Cretu
Ning Ding
Alexandra Dmitrienko
Ming Duan
Mark Felegyhazi
Carmen Fernandez-Gago
Dario Fiore
Martin Gagne
Zheng Gong
Juan Gonzalez
Choudary Gorantla
Tzipora Halevi
Christoph Herbst
Shlomo Hershkop
Tamas Holczer
Qiong Huang
Toshiyuki Isshiki
Stas Jarecki
Ayman Jarrous
Seny Kamara
Giorgos Karopoulos
Vasileios P. Kemerlis
Bum Han Kim
Kitak Kim
Ilya Kizhvatov
Miroslav Knezevic
Clemens Kolbitsch
Deguang Kong
Elisavet Konstantinou
Woo Kwon Koo
Leanid Krautsevich
Swarun Kumar
Virendra Kumar
Francesco la Torre
Fabien Laguillaumie
Aliaksandr Lazouski
Hyun Sook Lee
Kwangsu Lee
Fagen Li
Tieyan Li
Wei Li
Yan Li
Jingqiang Lin
Hanwu Liu
Joseph. K. Liu
Mei Cheng Liu
Xianhui Lu
Christoph Ludwig
Yiyuan Luo
Hans Lohr
Ilaria Matteucci
Marcel Medwed
Daisuke Moriyama
Andreas Moser
Francisco Moyano
Pablo Najera
Toru Nakanishi
Kris Narayan
Kris Narayan
Matthias Neugschwandtner
Ching Yu Ng
Ivica Nikolic
Geon Tae Noh
Adam O'Neill
Wakaha Ogata
Katsuyuki Okeya
Kazumasa Omote
Khaled Ouafi
Carles Padro
Jung Ha Paik
Vasilis Pappas
Sai Tej Peddinti
Chris Peikert
Kun Peng
Christophe Petit
Thomas Plantard
Bertram Poettering
Mariana Raykova
Tzachy Reinman
Evangelos Reklitis
Ruben Rios
Panagiotis Rizomiliotis
Rodrigo Roman

Bagus Santoso
Werner Schindler
Michael Schneider
Thomas Schneider
Dominique Schroder
Sharmila Deva selvi
Nicolas Sendrier
Daniele Sgandurra
Siamak Shahandashti
Jun Shao
Takeshi Shimoyama
Francesco Sica
Stelios Sidiroglou
Matt Smart
Nigel Smart
Ben Smyth
Miroslava Sotakova
Douglas Stebila
Thorsten Strufe
Chunhua Su
Xiaorui Sun
Martin Szydlowski
Kouya Tochikubo
Ashraful Tuhin
Michael Tunstall
Berkant Ustaoglu
Istvan Vajda
Serge Vaudenay
Jose Luis Vivas
Sree Vivek
Jonathan Voris
Camille Vuillaume
Christian Wachsmann
Zhongmei Wan
Xinyuan Wang
Ralf-Philipp Weinmann
Zhongming Wu
Fubiao Xia
Jing Xu
Guanhua Yan
Qiang Yan
Guomin Yang
Artsiom Yautsiukhin
Kuo-Hui Yeh
Kazuki Yoneyama
Junfeng Yu
Tsz Hon Yuen
Angelika Zavou
Bin Zhang
Min Zhang
Mingwu Zhang
Shengzhi Zhang
Xinwen Zhang
Yunlei Zhao
JinMin Zhong
Chunfang Zhou
Hong-Sheng Zhou
Bo Zhu

Table of Contents

On the Broadcast and Validity-Checking Security of PKCS#1 v1.5 Encryption

Aurélie Bauer[1], Jean-Sébastien Coron[2], David Naccache[1], Mehdi Tibouchi[1,2,⋆], and Damien Vergnaud[1]

[1] École normale supérieure – C.N.R.S. – I.N.R.I.A.
Département d'informatique, Groupe de cryptographie
45, rue d'Ulm, F-75230 Paris CEDEX 05, France
{aurelie.bauer,david.naccache,mehdi.tibouchi,damien.vergnaud}@ens.fr

[2] Université du Luxembourg
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg, Luxembourg
jean-sebastien.coron@uni.lu

Abstract. This paper describes new attacks on PKCS#1 v1.5, a deprecated but still widely used RSA encryption standard.

The first cryptanalysis is a broadcast attack, allowing the opponent to reveal an identical plaintext sent to different recipients. This is nontrivial because different randomizers are used for different encryptions (in other words, plaintexts coincide only partially).

The second attack predicts, using a *single* query to a validity checking oracle, which of two chosen plaintexts corresponds to a challenge ciphertext. The attack's success odds are very high.

The two new attacks rely on different mathematical tools and underline the need to accelerate the phase out of PKCS#1 v1.5.

Keywords: PKCS#1 v1.5, Encryption, Broadcast Encryption, Cryptanalysis.

1 Introduction

PKCS stands for *Public-Key Cryptography Standards* [15]. PKCS is a large *corpus* of specifications covering RSA encryption, Diffie-Hellman key agreement, password-based encryption, syntax (extended-certificates, cryptographic messages, private-key information and certification requests) and selected attributes. PKCS was initially developed by RSA Laboratories, Apple, Digital, Lotus, Microsoft, MIT, Northern Telecom, Novell and Sun and regularly updated since. Today, PKCS has become part of several standards and of a wide range of security products including Internet Privacy-Enhanced Mail.

Amongst the PKCS collection, PKCS#1 v1.5 describes a particular encoding method for RSA encryption called `rsaEncryption`. In essence, the protected

⋆ This research was completed while the fourth author was visiting the Okamoto Research Laboratory at the NTT Information Sharing Platform (Tokyo, Japan).

data is first encrypted under a randomly chosen key κ using a symmetric block-cipher (*e.g.* a triple DES in CBC mode) then κ is RSA-encrypted (wrapped) with the recipient's public key.

In 1998, Bleichenbacher [5] published an adaptive chosen-ciphertext attack against PKCS#1 v1.5 capable of recovering arbitrary plaintexts from about half a million ciphertexts. Although active adversary models are generally regarded as theoretical concerns, Bleichenbacher's attack makes use of an oracle that only detects conformance with respect to the padding format, a real-life assumption that resulted in a practical threat. PKCS#1 v1.5 was subsequently updated (release 2.0 [16]) and patches were issued to users wishing to continue using the old version of the standard. As we write these lines[1] and despite its vulnerabilities, PKCS#1 v1.5 *is still widely used*. Millions of (patched) PKCS#1 v1.5 programs remain deployed. Provably secure algorithms such as RSA-OAEP [11] and RSA-KEM are recommended replacements, but not widespread yet [17].

Independently, there exist several well-known chosen-plaintext attacks on RSA-based encryption schemes [8,6]. These typically enable an attacker to decrypt ciphertexts at moderate cost without factoring the public modulus. The most powerful cryptanalytic tool applicable to low exponent RSA is certainly an attack due to Coppersmith [7]. As a matter of fact, one major reason for adding randomness to encrypted messages[2] is to thwart such attacks.

The last publication concerning PKCS#1 v1.5's security [9] presented a somewhat atypical attack allowing the opponent to retrieve plaintexts ending by enough zero bits.

This paper describes two new weaknesses in PKCS#1 v1.5:

– The possibility to predict, *using a single decryption query*, which of two chosen plaintexts corresponds to a challenge ciphertext. The attack's success odds are very high.
– A broadcast attack allowing to decrypt an identical message sent to several recipients.

From a mathematical perspective, the two techniques are totally different. The authors regard these as a wake-up call to accelerate the phase out of PKCS#1 v1.5.

2 PKCS#1 v1.5 Encryption

We assume that the reader is familiar with the traditional public-key encryption definitions and security model preliminaries. For the sake of completeness we refer the reader to Appendix A.

2.1 The PKCS#1 v1.5 Encoding Function

PKCS#1 v1.5 describes a particular encoding method (`rsaEncryption`) for RSA encryption [20]. Consider an RSA modulus N, and let k denote its byte-length

[1] November 2009.

[2] Besides attempting to achieve indistinguishability.

(*i.e.* $2^{8(k-1)} < N < 2^{8k}$). Let m be an $|m|$-byte message with $|m| < k - 11$. The PKCS#1 v1.5 padding $\mu(m)$ of m is defined as follows:

1. A randomizer r consisting in $k - 3 - |m| \geq 8$ nonzero bytes is generated uniformly at random;
2. $\mu(m) = \mu(m, r)$ is the integer converted from the octet-string:

$$\mu(m, r) = \mathsf{0002}_{16}||r||\mathsf{00}_{16}||m \tag{1}$$

(the leading 00 octet guarantees that the encryption block is an integer smaller than N).

The encryption of a message m of $|m| < k - 11$ bytes is defined as

$$c = \mu(m, r)^e \bmod N$$

for some randomizer r of $k - 3 - |m|$ nonzero bytes.

To decrypt $c \in \mathbb{Z}_N^*$, compute $c^d \bmod N$, convert the result to a k-byte octet-string and parse it according to equation (1). If the string cannot be parsed unambiguously or if r is shorter than eight octets, the decryption algorithm $\mathcal{D}$ outputs $\perp$; otherwise, $\mathcal{D}$ outputs the plaintext m.

2.2 Previous Attacks on PKCS#1 v1.5

In 1998, Bleichenbacher [5] published an attack on PKCS#1 v1.5 capable of recovering arbitrary plaintexts from a large number of ciphertexts validation queries. This attack established that PKCS#1 v1.5 is not ℓ-GOAL-ATTACK for a large[3] ℓ, GOAL $\in$ {OW, IND, NM} and ATTACK $\in$ {VCA, CCA} (see Appendix A.2 for definitions of these security notions).

In 2000, Coron, Naccache, Joye and Paillier [9] introduced two new CPAs on PKCS#1 v1.5. The first attack can be considered as a IND-CPA when e is small (for plaintext ending by sufficiently many zeroes). The second attack applies to arbitrary e, provided that $|m|$ is large and most message bits are zeroes. Thus PKCS#1 v1.5 is not GOAL-CPA for a small e or a large $|m|$, for GOAL $\in$ {IND, NM}.

The previous crytanalytic results are summarized in Figure 1. UBK-CPA is equivalent to Factoring but establishing the UBK-VCA and UBK-CCA security is equivalent to proving (or refuting) the equivalence of the factoring and the RSA Problem (which is a long-standing open question in cryptography). In the rest of the paper, we will study the remaining security notions and prove that PKCS#1 v1.5 is:

– OW-CPA assuming the intractability of the RSA problem (§ 3);
– not OW-CCA for $\ell = 2$ (§ 3);
– not NM-CPA (§ 4.1);
– not IND-VCA for $\ell = 1$ (§ 4.2);
– not OW-CPA in a multi-user setting (§ 5).

[3] $3 \cdot 10^5 < \ell < 2 \cdot 10^6$ for $512 < \log_2 N < 1024$.

$$\begin{tikzcd}
\text{UBK-CCA} \arrow[d, Rightarrow] & \text{UBK-VCA} \arrow[l, Rightarrow] \arrow[d, Rightarrow] & \text{UBK-CPA} \arrow[l, Rightarrow] \arrow[d, Rightarrow, "= \text{Factoring}"] \\
\text{OW-CCA} \arrow[d, Rightarrow] & \cancel{\text{OW-VCA}} \arrow[l, Rightarrow] \arrow[d, Rightarrow, "\text{large } \ell \text{ ([5])}"] & \text{OW-CPA} \arrow[l, Rightarrow] \arrow[d, Rightarrow] \\
\text{IND-CCA} \arrow[d, Rightarrow] & \cancel{\text{IND-VCA}} \arrow[l, Rightarrow] \arrow[d, Rightarrow, "\text{large } \ell \text{ ([5])}"] & \cancel{\text{IND-CPA}} \arrow[l, Rightarrow] \arrow[d, Rightarrow, "\text{small } e \text{ or large } |m| \text{ ([9])}"] \\
\text{NM-CCA} & \text{NM-VCA} \arrow[l, Rightarrow] & \text{NM-CPA} \arrow[l, Rightarrow]
\end{tikzcd}$$

Fig. 1. PKCS#1 v1.5 Security

3 On PKCS#1 v1.5's OW-CPA-Security

In this paragraph, we prove the following result:

Proposition 1. *The OW-CPA security of* PKCS#1 v1.5 *is equivalent to the* RSA *Problem.*

In [10, Lemma 2], Coron, Joye, Naccache and Paillier proved that for suitable parameters, the existence of an algorithm that on input $y \in \mathbb{Z}_N^*$ outputs the k_1 least significant bits of $y^d \bmod N$ is equivalent to the existence of an RSA inverter. Following their approach, we prove the following lemma:

Lemma 1. *Let $\mathcal{A}$ be a OW-CPA-adversary against* PKCS#1 v1.5 *with success probability ε within time τ, with uniformly distributed messages of (maximum) length $k - 11$. There exists an algorithm $\mathcal{B}$ that solves the* RSA *Problem with success probability ε' within time τ', where:*

$$\begin{cases} \varepsilon' \geq \eta^2 \cdot \varepsilon^2 - 2^{-k}\varepsilon \\ \tau' \leq 2 \cdot \tau + \mathtt{poly}(k) \end{cases}$$

where η is a constant independent of k and $\eta \geq 5 \cdot 10^{-8}$.

Proof. Let $\mathcal{A}$ be a OW-CPA-adversary against PKCS#1 v1.5 with success probability ε within time τ. We construct an algorithm $\mathcal{B}$ that on input $y \in \mathbb{Z}_N^*$ outputs y^d with success probability ε' within time τ':

1. $\mathcal{B}$ picks $\alpha \in \mathbb{Z}_N^*$ uniformly at random;
2. $\mathcal{B}$ sets $y_0 = y$ and $y_1 = y \cdot \alpha^e$
3. Let us denote, for $i \in \{0, 1\}$:

$$y_i^d = \omega_i \cdot 2^\beta + m_i$$

with $\beta = 8(k - 11)$. y_i is a valid PKCS#1 v1.5 ciphertext if $\omega_i = \mathtt{0002}||r_i||00$ where r_i is a 8-(nonzero)-octet string. This happens with probability $\eta \geq (255/256)^8 \cdot 2^{-24}$. If this happens, then with probability ε, $\mathcal{A}$ will return m_i on input y_i (for $i \in \{0, 1\}$).

4. Therefore with probability at least $(\eta\varepsilon)^2$, we obtain:

$$\alpha(\omega_0 \cdot 2^{\beta} + m_0) = \omega_1 2^{\beta} + m_1 \bmod N.$$

Letting $c_1 = 2^{-\beta}(\alpha m_0 - m_1) \mod N$, we get the equation in (ω_0, ω_1):

$$\omega_1 - \alpha \cdot \omega_0 = c_1 \bmod N \tag{2}$$

From [10, Lemma 3], there exists an algorithm that given this system will output a solution (ω_0, ω_1) (should such a solution exist) with probability at least $1-2^{-k}$ on the choice of α. □

Using the same technique, this can be extended to messages of different length, with a possibly higher constant loss in the reduction.

As a byproduct of the previous proof one immediately gets that:

Proposition 2. PKCS#1 v1.5 *is not* 2-*OW-CCA*.

Proof. Thanks to RSA's homomorphic properties, an adversary can mask the challenge ciphertext c as $c' = cr^e$ for some random r and with two decryption oracle queries, compute the e-th root x' of c' as in the previous proof. Then $x = x'/r$ is the e-th root of c from which, the adversary retrieves readily the plaintext. □

4 PKCS#1 v1.5 Malleability and Indistinguishability

We now show that μ is neither NM-CPA-secure nor IND-VCA-secure. The general idea is the following. Let m be some message to be encrypted, and $\mu(m)$ a corresponding PKCS#1 v1.5 padded encryption block. If m has $Z > 2$ trailing zero bits, then $\mu(m)$ is divisible by 2^Z, and $\mu(m) - \mu(m)/2^Z \bmod N$ has a good probability of still being a valid encryption block. This is not usually the case when the Z LSBs of m are not all zero.

Let $c = \mu(m)^e \bmod N$ be a ciphertext of some message m. If m has $Z > 2$ trailing zeroes, $c' = c \cdot (1-2^{-Z})^e \bmod N$ will often be a valid ciphertext of some other message m' which can be related to m: this contradicts non-malleability under chosen plaintext attack. Moreover, if one is granted *one* query to a validity oracle, it is possible to distinguish ciphertexts of plaintexts with trailing zeroes and ciphertexts from plaintexts whose LSBs are not all zero: this contradicts indistinguishability under validity checking attack. Note that if i queries are allowed, the distinguishing success odds can quickly approach one by iterating the test with $c'_i = c \cdot (i - 2^{-Z})^e \bmod N$ for $i = 1, 2, \ldots$

We will develop this idea in further detail in the coming sections.

4.1 On PKCS#1 v1.5's NM-CPA Security

Let k be the byte-size of N, $Z = 4k$, and M a positive integer such that $M+Z+1$ is a multiple of 8 and $(M+Z+1)/8 < k - 11$. We consider messages of the following form:

$$m = \underbrace{\bar{m}}_{M \text{ bits}} \| 1_2 \| \underbrace{0 \cdots 0_2}_{Z \text{ zero bits}}$$

Let $\mathcal{M}$ denote the uniform distribution over messages of this form. Furthermore, we define a relation $\mathcal{R}$ over messages of length $l = M + Z + 1$ as follows: for any l-bit two messages m_1, m_2 (not necessarily of the previous form), $\mathcal{R}(m_1, m_2)$ holds if and only if the M MSBs of m_1 and m_2 coincide. In particular, for any given message m_2, there is exactly one $m_1 \in \mathcal{M}$ such that $\mathcal{R}(m_1, m_2)$.

Now, consider $m \leftarrow \mathcal{M}$. We can write $\mu(m) \cdot (1 - 2^{-Z})$ as:

$$\begin{array}{rl} & \mathsf{0002}_{16} \| r \| \mathsf{00}_{16} \| \bar{m} \| \mathsf{1}_2 \| \ \mathsf{0} \cdots \mathsf{0}_2 \\ - & \qquad\qquad\qquad\qquad \mathsf{0002}_{16} \| r \| \mathsf{00}_{16} \| \bar{m} \| \mathsf{1}_2 \\ \hline = & \mathsf{0002}_{16} \| r \| \mathsf{00}_{16} \| \bar{m} \| \mathsf{0}_2 \| \text{ some digits} \cdots \text{ some digits} \end{array}$$

Hence, $\mu(m) \cdot (1 - 2^{-Z}) = \mu(m')$ for some message $m' \neq m$ such that $\mathcal{R}(m, m')$.

Consider, the NM-CPA adversary $\mathcal{A}$ which outputs sampling algorithm $\mathcal{M}$ in the setup stage, and transforms a challenge ciphertext c into $c' = c \cdot (1 - 2^{-Z})^e \bmod N$. $\mathcal{A}$'s advantage is:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{NM\text{-}CPA}} = \Pr[\mathcal{R}(m, m')] - \Pr[m_0 \xleftarrow{\$} \mathcal{M}; \mathcal{R}(m_0, m')] = 1 - 2^{-M} \geq 1/2$$

which is non-negligible. Therefore, PKCS#1 v1.5 encryption is not NM-CPA-secure.

Noted that $\mathcal{A}$'s advantage is, in fact ~ 1. For a 1024-bit modulus and a 128 bit randomizer, we have $M = 383$, making it exceedingly unlikely that $\mathcal{A}$ will ever fail.

4.2 On PKCS#1 v1.5's IND-VCA Security

We now show how to contradict ciphertext indistinguishability under validity checking attack using a single oracle query. There are two natural types of validity oracles: one which determines whether a given query is a valid ciphertext associated to a plaintext of any length, and the other which also checks message length. We can always contradict IND-VCA-security in the non-length-checking case (which is the one considered in Bleichenbacher's attack [5]) with a single oracle query. Furthermore, if the byte-length of the randomizer is *constant*, as permitted by the PKCS#1 v1.5 standard, it is also possible to break IND-VCA-security with a single query to a length-checking oracle. Both attacks stem from the following result.

Proposition 3. *Let $c = \mu(m)^e \bmod N$ be the ciphertext associated to some byte-string message m, ω the byte-length of the randomizer and $c' = c \cdot (1 - 2^{-4})^e \bmod N$.*

1. *If the least significant nibble of m is not $\mathsf{0}_{16}$, then c' is never a valid ciphertext.*
2. *If m is a message consisting of a string of $\mathsf{00}_{16}$ bytes, then c' is a valid ciphertext with probability at least 0.47. c' is a valid ciphertext corresponding to a message of the same length as m with probability at least:*

$$\frac{64}{1445} \left(\frac{239}{255} \right)^{\omega - 1}$$

Proof. Starting with the first assertion, consider a message m such that c' is a valid ciphertext. This implies that $\mu(m) - \mu(m) \cdot 2^{-4} \bmod N$ is a valid encoding string that, in particular, begins with the same $\mathsf{0002}_{16}$ pattern as $\mu(m)$.

If, for an integer x, we denote by $\bar{x}$ the only integer in $(-N/2, N/2)$ such that $x \equiv \bar{x} \mod N$, it follows that:

$$|\overline{\mu(m) \cdot 2^{-4} \bmod N}| < 2^{8k-16}$$

Consider the set S of residue classes $x \bmod N$ such that $|\bar{x}| < 2^{8k-16}$. Clearly, $|S| = 2^{8k-15} - 1$. On the other hand, let T^+ be the set consisting of k-byte strings of the form $\mathsf{000}_{16}\|u\|\mathsf{0}_{16}$, and T be the union of T^+ and $-T^+$ (where opposites are taken $\bmod N$). We also have $|T| = 2^{8k-15} - 1$ and T maps into S under multiplication by $2^{-4} \bmod N$. Since multiplication by 2^{-4} is a permutation of $\mathbb{Z}_N$, we infer that $(y \cdot 2^{-4} \bmod N) \in S$ if and only if $y \in T$.

In particular, if c' is a valid ciphertext, we get $\mu(m) \in T$. By inspection of its top bits, we see that $\mu(m)$ cannot be in $-T^+$, so it has to be in T^+. Its least-significant nibble must thus be $\mathsf{0}_{16}$ as required.

Turning now to the second assertion, let m be the zero-message of some fixed byte-length. Write the encryption block $\mu(m)$ as follows:

$$\mu(m) = \mathsf{0002}_{16}\|r_{2\omega-1}\|r_{2\omega-2}\| \cdots \|r_1\|r_0\|\mathsf{00}_{16}\|\mathsf{00}\cdots\mathsf{00}_{16}$$

where $r_0, \ldots, r_{2\omega-1}$ are randomizer's nibbles. Recall that the randomizer bytes $r_{2j}\|r_{2j+1}$ are chosen uniformly and independently at random in the range $\mathsf{01}_{16}$, $\ldots$, FF_{16}.

Assuming that $r_{2\omega-1}$ is at least 4 (which happens with probability $(256 - 4 \times 16)/255 = 64/85$, and which we will henceforth assume), we can write $(1 - 2^{-4})\mu(m)$ as:

$$\begin{array}{rcccccccccc}
\mathsf{0002}_{16} & \| \; r_{2\omega-1} & \| \; r_{2\omega-2} & \| \; \cdots & \| \; r_1 & \| \; r_0 & \| & \mathsf{00}_{16} & \| \; \mathsf{00}\cdots\mathsf{00}_{16} \\
-\ \mathsf{0000}_{16} & \| \quad \mathsf{2}_{16} & \| \; r_{2\omega-1} & \| \; \cdots & \| \; r_2 & \| \; r_1 & \| & r_0\|\mathsf{0}_{16} & \| \; \mathsf{00}\cdots\mathsf{00}_{16} \\
\hline
=\ \mathsf{0002}_{16} & \| \; r'_{2\omega-1} & \| \; r'_{2\omega-2} & \| \; \cdots & \| \; r'_1 & \| \; r'_0 & \| & s & \| \; \mathsf{00}\cdots\mathsf{00}_{16}
\end{array}$$

where $r'_j \equiv r_j - r_{j+1} - \kappa_j \mod 16$ for some carry bit κ_j.

Then, $\mu' = (1 - 2^{-4})\mu(m)$ is a valid encoding block if and only if the first 8 randomizer bytes, namely $r'_{2\omega+1-2j}\|r'_{2\omega-2j}$, $j = 1, \ldots, 8$, are all nonzero. μ' is a valid encoding block for a message of the same length as m (or for short, "*strongly valid*") if and only if $s = 0$ and all the padding bytes $r'_{2j+1}\|r'_{2j}$, $j = 0, \ldots, \omega - 1$, are nonzero. We will find an explicit lower bound for the probability of these events.

Note first that a sufficient condition for $r'_{2j+1}\|r'_{2j}$ to be nonzero is that $r'_{2j+1} \neq 0$. This nibble is zero if and only if $r_{2j+2} \equiv r_{2j+1} - \kappa_{2j+1}$. Now r_{2j+2} is picked independently of r_{2j+1}, since they belong to different bytes; it is also independent of κ_{2j+1}, which only depends on lower order bytes. Consequently:

$$\begin{aligned}\Pr[r'_{2j+1} = 0] &= \sum_{\rho=0}^{15} \Pr[r_{2j+2} = \rho] \cdot \Pr[r_{2j+1} - \kappa_{2j+1} \equiv \rho \mod 16] \\ &\leq \max_{\rho} \Pr[r_{2j+2} = \rho] = \frac{16}{255}\end{aligned}$$

and this bound still holds conditionally to any assignment of the lower order nibbles r'_{2i+1}, $i < j$. Therefore:

$$\begin{aligned}\Pr[\mu' \text{ is valid}] &\geq \Pr[r_{2\omega-1} \geq 4 \wedge r'_{2\omega-3} \neq 0 \wedge r'_{2\omega-5} \neq 0 \wedge \cdots \wedge r'_{2\omega-15} \neq 0] \\ &\geq \frac{64}{85} \cdot \left(1 - \frac{16}{255}\right)^7 \geq 0.47\end{aligned}$$

Furthermore:

$$\begin{aligned}\Pr[\mu' \text{ is strongly valid}] &\geq \Pr[r_{2\omega-1} \geq 4 \wedge r'_{2\omega-3} \neq 0 \wedge \cdots \wedge r'_1 \neq 0 \wedge r_0 = 0] \\ &\geq \frac{64}{85} \cdot \left(1 - \frac{16}{255}\right)^{\omega-1} \cdot \frac{15}{255} = \frac{64}{1445} \cdot \left(\frac{239}{255}\right)^{\omega-1}\end{aligned}$$

The corresponding validity assertions for c' follow immediately. □

Consider the IND-VCA adversary $\mathcal{A}$ defined as follows. In the setup stage, $\mathcal{A}$ outputs two equal-length messages m_0, m_1 with m_1 not zero-terminated (e.g. $\mathsf{00} \cdots \mathsf{0001}_{16}$) and m_0 consisting of $\mathsf{00}_{16}$ bytes only. Then, upon receiving a challenge ciphertext $c = \mu(m_b)^e \bmod N$, $\mathcal{A}$ makes a single oracle query and outputs $b' = 0$ or 1 according to whether $c' = c \cdot (1 - 2^{-4}) \bmod N$ is a valid ciphertext or not. Its advantage is then:

$$\begin{aligned}\mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}VCA}} &= \Pr[b' = 0|b = 0] - \Pr[b' = 1|b = 0] = \Pr[b' = 0|b = 0] - 0 \\ &\geq \begin{cases} 0.47 & \text{if the oracle doesn't check message length} \\ \frac{64}{1445} \cdot \left(\frac{239}{255}\right)^{\omega-1} & \text{otherwise} \end{cases}\end{aligned}$$

which is non-negligible. In the length-checking case, it is over 2.8% (resp. 1.6%) for 64-bit (resp. 128-bit) randomizers.

In the non-length-checking case, we can obtain an even better advantage using $c' = c \cdot (1 - 2^{-8}) \bmod N$ (shifting by 8 bits instead of 4). The proof works similarly provided that N satisfies $N > 2^{8k-7}$, which is not required by the PKCS#1 v1.5 standard but is usually verified in practice. This yields an advantage of at least:

$$\frac{252}{255} \cdot \left(\frac{254}{255}\right)^7 > 0.96$$

5 Broadcast Attacks on PKCS#1 v1.5

We now examine the security of PKCS#1 v1.5 in a multiple users context.

In such a scenario, *i.e.* when broadcast encryption is performed, the sender wishes to transmit the same message m to ℓ parties $P_1, \ldots, P_\ell$. As each party has its own key $\mathsf{pk}_i = (e, N_i)$ (with a common public exponent e), the sender encrypts m using all the pk_i's and sends the resulting ciphertexts $c_1, \ldots, c_\ell$ to the corresponding recipients. It has long been known that textbook RSA encryption should not be used in such a context, since an attacker can easily recover the plaintext using the Chinese Remainder Theorem as long as $\ell \geq e$. Therefore, m has to be padded before applying the RSA function, and the padding has to be different for each recipient.

In 1988, Håstad [12] showed that using different linear paddings $\mu_i(m)$ for all parties is not enough to guarantee security. Indeed, when e is small, e ciphertexts are again sufficient to efficiently recover m provided that the encoding functions μ_i are known to the attacker. To achieve this result, Håstad expressed the attack in terms of finding small roots of a univariate modular polynomial, which he accomplishes using Coppersmith's techniques [7].

Håstad's attack does not apply to PKCS#1 v1.5, since the padding used for a given recipient is random, and is thus unknown to an attacker. The following sections will overcome this difficulty. Our main result is as follows:

Proposition 4. *Let $c_1, \ldots, c_\ell$ be ℓ PKCS#1 v1.5 ciphertexts of the same message m, of byte length $|m|$. Each c_i is encrypted for a receiver having $\mathsf{pk}_i = (e, N_i)$. All N_i are k-byte long. Then there exists a heuristic algorithm which, given $c_1, \ldots, c_\ell$, outputs m, if:*

$$\ell > \frac{e|m|}{k - e(k - |m| - 3)} > 0$$

Its complexity is polynomial in $e, |m|$ and k but exponential in the number of ciphertexts ℓ.

We describe this algorithm in the coming sections. The core idea is to reduce the problem to finding small modular roots of a multivariate polynomial equation, which can be achieved using a standard generalization of Coppersmith's techniques. As usual, this generalization relies on an assumption concerning independence between polynomials, which makes the algorithm heuristic.

5.1 The Multivariate Polynomial of Broadcast PKCS#1 v1.5

Recall (section 2.1) that to encrypt message m for recipient P_i, a PKCS#1 v1.5 sender first generates an $|r_i|$-byte randomizer r_i, and then computes the encoding function:

$$\mu(m, r_i) = \mathsf{0002}_{\mathsf{16}} \| r_i \| \mathsf{00}_{\mathsf{16}} \| m$$

Numerically, this gives:

$$\mu(m, r_i) = m + 2^{8|m|+8} r_i + 2^{8|m|+8|r_i|+9}$$

The ciphertext c_i is then computed as $c_i = \mu(m, r_i)^e \bmod N_i$.

Consider then an adversary $\mathcal{A}$ who obtains $c_1, \dots, c_\ell$. Since the N_i are of the same size, the randomizers r_i have a common byte length $|r|$. Therefore, the ciphertexts collected by $\mathcal{A}$ can be written as:

$$c_1 = (m + Ar_1 + B)^e \mod N_1, \quad \dots, \quad c_\ell = (m + Ar_\ell + B)^e \mod N_\ell$$

where $A = 2^{8|m|+8}$ and $B = 2^{8|m|+8|r|+9}$. Obviously, the N_i are pairwise coprime (otherwise, the factorization of some of the N_i could easily be recovered). Thus, the Chinese Remainder Theorem ensures that the previous equations can be rewritten as a single congruence mod $N = N_1 \cdots N_\ell$:

$$u_1 c_1 + \cdots + u_\ell c_\ell = u_1(m + Ar_1 + B)^e + \cdots + u_\ell(m + Ar_\ell + B)^e \mod N$$

where the constants $u_1, \dots, u_\ell$ are given by the extended Euclidean algorithm. It follows that the tuple $(m, r_1, \dots, r_\ell)$ is a root of the multivariate modular polynomial:

$$f(x, y_1, \dots, y_\ell) = u_1(x + Ay_1 + B)^e + \cdots + u_\ell(x + Ay_\ell + B)^e - C \mod N \quad (3)$$

where $C = u_1 c_1 + \cdots + u_\ell c_\ell$. This root is *small* in the sense that all of its components are bounded by quantities that are small compared to N: m is smaller than $2^{8|m|}$ and each r_i is bounded by $2^{8|r|}$. Under suitable conditions on $|m|$ and $|r|$ which will be detailed below, it will thus become feasible to recover this root, and hence m, in polynomial time using Coppersmith's techniques, as recalled in Appendix B.

In particular, we show in section B.3 that the lattice construction of Jochemsz and May [14] yields a heuristic polynomial time algorithm for recovering the small root of f under the following condition:

$$\ell|r| + |m| < \frac{t(\nu)}{s(\nu)} \cdot \ell k$$

where ν is a parameter which determines the attack's complexity, and $s(\nu)$, $t(\nu)$ are defined as:

$$s(\nu) = \sum_{i=0}^{e\nu} i \binom{e\nu - i + \ell}{\ell} = \binom{\ell + e\nu - 1}{\ell} \frac{(\ell + e\nu)(1 + \ell + e\nu)}{2 + 3\ell + \ell^2}$$

and

$$t(\nu) = \sum_{i=1}^{\nu} \binom{e\nu - ie + \ell + 1}{\ell + 1}$$

We also prove in the same section that $t(\nu)/s(\nu) \to 1/e$ as ν tends to $+\infty$, so that the best achievable bound on $|m|$ and $|r|$ for which the attack applies is:

$$\ell|r| + |m| < \frac{\ell k}{e}$$

Since $|r| = k - |m| - 3$ in PKCS#1 v1.5 we obtain the bound announced in Proposition 4. As usual when using Coppersmith's techniques, the complexity of the attack is polynomial in the dimension of the constructed lattice and in the size of the entries (see Appendix B for details on these parameters).

Given PKCS#1 v1.5's widespread use and the heuristic nature of multivariate Coppersmith-like techniques, it is important to *practically assess* our attack. We report practical experiment results in Appendix C.

6 Conclusion

Figure 2 summarizes our current knowledge of PKCS#1 v1.5 security.

$$\begin{tikzcd}
\text{UBK-CCA} \arrow[d, Rightarrow] & \text{UBK-VCA} \arrow[l, Rightarrow] \arrow[d, Rightarrow] & \text{UBK-CPA} = \text{Factoring} \arrow[l, Rightarrow] \arrow[d, Rightarrow] \\
\begin{matrix}\cancel{\text{OW-CCA}} \\ \ell = 2\ (\S\ 3)\end{matrix} \arrow[d, Rightarrow] & \begin{matrix}\cancel{\text{OW-VCA}} \\ \text{large } \ell\ ([5])\end{matrix} \arrow[l, Rightarrow] \arrow[d, Rightarrow] & \begin{matrix}\text{OW-CPA} = \text{RSA, single user } (\S\ 3) \\ \cancel{\text{OW-CPA}} \text{ small } e \text{ and large } |m|\text{, multi-user } (\S\ 5)\end{matrix} \arrow[l, Rightarrow] \arrow[d, Rightarrow] \\
\text{IND-CCA} \arrow[d, Rightarrow] & \begin{matrix}\cancel{\text{IND-VCA}} \\ \text{large } \ell\ ([5]) \\ \ell = 1\ (\S\ 4.2)\end{matrix} \arrow[l, Rightarrow] \arrow[d, Rightarrow] & \begin{matrix}\cancel{\text{IND-CPA}} \\ \text{small } e \text{ or large } |m|\ ([9])\end{matrix} \arrow[l, Rightarrow] \arrow[d, Rightarrow] \\
\text{NM-CCA} & \text{NM-VCA} \arrow[l, Rightarrow] & \cancel{\text{NM-CPA}}\ (\S\ 4.1) \arrow[l, Rightarrow]
\end{tikzcd}$$

Fig. 2. Updated Security Status for PKCS#1 v1.5

The authors regard the new flaws as an indication that the process of phasing out PKCS#1 v1.5 should be accelerated.

References

1. Baudron, O., Pointcheval, D., Stern, J.: Extended notions of security for multicast public key cryptosystems. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 499–511. Springer, Heidelberg (2000)
2. Bauer, A., Coron, J.-S., Naccache, D., Tibouchi, M., Vergnaud, D.: On the broadcast and validity-checking security of PKCS#1 v1.5 encryption. Full version of this paper. Cryptology ePrint Archive, Report 2010/135, `http://eprint.iacr.org/`
3. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)
4. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 549–570. Springer, Heidelberg (1998)
5. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)

6. Coppersmith, D.: Finding a small root of a univariate modular equation. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 155–165. Springer, Heidelberg (1996)
7. Coppersmith, D.: Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. Journal of Cryptology 10(4), 233–260 (1997)
8. 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)
9. Coron, J.-S., Naccache, D., Joye, M., Paillier, P.: New attacks on PKCS#1 v1.5 encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 369–381. Springer, Heidelberg (2000)
10. Coron, J.-S., Naccache, D., Joye, M., Paillier, P.: Universal Padding Schemes for RSA. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 226–241. Springer, Heidelberg (2002)
11. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: rsa-oaep is secure under the RSA assumption. Journal of Cryptology 17(2), 81–104 (2004)
12. Håstad, J.: Solving simultaneous modular equations of low degree. SIAM Journal on Computing 17(2), 336–341 (1988)
13. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
14. Jochemsz, E., May, A.: A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 267–282. Springer, Heidelberg (2006)
15. Kaliski, B.: PKCS#1: RSA Encryption Standard, Version 1.5, RSA Laboratories (November 1993)
16. Kaliski, B.: PKCS#1: RSA Encryption Standard, Version 2.0, RSA Laboratories (September 1998)
17. Kaliski, B.: RSA Laboratories, personal communication (October 2009)
18. Lenstra, A.K., Lenstra, H.W., Lovàsz, L.: Factoring polynomials with rational coefficients. Math. Annalen 261, 513–534 (1982)
19. Pointcheval, D.: Provable security for public-key schemes. In: Contemporary cryptology. Advanced courses in mathematics, pp. 133–190. Birkhäuser, Basel (2005)
20. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)

A Preliminaries

In this appendix, we recall a few basic definitions necessary for an accurate description of our attack. We refer the reader to the full version of this paper [2] for more details.

A.1 Public-Key Encryption

A *Public-Key Encryption Scheme* $\mathcal{P} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a collection of three (probabilistic) algorithms:

1. **A Key Generation Algorithm** $\mathcal{K}$**.** Given a security parameter $k \in \mathbb{N}$, the algorithm $\mathcal{K}$ produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys (which implicitly define a message space $\mathcal{M}$).

2. **An Encryption Algorithm $\mathcal{E}$.** Given a message $m \in \mathcal{M}$ and a public key pk, $\mathcal{E}$ produces a ciphertext $c = \mathcal{E}(\mathsf{pk}, m)$.
3. **A Decryption Algorithm $\mathcal{D}$.** Given a ciphertext c and the private key sk, $\mathcal{D}(\mathsf{sk}, c)$ recovers a plaintext m or a special symbol $\perp \notin \mathcal{M}$.

A.2 Security Definitions

Security is traditionally defined by combining an *adversarial goal* and an *attack model*. We refer to classical texts on provable security, such as [19], for precise statements of security definitions. Intuitively, a public-key encryption scheme $\mathcal{P}$ is:

– **Unbreakable (UBK).** If no adversary $\mathcal{A}$ can compute sk given pk.
– **One-Way (OW).** If no adversary $\mathcal{A}$ can recover $m \in \mathcal{M}$ given c and pk.
– **Indistinguishable (IND).** If no adversary $\mathcal{A}$ can derive significant information about $m \in \mathcal{M}$ given only c and pk. IND is sometimes alternatively referred to as "semantically secure".
– **Non-Malleable (NM).** If no adversary $\mathcal{A}$ can produce, given c and pk, a new ciphertext corresponding to a plaintext m' meaningfully related to m.

Security notions for encryption schemes are obtained by combining an adversarial goal with an *attack model*:

– **Chosen-Plaintext Attack (CPA).** $\mathcal{A}$ is given nothing more than pk.
– **Validating-Checking Attack (VCA).** $\mathcal{A}$ is given access to a validity-checking oracle indicating only if a given ciphertext is valid or not (*i.e.* returning the bit $\mathcal{D}(\mathsf{sk}, c) \overset{?}{\neq} \perp$). Note that this is *not* the same thing as a Plaintext-Checking Attack (PCA) [19].
– **Chosen-Ciphertext Attack (CCA).** $\mathcal{A}$ has access to a decryption oracle.

Whenever oracle access is used $\mathcal{A}$ cannot submit to the oracle the challenge ciphertext he has to attack. These definitions are classical and we refer the reader to [19] for more details.

We denote security notions positively: *e.g.* OW-$\mathsf{VCA}[\mathcal{S}]$ is the problem of contradicting the one-wayness of scheme $\mathcal{S}$ under validity-checking attack. This convention permits the easy description of hierarchies [4] between security notions using reductions (see Figure 1 as an example). When oracle access is permitted, we denote by ℓ-GOAL-$\mathsf{ATTACK}[\mathcal{S}]$ the problem of contradicting $\mathsf{GOAL} \in \{\mathsf{UBK}, \mathsf{OW}, \mathsf{IND}, \mathsf{NM}\}$ in less than ℓ oracle queries under an $\mathsf{ATTACK} \in \{\mathsf{VCA}, \mathsf{CCA}\}$.

B Finding Small Modular Roots of a Multivariate Polynomial

The problem of solving modular polynomial equations is believed to be difficult in the general case. Nevertheless, when we restrict the problem to finding small roots only, the problem becomes easier to solve. Indeed, in 1996, Coppersmith [6] introduced a technique, based on lattice reduction, allowing to recover the root of a univariate modular polynomial provided that this root is small enough. This construction was reformulated in simpler terms by Howgrave-Graham [13] and its extensions to more variables found numerous cryptanalytic applications.

B.1 Coppersmith's Technique

Starting from a polynomial f modulo a known composite integer N, the idea behind Coppersmith's method is to construct a set of polynomials $h_1, \ldots, h_n$ sharing the same sought root over the integers. If the number of these generated polynomials is sufficiently large (greater than the number of variables) and under the assumption that all resultant computations lead to non-zero results, then the root can easily be recovered. Note that this assumption makes the method heuristic.

A sufficient condition ensuring that the polynomials $h_1, \ldots, h_n$ share a common root in $\mathbb{Z}$ was formulated by Howgrave-Graham.

Lemma 2 (Howgrave-Graham [13]). *Let $h \in \mathbb{Z}[x_1, \ldots, x_n]$ be an integer polynomial that consists of at most ω monomials. Suppose that*

1. $h(x_{01}, \ldots, x_{0n}) \equiv 0 \bmod N$ *for some* $|x_{01}| < X_1, \ldots, |x_{0n}| < X_n$
2. $\|h(x_1 X_1, \ldots, x_n X_n)\| < \frac{N}{\sqrt{\omega}}$.

Then $h(x_{01}, \ldots, x_{0n}) = 0$ *holds over the integers.*

The problem can thus be reduced to finding polynomials $h_1, \ldots, h_n$ of small norm having the same modular root as f. This can be achieved by representing polynomials as coefficient vectors (using a suitable ordering on monomials) and using lattice reduction techniques such as LLL [18] to search for small vectors in a lattice spanned by polynomials which are known to have the sought modular root.

If $\mathcal{L}$ is a lattice of polynomials consisting of at most ω monomials and all having the same modular root as f, then the condition

$$2^{\frac{\omega(\omega-1)}{4(\omega+1-n)}} \det(\mathcal{L})^{\frac{1}{(\omega+1-n)}} < \frac{N}{\sqrt{\omega}} \tag{4}$$

ensures first n polynomials obtained by applying LLL to the lattice $\mathcal{L}$ match Howgrave-Graham's bound. In the analysis, we let terms that do not depend on N contribute to an error term ε, and simply use the determinant condition $\det(\mathcal{L}) \leq N^{w+1-n}$.

B.2 Lattice Construction

A variety of methods for constructing the lattice $\mathcal{L}$ have been proposed in the literature. In what follows, we choose to rely on the technique introduced by Jochemsz and May in [14].

Recall that we have a polynomial f with an unknown root $\boldsymbol{x_0} = (x_{01}, \ldots, x_{0n})$ modulo some composite integer N whose factorization is unknown. This root is *small* in the sense that each of its components is bounded: $|x_{0i}| < X_i$ for $i \in \{1, \ldots, n\}$. We denote by λ the leading monomial of the polynomial f and by $\mathcal{M}(f)$ the set of monomials appearing in f. Of course, λ can be assumed to be monic as otherwise one simply has to multiply f by the modular inverse of its initial coefficient.

Given $\varepsilon > 0$, we fix an integer $\nu = \nu(\varepsilon)$ and without loss of generality we assume that $\mathcal{M}(f^j) \subseteq \mathcal{M}(f^\nu)$ for $j \in \{1, \ldots, \nu - 1\}$. If k is an integer between 0 and $\nu + 1$, we define the set M_k as $\mathcal{M}(f^\nu) \cap \lambda^k \mathcal{M}(f^{\nu-k})$ (in particular $M_0 = \mathcal{M}(f^\nu)$ and $M_{\nu+1} = \varnothing$). Next, we define the following *shift polynomials*:

$$g_{i_1 \ldots i_n}(x_1, \ldots, x_n) = \frac{x_1^{i_1} \cdots x_n^{i_n}}{\lambda^k} f^k N^{\nu-k}$$

for $k \in \{0, \ldots, \nu\}$ and $x_1^{i_1} \cdots x_n^{i_n} \in M_k \setminus M_{k+1}$. By definition, all polynomials g have the root $(x_{01}, \ldots, x_{0n})$ modulo N^ν. We can now define $\mathcal{L}$ as the lattice generated by the coefficient vectors of all polynomials $g_{i_1 \ldots i_n}(x_1 X_1, \ldots, x_n X_n)$. If the monomial ordering has been chosen correctly, the matrix corresponding to that lattice is lower triangular and the determinant becomes easy to compute. Indeed, the diagonal elements are those corresponding to the monomial λ^k in f^k for each row. Therefore, the diagonal terms of the matrix are $X_1^{i_1} \cdots X_n^{i_n} N^{\nu-k}$ for $k \in \{0, \ldots, \nu\}$ and $x_1^{i_1} \ldots x_n^{i_n} \in M_k \setminus M_{k+1}$. By doing a simple computation and neglecting low order terms, one can finally reduce the condition (4) to the following new one:

$$\prod_{j=1}^{n} X_j^{s_j} < N^{s_N} \quad \text{for} \quad \begin{cases} s_j = \sum_{x_1^{i_1} \cdots x_n^{i_n} \in M_0} i_j \quad (1 \le j \le n) \\ s_N = \sum_{k=1}^{\nu} |M_k| \end{cases} \tag{5}$$

This formula expresses an asymptotic condition on the bounds $X_1, \ldots, X_n$ allowing to recover the root in polynomial time.

Remark 1. The method outlined above is what Jochemsz and May called the "*basic strategy*"; they also proposed an "extended strategy" in which we can use extra shifts of a certain variable and replace M_k for instance by $M_k = \bigcup_{j=1}^{t} x_1^j \left(\mathcal{M}(f^\nu) \cap \lambda^k \mathcal{M}(f^{\nu-k})\right)$ for some well-chosen parameter t.

B.3 The Jochemsz-May Lattice in Broadcast PKCS#1 v1.5

Let us examine what the lattice $\mathcal{L}$ looks like in the particular setting of broadcast PKCS#1 v1.5 encryption.

Recall from section 5.1 that recovering m from $c_1, \ldots, c_\ell$ reduces to finding the root $(x_0, y_{0,1}, \ldots, y_{0,\ell}) = (m, r_1, \ldots, r_\ell)$ of the following modular polynomial:

$$f(x, y_1, \ldots, y_\ell) = u_1(x + Ay_1 + B)^e + \cdots + u_\ell(x + Ay_\ell + B)^e - C \mod N$$

We know that this root satisfies the bounds $|x_0| < X$ and $|y_{0i}| < Y$ for all $i \in \{1, \ldots, \ell\}$ with $X = 2^{8|m|}$ and $Y = 2^{8|r|}$. We examine how the Jochemsz-May bounds described in the previous section translate into bounds on $|m|$ and $|r|$ allowing the message to be recovered in polynomial time.

Form of the Sets M_k. The analysis' first step consists in describing the sets M_k. Note first that the set of monomials $\mathcal{M}(f)$ is included in $\{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leq e\}$. In other words, the geometrical shape of the polynomial f is included in a "pyramid" of dimension $\ell + 1$ of monomials with total degree less than e. We choose the *deglex* monomial order, according to which the leading monomial of f is x^e. The sets M_k can then be described as follows:

$$\begin{aligned}
M_0 &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leq e\nu\} \\
M_1 &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leq e\nu \text{ with } a \geq e\} \\
&\vdots \\
M_\nu &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leq e\nu \text{ with } a \geq e\nu\}
\end{aligned}$$

which makes it easy to count the number of monomials in each of them.

Condition on the Bounds. Given the above description, we can evaluate the quantities s_j and s_N of equation (5) as follows. First, by symmetry, $s_i, s_{j_1}, \ldots s_{j_\ell}$ are all equal to:

$$s(\nu) = \sum_{i=0}^{e\nu} \frac{i(e\nu - i + 1)(e\nu - i + 2) \cdots (e\nu - i + \ell)}{\ell!}$$

Furthermore, we have:

$$s_N = t(\nu) = \sum_{i=1}^{\nu} \frac{(e\nu - ie + 1)(e\nu - ie + 2) \cdots (e\nu - ie + \ell + 1)}{(\ell + 1)!}$$

Condition (5) can then be rewritten as $X^{s(\nu)} Y^{\ell s(\nu)} < N^{t(\nu)}$, and since N is of byte size ℓk, this gives:

$$\ell|r| + |m| < \frac{t(\nu)}{s(\nu)} \cdot \ell k$$

Asymptotic Bound. The functions $s(\nu)$ and $t(\nu)$ are polynomials in ν. Hence, it suffices to evaluate their leading coefficients to obtain an asymptotic estimate as $\nu \to +\infty$. Note further that $s(\nu)$ and $t(\nu)$ are easily expressed in terms of the antidifference operator, which takes a polynomial $P(X)$ to the polynomial $\sigma(P)(X)$ defined by $\sigma(P)(j) = \sum_{i=1}^{j} P(i)$ for $j \in \mathbb{N}$. Indeed:

$$\begin{aligned}
s(\nu) &= \sum_{i=0}^{e\nu} (e\nu - i) P(i) = e\nu \cdot \sigma(P)(e\nu) - \sigma(XP)(e\nu) + e\nu \\
&\quad \text{with} \quad P(X) = \frac{(X+1) \cdots (X+\ell)}{\ell!} \\
t(\nu) &= \sum_{i=1}^{\nu} Q(i) = \sigma(Q)(\nu) \\
&\quad \text{with} \quad Q(X) = \frac{(eX - e + 1) \ldots (eX - e + \ell + 1)}{(\ell + 1)!}
\end{aligned}$$

Now it is easily seen that if the leading coefficient of P is $c_d X^d$, the leading coefficient of $\sigma(P)$ is $c_d X^{d+1}/(d+1)$. It follows that, as $\nu \to +\infty$, we have:

$$s(\nu) \sim e\nu \cdot \frac{(e\nu)^{\ell+1}}{\ell!(\ell+1)} - \frac{(e\nu)^{\ell+2}}{\ell!(\ell+2)} = \frac{(e\nu)^{\ell+2}}{(\ell+2)!} \quad \text{and} \quad t(\nu) \sim \frac{e^{\ell+1}\nu^{\ell+2}}{(\ell+2)!}$$

In particular, $t(\nu)/s(\nu) \to 1/e$ when $\nu \to +\infty$. Thus, the best asymptotic bound on $|m|$ and $|r|$ for which the attack is theoretically possible is:

$$\ell|r| + |m| < \frac{\ell k}{e}$$

C Broadcast Attack Experimental Results

Given PKCS#1 v1.5's widespread use and the heuristic nature of Coppersmith's techniques in the multivariate case, it is important to *practically assess* our attack. In particular, one of the main questions remains to know how many ciphertexts an attacker really needs *in practice* to recover m. In the particular instance $\{\log_2 N = 1024, e = 3\}$, the number of required ciphertexts is, in fact, really low.

Corollary 1. *If a* PKCS#1 v1.5 *user encrypts the same message m with* 64-*bit randomizers to multiple recipients using* 1024-*bit moduli and $e = 3$, then there exists a heuristic polynomial time algorithm that recovers m from $\ell = 4$ ciphertexts.*

Proof. This is a direct consequence of Proposition 4, given that, for 1024-bit moduli and 64-bit randomizers, message size is equal to 936 bits.

These parameters, corresponding to optimal message size and encryption speed for 1024-bit moduli, are quite realistic and widely implemented. The practical implications of this result are potentially serious.

C.1 Partial Information

Consider an attacker who does not collect all the ℓ required ciphertexts. In that specific case, even if m can not be fully recovered, the attacker can nevertheless obtain partial information on m. In particular, in a scenario where m would not be of full size and would have been previously padded with zero bits (*e.g.* using an AES key), the attack can still be performed.

C.2 Practical Implementations

To check the applicability of the attack, we investigated three configurations: An attacker having access to two, three and four ciphertexts. Before implementing the attack in each scenario, we first evaluated the dimension of the corresponding lattices (for reasonably small values of the parameter ν) and then expressed the number of bits on m that should be recovered in practice. The results obtained for 1024-bit moduli and $e = 3$ are shown in the following table.

	$\ell=2$		$\ell=3$		$\ell=4$	
ν	$\dim(\mathcal{L})$	$\lvert m\rvert$	$\dim(\mathcal{L})$	$\lvert m\rvert$	$\dim(\mathcal{L})$	$\lvert m\rvert$
2	84	213	210	246	462	249
3	220	306	715	395	2002	451
4	455	359	1820	483	6188	578
5	816	394	3876	542	15504	664
6	1330	418	7315	584	33649	726
7	2024	435	12650	615	65780	773
10	5456	469	46376	675	324632	863

As we can see, the number of bits of m that we are able to recover increases with ν, and approaches 936 bits for $(\ell, \nu) = (4, 10)$. Unfortunately, the dimensions of the constructed lattices are often quite impractical. Indeed, size often turns out to exceed 1000, making lattice reduction unfeasible in practice. As a result, and because we had limited processor time at our disposal, we only ran practical experiments in the small cases, namely $\ell = 2$ and $\nu = 2, 3$.

Experiments have been performed on a hepta-processor Intel Xeon clocked at 1.86GHz. Each test was done in the same way: construction of the appropriate lattice, LLL-reduction and then extraction of short vectors. Although one only theoretically requires a number of vectors equal to the number of variables, in practice we decided to take as much vectors as possible to increase the attack's success odds. Most CPU time was claimed by the LLL-reduction step (approximately 3 hours for $(\ell, \nu) = (2, 3)$). With 1024-bit moduli and 64-bit randomizers, we managed to recover a 115-bit message (padded with zero MSBs).

C.3 Toy example

Here is a toy example for 150-bit moduli and 5-bit m, corresponding to a lattice of dimension 84. The first 50 vectors corresponded to polynomials having the desired root over the integers. We then took all these polynomials and computed a Gröbner basis of the ideal they generated. The results were the following:

$$N_i = 150 \text{ bits}, m = 5 \text{ bits}, r = 6 \text{ bits}, e = 3, \nu = 2$$
$$f(x, y, z) \text{ with modular root } (24, 58, 34)$$

$$\begin{cases} p_1 = z^4 + 512z^3 + 98304z^2 + 86093442y - 94710946z - 1908346880 \\ p_2 = yz^2 - \frac{89}{81}z^3 + 256yz - \frac{7936}{27}z^2 + 16384y - \frac{573440}{27}z - \frac{33783328}{81} \\ p_3 = y^2 - \frac{62}{27}yz + \frac{961}{729}z^2 - \frac{1024}{27}y + \frac{31744}{729}z + \frac{262144}{729} \\ p_4 = x - 55297y + 63489z + 1048576 \end{cases}$$

The Gröbner basis computation does not allow us to directly recover the message m, since the corresponding subvariety is not of dimension zero. In fact, we commonly faced problems of algebraic dependence between the resulting polynomials (hence our choice to take a large number of polynomials, rather than the first few, to compute the Gröbner basis). Nevertheless, it was usually possible to recover the message, as the polynomials in the Gröbner basis had a very simple form. In this particular case, for instance, p_4 is affine and p_3 can be written as $(ay + bz + c)^2$, making it easy to recover the common root.

How to Construct Interval Encryption from Binary Tree Encryption

Huang Lin*, Zhenfu Cao, Xiaohui Liang, Muxin Zhou,
Haojin Zhu, and Dongsheng Xing

Department of Computer Science and Engineering, Shanghai Jiao Tong University
faustlin@sjtu.edu.cn*

Abstract. In a broadcast encryption system with a total of n users, each user is assigned with a unique index $i \in [1, n]$. An encryptor can choose a receiver set $S \subseteq [1, n]$ freely and encrypt a message for the recipients in S such that only those receivers can open the message. The transmission overload of most previous broadcast encryption systems grows in line with the number of revoked users r and thus they are suitable for the scenario where the target receiver set is large when $r \ll n$ holds. Some other recently proposed constructions for arbitrary receiver set require a unreasonably large user storage and long decryption time. On the other hand, it is observed that, in a practical broadcast encryption system, the receiver set can be regarded as a collection of k natural intervals, where the interval number k should be much less than r for most cases. This observation motivates us to introduce a novel type of encryption, called interval encryption, which could realize a more efficient broadcast encryption. To achieve this, we first present a generic way to transform a binary tree encryption scheme into interval encryption. One concrete instantiation of this method based on the hierarchical identity based encryption scheme by Boneh et al. only requires a $O(k)$ transmission cost and $O(\log n)$ private storage consumption, while the decryption is dominated by $O(\log n)$ group operations. With detailed performance analysis, we demonstrate that the proposed interval encryption strategy has the superiority on improved efficiency and thus is expected to serve as a more efficient solution in more cases than the traditional systems in practice. Interestingly, our methodology can also be employed to transform a fully secure hierarchical identity based encryption scheme proposed by Lewko and Waters into an adaptively secure interval encryption scheme with a $O(k)$ transmission cost and $O(\log n)$ private storage consumption. Finally, we also discuss several other promising applications of interval encryption.

Keywords: Interval encryption, Public key broadcast encryption, Binary tree encryption, Hierarchical IBE.

1 Introduction

A broadcast encryption (BE) scheme enables a broadcaster to choose a subset S of n users, who are listening on the broadcast channel and encrypt a message for this subset. Any user in S is allowed to successfully decrypt the message while even if all the users outside of S collude together they can not obtain any useful information on the broadcast message. In the following, we also use r to represent the number of revoked users, i.e., $r = n - |S|$ where $|S|$ is the size of S. Compared with a private key broadcast encryption scheme [25,1], a public key broadcast encryption has the benefit that there is no need for the users to pre-share any private information. Therefore, in this study, we mainly focus on pubic key broadcast encryption. Three efficiency parameters of a broadcast encryption scheme are of our major concern: the transmission cost, user storage, and the decryption time.

1.1 Related Work

The transmission overload of most current public key broadcast encryption constructions will grow along with increase of the revocation number r. Naor et al. [22] presented a BE construction (NNL method) with an average ciphertext size of $1.38r$ and private key size $O(\log^2 n)$. The private key size is further improved to $O(\log^{1+\epsilon} n), 0 < \epsilon < 1$ in HS construction [18], where the ciphertext size blows up with a $\frac{1}{\epsilon}$ factor. The private key size is further improved to $O(\log n)$ by Goodrich et al. [16]. Dodis and Fazio [12] presented a generic method (DF transformation) to transfer the NNL method and HS construction into a public key broadcast system using hierarchical identity based encryption (HIBE). The transmission overload remains unchanged and the private key consists of $O(\log^2 n)$ and $O(\log^{1+\epsilon} n)$ HIBE node secret keys if DF transformation is instantiated with BBG HIBE [3]. The security is reduced to standard Decisional BDHE assumption and the decryption time cost is $O(\log n)$. The decryption time is then improved to constant by Liu and Teng [21]. However, their security is reduced to decisional BDH assumption in the random oracle model. Recently, Sahai and Waters proposed a broadcast encryption system with a transmission overload linearly dependent on r and constant storage cost. However, the decryption cost is linearly dependent on r and the security is reduced to a complex assumption called q-MEBDH assumption. Actually, it has been pointed out in [19] that at least one key per each revoked user should be included in the transmission overhead and hence r might be the lower bound of the transmission overload in any broadcast encryption scheme with reasonable decryption computational and storage cost. Therefore, constructing a BE system with a transmission overload lower than r as well as reasonable user storage and computational cost is still an open problem, which is one of the major motivation of this paper.

On the other hand, there are two major application scenarios [5] for broadcast encryption: applications where we broadcast to large sets, namely sets of size $n - r$ for $r \ll n$ and applications where we broadcast to small sets, namely sets of size $|S| \ll n$. Apparently, a broadcast encryption system with a transmission overload dependent on r is not efficient when r grows, and especially it fails to be an optimal choice for the second kind of application where r is very close to n. Before BGW proposed their construction [5], the only suitable solution for the latter scenario is the trivial solution, i.e., encrypting the message under each recipient's key.

In order to construct a BE scheme suitable for *arbitrary* receiver sets, we need to break the barrier of r. BGW [5] proposed an elegant BE scheme with constant size ciphertext as the first attempt to solve this problem. Although the ciphertext and private key size of their construction is constant, the public key material is linearly dependent on n. The public key must be accessible to any decryptor, which implies a high storage cost of size $O(n)$. This makes their system unsuitable for the application scenario where users have only limited storage capability [24]. Their underlying assumption is the standard Decisional BDHE assumption. Later, Delerablee [11] proposed a BE construction where the public key size depends on the maximum size of S while both ciphertext and private key remain constant size. However, this still does not serve as an efficient solution for applications where the receiver set is large, namely $r \ll n$. The security of this construction is reduced to a complex assumption called GDHE assumption in the random oracle model. Besides, the decryption of both constructions is not efficient. The decryption cost of the BGW construction depends on n, and the decryption of Delerablee's construction requires $O(|S|)$ operations.

1.2 Our Contribution

In this paper, we study this problem from a brand-new angle and a more practical point of view. The basic motivation comes from the following observation: in a broadcast encryption system with n users, where each user is assigned with an index $i \in [1, n]$. The receiver set S can be regarded as a collection of k intervals. Considering the fact that the number of intervals containing in S is always less than $r + 1$ and in the best cases k could even be much less than r, the system performance can be dramatically increased if the transmission overhead of the broadcast encryption system is only determined by the interval number k while irrelevant of r. In this study, we will use more detailed performance analysis and simulation to show that a BE construction based on k is always more efficient than the previous scheme dependent on r, and suitable for more cases in practice.

In order to realize a broadcast encryption system with a transmission overload dependent on k, this paper proposes a new type of encryption called interval encryption. In interval encryption, a message is encrypted under a collection of natural intervals $S = \bigcup_{j=1}^{k} NI_j$, where NI_j is a natural interval in $[1, n]$. Each receiver is identified by a unique natural number $i \in [1, n]$ and assigned with the respective private key. The decryption is successful if and only if the natural number i belongs to S.

We present a generic methodology which can transfer a series of binary tree encryption scheme into interval encryption. We illustrate the basic methodology using the BBG HIBE scheme [3]. The construction achieves a ciphertext size of $O(k)$, and $O(\log n)$ private storage. The decryption is dominated by at most $O(\log n)$ group operations. The security is reduced to the Decisional BDHE assumption. We note that one of the best public key BE schemes under this assumption is the DF transformation of the HS construction which requires a transmission overload of $O(r/\epsilon)$ size and the private key consists of $O(\log^{1+\epsilon} n)$ HIBE node secret keys, where $0 < \epsilon < 1$.

We also apply our basic methodology to the fully secure HIBE [20] scheme proposed by Lewko and Waters to present an adaptively secure interval encryption scheme. Gentry and Waters [15] proposed the first adaptively secure broadcast encryption scheme under a complex bilinear assumption. The public parameter size of their construction is of $O(|S|)$. The private key size is constant, and the ciphertext size of their construction is of $O(max|S|)$. After that, Waters [26] gave the first short ciphertext adaptively secure broadcast encryption system under static (i.e. non q-based) assumptions. However, both of the public parameter and private key size are linearly dependent on n. The public parameter of our construction is of size $O(\log n)$ and the ciphertext size is of $O(k)$. It only requires $O(\log n)$ private storage. In other words, our construction serves as one of the most efficient adaptively secure broadcast encryption systems. Besides, our construction also reduces its security to static assumptions.

Since we consider the proposal of this new concept and the corresponding methodology one of our major contributions, an inclusive extended interval encryption is proposed as another illustration of the power of our basic methodology. A message is encrypted under a collection of intervals $S = \bigcup_{j=1}^{k} NI_j$ in this extended construction. A user's private key corresponds to a certain interval NI_ω. The decryption is successful if and only if there's at least one interval NI_j, $j \in [1, k]$ such that $NI_\omega \subseteq NI_j$. The construction also provides user with delegation capability. We also discuss several interesting applications of interval encryption. In particular, we propose a useful concept of range attribute based encryption and present an efficient construction from interval encryption.

1.3 Organization

At first, some preliminaries will be given in Section 2. As an important step of understanding the primitive idea of our construction, we'll introduce the notion of binary tree encryption and a different view on forward secure encryption constructed from binary tree encryption in Section 3. The notations used in this paper are introduced in Section 4. A generic transformation from binary tree encryption to interval encryption will be presented in Section 5. In Section 6, we'll give our concrete instantiations based on BBG HIBE and then discuss the system performance in details. In section 7, we introduce how to present an inclusive extended interval encryption using our method. How to construct an efficient adaptively secure interval encryption scheme is shown in section 8. At last, some interesting applications and extensions of interval encryption, including how to construct a range attribute based encryption from interval encryption, are given with some open problems in Section 9.

2 Preliminaries

2.1 Assumptions

Bilinear maps [23] are crucial to our construction. A pairing is an efficiently computable, non-degenerate function, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, with the bilinearity property that $\hat{e}(g^r, g^s) = \hat{e}(g, g)^{rs}$. Here, $\mathbb{G}_1$, and $\mathbb{G}_2$ are all multiplicative groups of prime order p, respectively generated by g and $\hat{e}(g, g)$.

The security proof of our constructions relies on the Decisional $d+1$ BDHE assumption, which can be stated as [8]: Given a tuple $[h, g, g^\alpha, g^{(\alpha^2)}, \cdots, g^{\alpha^d}, g^{(\alpha^{d+2})}, \cdots, g^{(\alpha^{2d})}, Z] \in \mathbb{G}_1^{2d+1} \times \mathbb{G}_2$ for a random exponent $\alpha \in \mathbb{Z}_p$, decide whether $Z = \hat{e}(g, h)^{\alpha^{d+1}}$.

2.2 Security Definitions

Our construction is a Key Encapsulation Mechanism (KEM)[1], thus long messages can be encrypted under a short symmetric key. An interval encryption scheme is made up of four randomized algorithms:

Setup(n)**.** Takes as input a natural interval $[1, n]$. It outputs a public key PK and the system master key SK_ε.

PvkGen(ω, SK_ε)**.** Takes as input a natural number $\omega \in [1, n]$ and the system master key SK_ε. It outputs a private key D_ω.

Encrypt(S, PK)**.** Takes as input a public key PK, and a k-wise natural interval set $S = \bigcup_{j=1}^{k} NI_j$ where $NI_j = [l_j, r_j]$ satisfying $1 \le l_1 \le r_1 < l_2 \le r_2 \cdots < l_k \le r_k \le n$. For $j \in [1, k]$, it outputs k pairs $\{\text{Hdr}_j, K_j\}$. We call $\text{Hdr} = \{\text{Hdr}_j\}_{j=1}^{k}$ the header and $K = \{K_j\}_{j=1}^{k}$ the message encryption keys.

Let M be a message that should be decipherable precisely by the receivers holding the private key corresponding to $\omega \in S$. For $j \in [1, k]$, let C_j be the encryption of M under the message encryption key K_j. Let C_M be the collection of these encryption, namely $C_M = \{C_j\}_{j=1}^{k}$. The whole ciphertext consists of (S, Hdr, C_M).

Decrypt ($S, \omega, D_\omega, \text{Hdr}, PK$) **.** Takes as input a k-wise natural interval set $S = \bigcup_{j=1}^{k} NI_j$ and the private key D_ω for a natural number $\omega \in [1, n]$, a header Hdr, a public key PK. If $\omega \in NI_j$, $1 \le j \le k$, then the algorithm outputs the corresponding message encryption key $K_j \in \mathcal{K}$.

We say the system to be correct, if and only if that for all k-wise natural interval sets $S = \bigcup_{j=1}^{k} NI_j$ and natural numbers $\omega \in NI_j$ (where $j \in [1, k]$), if $PK \overset{R}{\leftarrow}$ **Setup**(n), $D_\omega \overset{R}{\leftarrow}$ **PvkGen**(ω, SK_ε) and $(\text{Hdr}, K) \overset{R}{\leftarrow}$ **Encrypt**(S, PK), then **Decrypt**(S, ω, D_ω, Hdr, PK)=K_j. The concept of interval encryption is close to private linear broadcast encryption (PLBE) mentioned in [7], and can be viewed as an extension of PLBE.

Semantic Security(IND-sI-CPA) **.** The selective interval game is very similar to that of BE [11], and it forms as follow:

Init. The adversary outputs a k-wise natural interval set $S^* = \bigcup_{j=1}^{k} NI_j^*$, where $NI_j^* = [l_j^*, r_j^*]$ satisfying $1 \le l_1^* \le r_1^* < l_2^* \le r_2^* \cdots < l_k^* \le r_k^* \le n$, which it wishes to attack.

Setup. The challenger runs **Setup**(n) to obtain a public key PK for the adversary.

Phase 1. The adversary issues query for private key of $\omega \notin S^*$.

Challenge . The challenger runs algorithm *Encrypt* to obtain $(\text{Hdr}^*, K) \overset{R}{\leftarrow}$ *Encrypt* (S^*, PK) where $K \in \mathcal{K}^k$. Next, the challenger picks a random $\beta \in \{0, 1\}$. It sets $K^* = K$ if $\beta = 1$ and sets K^* to a random string of length equal to $|K|$ otherwise. It then sends Hdr^*, K^* to the adversary.

Phase 2. Same as phase 1.

Guess . The adversary outputs its guess $\beta' \in \{0, 1\}$ for β and wins the game if $\beta' = \beta$.

[1] We adopt KEM for the ease of comparison since all the BE constructions in the literature employ the same mechanism.

The adversary's advantage is the absolute value of the difference between its success probability and $\frac{1}{2}$.

Definition 1. *An interval encryption scheme is selective-interval chosen plaintext secure (IND-sI-CPA) if all polynomial time adversaries have at most a negligible advantage in winning the above security game.*

The adaptive CPA security can be defined in a similar way except that there is no **Init** stage in the adaptive game and the challenge interval S^* in the **Challenge** stage should be provided under the restriction that none of the identities ω for the key queries of **Phase 1** and **Phase 2** belongs to S^*, i.e., $\omega \notin S^*$.

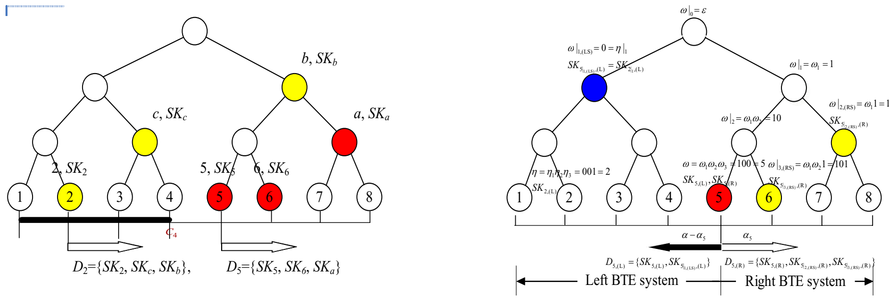
The ultimate security goal is to realize IND-CCA security where the adversary doesn't need to choose the interval set at the beginning and is provided with a decryption oracle. However, this paper concentrates on IND-sI-CPA security, and leaves the formal definition of IND-CCA security in the full version.

3 Binary Tree Encryption and a Different View on Forward Secure Encryption

The concept of binary tree encryption (BTE) was first proposed by Canetti, et al [10]. BTE is a relaxation of hierarchical identity-based encryption (HIBE) [14]. As in HIBE, a "master" public key PK is associated with a binary tree in BTE; each node ω in this tree has a corresponding secret key SK_ω. To encrypt a message "targeted" for some node, one uses both PK and the name of the target node; the resulting ciphertext can then be decrypted using the secret key of the target node. Moreover, as in HIBE the secret key of any node can be employed to derive the secret keys for the children of that node. The only difference between HIBE and BTE is that the latter insists on a *binary* tree, where each non-leaf node only has two child nodes.

Technically speaking, forward secure encryption (FSE) is an elegant application of BTE. Let the depth of a binary tree be d which implies it has $n = 2^d$ leaf nodes. In a FSE scheme, the lifetime of a system is divided into $n = 2^d$ time periods, each of which is associated with a unique leaf node of the tree. A user holding a private key for time period ω can open all the messages encrypted under the subsequent time periods, namely $\omega' \in [\omega, n]$. The private key D_ω in a FSE construction contains the node secret keys SK_ω for the leaf node ω as well as node secret keys for the right siblings of the nodes on the path from the root to node ω, where all these node secret keys come from the underlying BTE scheme. To encrypt a message for a certain period ω', one uses both PK and the name of respective leaf node ω' as in the BTE scheme; the resulting ciphertext can then be decrypted using node secret key $SK_{\omega'}$, which is also similar to the BTE scheme. As shown in Fig. 1. a, a private key D_2 containing the node secret keys SK_2, SK_c, SK_b can be used to derive all the node secret keys for leaf nodes falling into the interval $[2, 8]$. Therefore, D_2 can be used to open all the messages encrypted under time periods in the interval $[2, 8]$.

Indeed, forward secure encryption can be viewed as a special case of interval encryption. As shown in Fig. 1. a, if we use ciphertext C_4 encrypted under leaf node 4 to represent the interval $[1, 4]$, then only the private key for time period $\omega \in [1, 4]$ can



b, SK_b
c, SK_c
a, SK_a
$2, SK_2$
$5, SK_5$
$6, SK_6$
1
2
3
4
5
6
7
8
C_4
$D_2=\{SK_2, SK_c, SK_b\}$,
$D_5=\{SK_5, SK_6, SK_a\}$
$\omega|_0 = \varepsilon$
$\omega|_{1,(LS)} = 0 = \eta|_1$
$SK_{5|_{1,(LS)},(L)} = SK_{2|_1,(L)}$
$\omega|_1 = \omega_1 = 1$
$\omega|_{2,(RS)} = \omega_1 1 = 11,$
$SK_{5|_{2,(RS)},(R)}$
$\omega|_2 = \omega_1\omega_2 = 10$
$\eta = \eta_1\eta_2\eta_3 = 001 = 2$
$SK_{2,(L)}$
$\omega = \omega_1\omega_2\omega_3 = 100 = 5$
$SK_{5,(L)}, SK_{5,(R)}$
$\omega|_{3,(RS)} = \omega_1\omega_2 1 = 101$
$SK_{5|_{3,(RS)},(R)}$
$\alpha - \alpha_5$
α_5
$D_{5,(L)} = \{SK_{5,(L)}, SK_{5|_{1,(LS)},(L)}\}$
$D_{5,(R)} = \{SK_{5,(R)}, SK_{5|_{2,(RS)},(R)}, SK_{5|_{3,(RS)},(R)}\}$
Left BTE system
Right BTE system

Fig. 1. (a). The key distribution mode of forward secure encryption, C_4 represents the interval $[1, 4]$, and the private key for a user ω can be used to derive the node secret keys for all the nodes in the interval $[\omega, n]$ (b). Key distribution mode of interval encryption: we let $\omega = 5$ here. The respective private key D_5 contains left private key$D_{5,(L)}$ and right private key $D_{5,(R)}$. $D_{5,(L)} = \{SK_{5,(L)}, SK_{5|_{1,(LS)},(L)}\}$ which are derived from left master key $\alpha - \alpha_5$. Similarly, we have $D_{5,(R)} = \{SK_{5,(R)}, SK_{5|_{2,(RS)},(R)}, SK_{5|_{3,(RS)},(R)}\}$ derived from right master key α_5. Let the left bound η of an interval be 2 here, then $SK_{2,(L)}$ can be derived from $SK_{2|_1,(L)}$ which is equal to $SK_{5|_{1,(LS)},(L)}$ belonging to $D_{5,(L)}$.

be used to open the message, e.g., D_2 could be used for the decryption of C_4 because SK_4 can be derived from SK_c, which belongs to D_2. However, D_5 cannot be used for decrypting C_4 for it is impossible to deduce SK_4 from any node secret keys included in D_5.

In the remainder of this paper, we use a right direction arrow from a certain leaf node (or the corresponding index in the axis) to denote this particular private key distribution mode. A right direction arrow from a leaf node ω means that all the node secret keys of the leaf nodes in the interval $[\omega, n]$ are computable from its own private key. Therefore, this private key can be used to open all the message encrypted under these nodes. Besides, we also use a left direction arrow from a leaf node ω to denote an opposite decryption ability, namely the respective private key can be used to open all the messages encrypted under the leaf nodes in the interval $[1, \omega]$. It is feasible by simply assigning a user with the node secret keys for node ω as well as node secret keys for the left siblings of all the nodes on the path from the root to ω. Generally speaking, a FSE construction is treated as a special interval encryption scheme in which the encryptor can set the interval form as $[1, j]$. The upper bound j depends on which leaf node the ciphertext corresponds to. Now, our goal is to realize an interval encryption scheme covering multiple intervals, each of which has two freely chosen bounds determined by the encryptor.

4 Notation

We inherit most notations from the underlying BTE and FSE [9] construction. Recall that d denotes the depth of the tree, and $n = 2^d$ is the number of leaf nodes. We set the root node to be ε by convention. The other nodes on the tree have an associated name chosen from $\{0, 1\}^{\leq d}$. The left child of a node is concatenated with 0, and the right child is concatenated with 1. Therefore each leaf node will also have an associated binary name $[\omega_1\omega_2\cdots\omega_d]$. We also let a natural number $\omega \in [1, n]$ associate with the ω-th leaf node of the binary tree (starting from left to right). We implicitly let $\omega = [\omega_1\omega_2\cdots\omega_d]$ in the remainder of this paper. The j-bit prefix of a string $\omega = [\omega_1\omega_2\cdots\omega_d]$ is denoted by $\omega|_j$, namely $\omega|_j$=$[\omega_1\omega_2\cdots\omega_j]$. We implicitly set $\omega|_0 = \varepsilon$ and $\omega|_d = \omega$. It is easy to observe that a set of nodes $\omega|_j, j \in [1, d]$ corresponds to the nodes on the path from the root to the leaf node ω (see Fig. 1. (b)). Besides, we use $\omega|_{j,(\mathrm{RS})}$ or $\omega|_{j,(\mathrm{LS})}$ to denote the right or left sibling of $\omega|_j$ respectively if $\omega|_j$ has such a sibling. Namely, $\omega|_{j,(\mathrm{RS})} = [\omega_1\omega_2\cdots\omega_{j-1}1]$ or $\omega|_{j,(\mathrm{LS})} = [\omega_1\omega_2\cdots\omega_{j-1}0]$.

Generally speaking, our BE system consists of two parallel BTE systems: the right BTE system and the left BTE system. The right BTE system covers all the leaf nodes in the interval $[\omega, n]$ and the left BTE system covers all the leaf nodes in the interval $[1, \omega]$. User ω will be assigned with a unique right master key and a left master key. All the node secret keys or private keys for ω in the right BTE system are derived from the right master key while its node secret keys or private keys in the left BTE system are derived from the left master key. We use two different subindexes (L) or (R) in the notations of all these keys to distinguish the left or right BTE system they correspond to respectively.

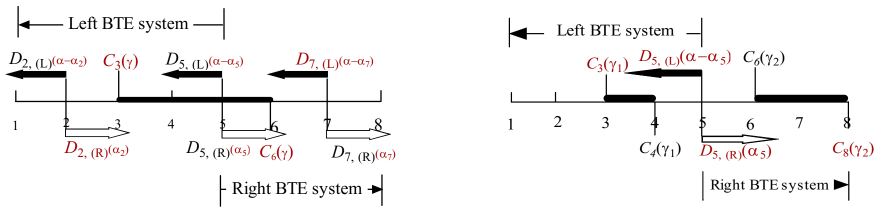
5 Primitive Idea: A Generic Transformation from BTE to Interval Encryption

5.1 Trivial Constructions

A trivial interval encryption scheme can be given directly from attribute based encryption [17] if one treats $\log n$ bits to represent a number from 1 to n as attributes and builds an access tree allowing specific intervals. However, even the most efficient trivial methodology would inevitably result in an interval encryption construction with a ciphertext size of $O(k \log n)$, where k is the number of intervals. As introduced in the introduction, our goal is to realize a broadcast encryption system in which the ciphertext size is determined by the number of intervals k. *If all the messages are only encrypted under the bounds of each interval like in the FSE scheme*, then this goal is reachable. However, how to make sure that only those receivers with an index within two bounds of each interval can open the message still represents a challenge.

5.2 A Generic Transformation from BTE to Interval Encryption

Yet there remains some difficulties to conquer. The first difficulty is how to differentiate the decryption ability of an index in and outside of an interval. Taking the interval [3, 6] shown in Fig. 2. a for instance, we could easily find the required difference if we project two oppositely direction arrows from each index in the axis, where the connotation of the arrows can be found in our exposition of the last paragraph in Sec. 3. The key observation to our transformation is that: *the two opposite direction arrows starting from index* 5 *can cross both bounds* 3 *and* 6 *respectively and therefore decrypt the corresponding partial ciphertext in two different manners* (We will show how to differentiate the partial decryption from two different directions, and how this will eventually lead to successful generation of the corresponding message encryption key in the sequel). *However, only one unique direction arrow from index* 2 *or* 7 *can cross the two bounds, i.e., only the right direction arrow from index* 2 *can cross* 3 *and* 6 *while only the left direction arrow from index* 7 *can cross* 3 *and* 6. *This implies that those outside of an interval can only decrypt the partial ciphertext in a unique manner.* The private key for right direction arrow is called right private key in the concrete construction while the one for left direction arrow is left private key.



a. prevention of two-user collusion: $\alpha_2\gamma+(\alpha-\alpha_7)\gamma\neq\alpha\gamma$

b. prevention of single user collusion: $\alpha_5\times\gamma_2+(\alpha-\alpha_5)\times\gamma_1\neq\alpha\times\gamma_2\neq\alpha\times\gamma_1$

Fig. 2. Collusion and its prevention

We require that the master key of the underlying BTE or HIBE scheme only contains *one group element*. The message encryption key for each interval corresponds to $\alpha \cdot \gamma$ in the exponent of a pairing, where α is the system master key and γ is a randomness chosen by the encryptor. For each user ω, we choose a random number α_ω and split the master key α into two parts: one is the right master key α_ω, which serves as the root master key for the right BTE system, from which the right private key $D_{\omega,(R)}$ of ω is derived; the other part is the left master key $\alpha - \alpha_\omega$, i.e., the root master key for the left BTE system, from which the left private key $D_{\omega,(L)}$ of ω is derived. It is observable that the two private keys for ω can be distributed similarly to a FSE scheme as shown in Fig. 1. b. Consequently, a partial decryption using the user's right private key contains $\alpha_\omega \cdot \gamma$ in the exponent of a pairing while a partial decryption using the left private key will have $(\alpha - \alpha_\omega) \cdot \gamma$ in its exponent. Then, the message encryption key containing $\alpha \cdot \gamma$ in its exponent will be recovered since $\alpha = \alpha_\omega + \alpha - \alpha_\omega$ holds.

In this way, we can actually prevent a possible collusion attack called two-user collusion. For example (shown in Fig. 2. a), a user $\omega = 7$ with a left private key $D_{7,(L)}$ (which could decrypt the partial ciphertext C_3) and a user $\omega = 2$ with right private key $D_{2,(R)}$ (which could decrypt the partial ciphertext C_6) might collude to open the message aiming for interval $[3, 6]$ (since they could also complete the partial decryption in two different manners) although neither of them is in this particular interval. In our system, the partial decryption from $D_{7,(L)}$ will contain $(\alpha - \alpha_7) \cdot \gamma$ while the partial decryption from $D_{2,(R)}$ contains $\alpha_2 \cdot \gamma$ in their exponents, and hence the collusion will fail since there's no way for them to incorporate $\alpha \cdot \gamma$ in the final step.

Besides, we require the encryptor to use a unique randomness γ_j while generating the ciphertext for each interval NI_j. This aims to prevent another attack called a single-user collusion. This attack only occurs in the scenario with multiple intervals (where $k \geq 1$). For instance (shown in Fig. 2. b), in an interval encryption system with two intervals $[3,4] \bigcup [6,8]$, the partial decryption on C_3 from the left private key $D_{5,(\mathrm{L})}$ contains $\alpha - \alpha_5$ and the partial decryption on C_8 from the right private key $D_{5,(\mathrm{R})}$ contains the other half randomness α_5, the message encryption key corresponding to $\alpha \cdot \gamma$ might be recovered if these two intervals use the same randomness. However, a unique randomness for each interval can guarantee that only a user within a certain interval can successfully open the message. For example (as shown in Fig. 2. b), a randomness γ_1 is used in the ciphertext for interval $[C_3, C_4]$ and γ_2 is used in the encryption for interval $[C_6, C_8]$. The message encryption keys of these two intervals correspond to $\alpha \cdot \gamma_1$ and $\alpha \cdot \gamma_2$ respectively. A single-user collusion fails since the randomized partial decryption $(\alpha - \alpha_5)\gamma_1$ and $\alpha_5\gamma_2$ won't incorporate a meaningful encryption key in the final step (see Fig. 2. b).

Although the proposed methodology is generic, it is not fully generic since it somehow relies on the property of bilinear mapping. Therefore, we only illustrate our methodology using concrete examples rather than providing a formal description of a generic interval encryption system in the following sections.

6 Basic Construction: A Concrete Instantiation Based on BBG HIBE

In the following section, we'll describe how the proposed methodology can be applied to the BBG HIBE(viewed as a binary tree encryption scheme here) construction [3] to propose an interval encryption system. Note that there is an additional algorithm **DeckeyDer** $(D_\omega = \{D_{\omega,(\mathrm{L})}, D_{\omega,(\mathrm{R})}\}, \zeta, \eta)$ compared with the original definition of interval encryption in Sec. 2.2. This algorithm is a preliminary step for the decryption algorithm, and we treat it as an independent algorithm for clarity. Besides, there's an additional slightly technical modification to the underlying BTE construction in the sense that we basically have two concrete instantiations of a hash function to guarantee that we could cover both the two bounds of each interval in the security proof.

Let $\mathbb{G}_1$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_1$. In addition, let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ denote the bilinear map. A security parameter, κ, will determine the size of the groups. Assume the system accommodates $n = 2^d$ users, where d is an integer.

Setup(n)**:** Select a random $\alpha \in \mathbb{Z}_p$ and set $g_1 = g^\alpha$. Choose random elements g_2, $g_{3,(\mathrm{L})}$, $g_{3,(\mathrm{R})}$, $h_{1,(\mathrm{L})}, \cdots, h_{d,(\mathrm{L})}, h_{1,(\mathrm{R})}, \cdots, h_{d,(\mathrm{R})}$ from $\mathbb{G}_1$.

The public key is $PK = (g, g_1, g_2, g_{3,(\mathrm{L})}, g_{3,(\mathrm{R})}, h_{1,(\mathrm{L})}, \cdots, h_{d,(\mathrm{L})}, h_{1,(\mathrm{R})}, \cdots, h_{d,(\mathrm{R})})$. For a binary string $v = [v_1 v_2 \cdots v_j]$ where $j \in [1,d]$, define two publicly computable functions: $F_{(\mathrm{L})}(v) = g_{3,(\mathrm{L})} \cdot \prod_{i=1}^{j} h_{i,(\mathrm{L})}^{v_i}$ and $F_{(\mathrm{R})}(v) = g_{3,(\mathrm{R})} \cdot \prod_{i=1}^{j} h_{i,(\mathrm{R})}^{v_i}$. The system master key is $SK_\varepsilon = g_2^\alpha$.

PvkGen(ω, SK_ε)**:** For receiver $\omega = [\omega_1\omega_2 \cdots \omega_d]$ which is associated with the ω-th leaf node (starting from left to right), the algorithm first chooses a random number α_ω. The right master key for ω is $SK_{\varepsilon,(\mathrm{R})}{=}g_2^{\alpha_\omega}$, and the left master key is $SK_{\varepsilon,(\mathrm{L})}{=}g_2^{\alpha-\alpha_\omega}$. The algorithm first generates two node secret keys $SK_{\omega,(\mathrm{R})}{=}[g_2^{\alpha_\omega}(F_{(\mathrm{R})}(\omega))^{r_\omega}, g^{r_\omega}]$ and $SK_{\omega,(\mathrm{L})}{=}[g_2^{\alpha-\alpha_\omega}(F_{(\mathrm{L})}(\omega))^{r_\omega}, g^{r_\omega}]$ for leaf node ω where r_ω is a random number from $\mathbb{Z}_p$.

For all the nodes $\omega|_j, j = 1, \cdots, d$ on the path from the root to the leaf node ω, if it has a right sibling $\omega|_{j,(\mathrm{RS})}=[\omega_1\omega_2 \cdots \omega_{j-1}1]$, the algorithm uses the right master key to generate the respective node secret key as $SK_{\omega|_{j,(\mathrm{RS})},(\mathrm{R})}=[g_2^{\alpha_\omega}\ (F_{(\mathrm{R})}(\omega|_{j,(\mathrm{RS})}))^{r_j},\ g^{r_j},\ h_{j+1,(\mathrm{R})}^{r_j}, \cdots,\ h_{d,(\mathrm{R})}^{r_j}]$ where r_j is also a random number; Otherwise the algorithm uses the left master key to generate node secret key for its left sibling $\omega|_{j,(\mathrm{LS})}=[\omega_1\omega_2\cdots\omega_{j-1}0]$ as $SK_{\omega|_{j,(\mathrm{LS})},(\mathrm{L})}=[g_2^{\alpha-\alpha_\omega}\ (F_{(\mathrm{L})}(\omega|_{j,(\mathrm{LS})}))^{r_j},\ g^{r_j},\ h_{j+1,(\mathrm{L})}^{r_j}, \cdots,\ h_{d,(\mathrm{L})}^{r_j}]$.

Output private key $D_\omega = \{D_{\omega,(\mathrm{R})}, D_{\omega,(\mathrm{L})}\}$, where $D_{\omega,(\mathrm{R})} = \{SK_{\omega,(\mathrm{R})}, SK_{\omega|_{j,(\mathrm{RS})},(\mathrm{R})}\}_{j\in[1,d]}$ and $D_{\omega,(\mathrm{L})} = \{SK_{\omega,(\mathrm{L})}, SK_{\omega|_{j,(\mathrm{LS})},(\mathrm{L})}\}_{j\in[1,d]}$.

Encrypt(S, PK)**:** The encryptor first chooses a k-wise natural interval set $S = \bigcup_{j=1}^{k} NI_j$, where $NI_j = [l_j, r_j]$. For each interval, pick γ_j uniformly from $\mathbb{Z}_p$ at random. Let the binary name of the corresponding leaf nodes for the two bounds be $r_j = [r_{j1} \cdots r_{jd}]$ and $l_j = [l_{j1} \cdots l_{jd}]$.

Output the respective ciphertext $C_{l_j}=\{g^{\gamma_j},\ (F_{(\mathrm{L})}(l_j))^{\gamma_j}\}$ and $C_{r_j}=\{g^{\gamma_j},\ (F_{(\mathrm{R})}(r_j))^{\gamma_j}\}$. Set the message encryption key for each interval NI_j as $K_j=\hat{e}(g_1,\ g_2)^{\gamma_j} \in \mathbb{G}_2$. The collection of these partial ciphertexts constitute the header Hdr$=\{C_{l_j},\ C_{r_j}\}_{j=1}^{k}$.

DeckeyDer $(D_\omega = \{D_{\omega,(\mathrm{L})}, D_{\omega,(\mathrm{R})}\}, \zeta, \eta)$**:** This algorithm derives the node secret key $SK_{\eta,(\mathrm{L})}$ for the lower bound η, and $SK_{\zeta,(\mathrm{R})}$ for the upper bound ζ.

1. Let a natural number $\eta \le \omega$ denote the η-th leaf node, and thus η is on the left of ω in the binary tree. Assume the binary representation of η is $\eta = \eta_1 \cdots \eta_d$. There must exist a node secret key $SK_{\eta|_j,(\mathrm{L})}, j \in [1, d]$ which belongs to $D_{\omega,(\mathrm{L})}$ (as shown in Fig. 1. b). Run the derivation algorithm of the underlying BTE scheme iteratively, which means the following steps need to be executed iteratively for $i = j$ to $i = d - 1$:

(a) Let $\eta|_i = \eta_1 \cdots \eta_i$. Parse $SK_{\eta|_i,(\mathrm{L})}$ as $\left(g_2^{\alpha-\alpha_\omega}(F_{(\mathrm{L})}(\eta|_i))^{r_i},\ g^{r_i},\ h_{i+1,(\mathrm{L})}^{r_i}, \cdots,\ h_{d,(\mathrm{L})}^{r_i}\right)=(a_0, a_1, b_{i+1}, \cdots, b_d)$.

(b) Choose random $t \in \mathbb{Z}_p$, and output $SK_{\eta|_{i+1},(\mathrm{L})} = (a_0 \cdot b_{i+1}^{\eta_{i+1}} \cdot (F_{(\mathrm{L})}(\eta|_{i+1}))^t, a_1 \cdot g^t, b_{i+2} \cdot h_{i+2,(\mathrm{L})}^t, \cdots, b_d \cdot h_d^t)$ and set $i = i + 1$.

Finally, it will output a node secret key $SK_{\eta,(\mathrm{L})}=[g_2^{\alpha-\alpha_\omega}(F_{(\mathrm{L})}(\eta))^{r'},\ g^{r'}]$ for the lower bound η.

2. Let a natural number $\zeta \ge \omega$ denote the ζth leaf node. Assume the binary representation of ζ is $\zeta = \zeta_1 \cdots \zeta_d$. Therefore there must exist a node secret key $SK_{\zeta|_j,(\mathrm{R})}$ which belongs to $D_{\omega,(\mathrm{R})}$. Run the derivation algorithm of the underlying BTE scheme iteratively, which means steps 1(a)-1(b) need to be executed iteratively.

Output a node secret key $SK_{\zeta,(\mathrm{R})}=[g_2^{\alpha_\omega}(F_{(\mathrm{R})}(\zeta))^{r''},\ g^{r''}]$ for the upper bound ζ.

Decrypt $(S, \omega, D_\omega, \mathrm{Hdr}, PK)$**:** If $\omega \in NI_j = [l_j, r_j], 1 \le j \le k$ which implies that $l_j \le \omega \le r_j$, then it runs **DeckeyDer** (D_ω, r_j, l_j) to generate decryption key $SK_{r_j,(\mathrm{R})}$ and $SK_{l_j,(\mathrm{L})}$. It obtains the corresponding secret key $SK_{r_j,(\mathrm{R})}=[g_2^{\alpha_\omega}(F_{(\mathrm{R})}(r_j))^{r''},\ g^{r''}]$ and the partial ciphertext for the upper bound $C_{r_j}=\{g^{\gamma_j},\ (F_{(\mathrm{R})}(r_j))^{\gamma_j}\}$. Compute $\frac{\hat{e}[g^{\gamma_j},\ g_2^{\alpha_\omega}(F_{(\mathrm{R})}(r_j))^{r''}]}{\hat{e}[g^{r''},\ (F_{(\mathrm{R})}(r_j))^{\gamma_j}]} = \hat{e}(g,\ g_2)^{\gamma_j\alpha_\omega}$. It also obtains the corresponding secret key $SK_{l_j,(\mathrm{L})}=[g_2^{\alpha-\alpha_\omega}(F_{(\mathrm{L})}(l_j))^{r'},\ g^{r'}]$ and the partial ciphertext for the lower bound $C_{l_j}=\{g^{\gamma_j},\ (F_{(\mathrm{L})}(l_j))^{\gamma_j}\}$. Compute $\frac{\hat{e}[g^{\gamma_j},\ g_2^{\alpha-\alpha_\omega}(F_{(\mathrm{L})}(l_j))^{r'}]}{\hat{e}[g^{r'},\ (F_{(\mathrm{L})}(l_j))^{\gamma_j}]} = \hat{e}(g,\ g_2)^{\gamma_j(\alpha-\alpha_\omega)}$. Finally, it manages to compute $\hat{e}(g,\ g_2)^{\gamma_j\alpha_\omega} \cdot \hat{e}(g,\ g_2)^{\gamma_j(\alpha-\alpha_\omega)} = \hat{e}(g^\alpha,\ g_2)^{\gamma_j} = \hat{e}(g_1,\ g_2)^{\gamma_j}$

6.1 Discussion on Efficiency and Security

In this construction, the public key size is $O(\log n)$, and the private key only contains $O(\log n)$ BTE node secret keys. Note that the private key in the DF transformation [12] of the NNL method or the HS construction contains $O(\log^2 n)$ or $O(\log^{1+\epsilon} n)$ node secret keys respectively. It is important to point out that a widely used tool, updateable public storage in the FSE scheme [4], can be also adopted in our proposed interval encryption system to limit the private storage cost to $O(\log n)$. The above efficiency parameters could be further improved if the random oracle is adopted, i.e., the public key size can be reduced to $O(1)$ in this case.

The decryption cost is dominated by the derivation of the two node secret keys. The derivation cost can be reduced to $O(\log n)$ by doing the following computation: in order to deduce the node secret key $SK_{\eta,(\mathrm{L})}=[g_2^{\alpha-\alpha_\omega}\cdot (F_{(\mathrm{L})}(\eta))^{r'}, g^{r'}]=(a_0', a_1')$ from $SK_{\eta|_i,(\mathrm{L})}=\left(g_2^{\alpha-\alpha_\omega}(F_{(\mathrm{L})}(\eta|_i))^{r_i}, g^{r_i}, h_{i+1,(\mathrm{L})}^{r_i}, \cdots, h_{d,(\mathrm{L})}^{r_i}\right)=(a_0, a_1, b_{i+1}, \cdots, b_d)$, we could compute $a_0' = a_0 \cdot \prod_{k=i+1}^{d} b_k^{\eta_k} \cdot (F_{(\mathrm{L})}(\eta))^t, a_1' = a_1 \cdot g^t$ where we force $r' = r_i + t$. We could deduce the node secret key $SK_{\zeta,(\mathrm{R})}$ from $SK_{\zeta|_i,(\mathrm{R})}$ in a similar way. The overall decryption time is then reduced to $O(\log n)$ since the rest of the decryption procedure only requires a constant number of group operations.

Why is $O(k)$ better: A system with a transmission overhead proportional to k is more efficient than the traditional systems, especially the system where communication load is linearly dependent on r such as the revocation system proposed in [24,27]. To demonstrate that, we compare the performance of both systems in presence of different values for r as well as k. We assume that the total node number n is set to $2^{17} = 131072$ and let r increase from 1 to n. For a specific r, we randomly generate 1000 revoked sets, which correspond to 1000 different interval number k, and thus obtain an average interval number $\overline{k}$ as well as the average transmission overhead of the proposed scheme, which has been shown in Fig. 3. From Fig. 3, it is observed that, when the revocation set is small, the performance of the proposed scheme is very close to the tradition systems, however the difference will be scaled up along with the increase of r. If the revoked set number exceeds 50% of the total number, the communication load of the proposed scheme will decrease with increase of r. It is also observed that the proposed scheme can achieve the best performance in case that r is very large, which further demonstrates that the proposed scheme is suitable for cases when a small receiver set is employed. Compared with the BGW generalized construction [5] with a $\sqrt{n}$ size transmission overload which only serves as a better choice than the trivial solution and the traditional systems when $r > \sqrt{n}$, we have the benefit that our system keeps the advantage of the traditional constructions when r is a small number, namely $r \ll n$.

From the above results, we conclude that the proposed construction fits into more cases than the traditional systems dependent on r, and therefore constitutes a more favorable choice in practice.

The selective security of the proposed construction can be proven secure under the $d+1$-BDHE assumption, and it's stated as follow. We leave the concrete proof in the full version.

Theorem 1. *If the Decisional $(d+1)$-BDHE assumption holds in $\mathbb{G}_1, \mathbb{G}_2$, then the proposed interval encryption scheme is selective chosen plaintext secure.*

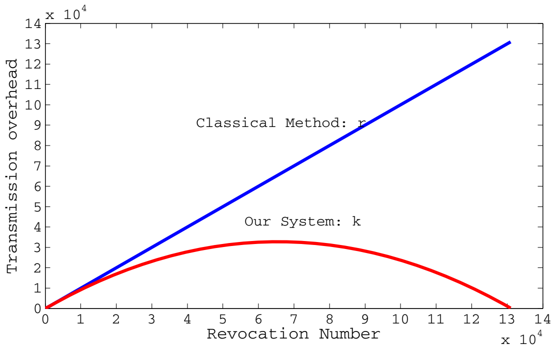


Fig. 3. Comparison between k and r

7 Inclusive Extended Interval Encryption

An inclusive extended interval encryption scheme deals with the scenario where the message is encrypted under a collection of intervals $S = \bigcup_{j=1}^{k}[l_j, r_j]$, and the private key D_ω of a user ω corresponds to an interval $[l_\omega, r_\omega]$. The decryption is successful if and only if there exists at least one interval $[l_j, r_j]$, $j \in [1, k]$ such that $[l_\omega, r_\omega] \subseteq [l_j, r_j]$. To generate a private key corresponding to an interval $[l_\omega, r_\omega]$ (see Fig. 4), we simply generate a left private key $D_{l_\omega,(\mathrm{L})}$ corresponding to the lower bound l_ω using the left master key $\alpha - \alpha_\omega$ as in the basic construction. Similarly we generate a right private key $D_{r_\omega,(\mathrm{R})}$ for the upper bound r_ω using the right master key α_ω. The rest of the above algorithms have no significant differences from those in the basic construction. Furthermore, it is easy to observe that a man holding a private key for an interval $[l_\omega, r_\omega]$ can delegate a private key for another interval $[l_{\omega'}, r_{\omega'}]$ using the **DeckeyDer** algorithm as long as $[l_\omega, r_\omega] \subseteq [l_{\omega'}, r_{\omega'}]$. This is a property somewhat close to a recently proposed concept called inclusive identity based encryption (IBE) [6]. We consider this extended construction of important theoretical interest since there exist very few inclusive constructions [13] since the proposal of inclusive IBE.

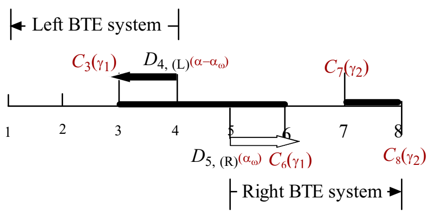


Fig. 4. Extended Interval Encryption: the generation of a private key for an interval [4, 5]

8 Adaptively Secure Interval Encryption

This construction is based on Lewko and Waters' HIBE construction [20]. The basic idea is to apply our proposed transformation method to Lewko and Waters' HIBE scheme, and the concrete construction will be shown in the full version.

9 Extensions and Future Work

9.1 Range Attribute Based Encryption

In a key policy attribute based encryption (ABE) [17], a private key might be associated with an access policy such as "Old man **AND** tall". A man holding this private key can open a message encrypted under an attribute set {"Old man", "tall"} since this attribute set satisfies the above access policy. In practice, the attributes in an attribute set might have certain range and the attributes in an access policy might be assigned with certain concrete evaluations. In the above example, the access policy might be denoted as a formula "Age: 60 **AND** Height: 180 (cm)". A man holding a private key associated with the above policy should be able to open a message encrypted under an attribute set {"Age: 50 to 100", "Height: 175 to 250 (cm)"}. The reason for the successful decryption is that both evaluations of the two attributes fall into the range required in the attribute set and hence the access policy is satisfied. However, A man holding a private key associated with an access policy "Age: 49 **AND** Height: 180 (cm)" cannot decrypt this message since the evaluation "Age: 49" is not within the corresponding range "Age: 50 to 100" in the attribute set. A range ABE scheme is realizable from a traditional ABE scheme. However, the ciphertext will blow up with a $\log n$ factor as shown in our trivial example of constructing interval encryption from ABE.

The proposed interval encryption scheme can be easily modified to a range ABE scheme with a constant ciphertext size. The primitive idea and concrete construction can be found in the full version.

9.2 Interval Encryption under Simpler Assumption

The proposed method also applies to those BTE or HIBE schemes, in which cases their master keys only contain one single group element such as [2,9]. We can construct interval encryption schemes based on the decisional bilinear Diffie-Hellman assumption. The concrete steps are similar to that of Sec.6 and hence trivial. The weakness of these constructions is that the ciphertext size will blow up with a $\log n$ factor compared with the basic construction while the private key size remains $O(\log n)$.

9.3 Encryption under a Graph

Consider the following application: a message might be encrypted under a digital map of a certain territory on earth (which a close two-dimensional graph can represents) and only those who hold a private key for a location in the territory can open the message. This notion might actually intrigue several interesting applications. For example, a launch order of a certain weapon might be encrypted under a map of a specific region and only those who have a private key corresponding to a location within this region can launch this weapon. Apparently, we could map all the points in a two-dimensional digital map to the points in an one-dimensional axis. We could simply calculate $i = (y - 1)c + x$ where (x, y) is a point in a two-dimensional map with width c and height d. If we set $n = c * d$, then all the points can be mapped into an index $i \in [1, n]$. In other words, all the points within the territory of this digital map can be mapped into a collection of intervals. Therefore, the proposed interval encryption provides a solution for the above scenario. The count of intervals depends on the perimeter of this graph.

9.4 Future Work and Open Problems

The reason why the proposed construction can improve the transmission overload relies on the fact that we utilize the difference between the points in and outside a certain interval. How to use the difference between a point in and outside of a graph to reduce the transmission overload, especially to minimize the constant factor k during the multi-dimensional scenario is left as an important open problem. It is possible to borrow some idea from computational geometry to solve this problem. To propose a BTE or HIBE construction with improved efficiency or under a weaker assumption which fits into our framework is very interesting since this directly implies the improvement of interval encryption.

References

1. Attrapadung, N., Imai, H.: Graph-decomposition-based frameworks for subset-cover broadcast encryption and efficient instantiations. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 100–120. Springer, Heidelberg (2005)
2. 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)
3. 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)
4. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext (2005), http://eprint.iacr.org/2005/015
5. 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)
6. 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)
7. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
8. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)

9. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
10. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. J. Cryptology 20(3), 265–294 (2007)
11. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007)
12. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 61–80. Springer, Heidelberg (2003)
13. Gentry, C., Halevi, S.: Hierarchical identity based encryption with polynomially many levels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 437–456. Springer, Heidelberg (2009)
14. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
15. Gentry, C., Waters, B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 171–188. Springer, Heidelberg (2009)
16. Goodrich, M.T., Sun, J.Z., Tamassia, R.: Efficient tree-based revocation in groups of low-state devices. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 511–527. Springer, Heidelberg (2004)
17. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
18. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 47–60. Springer, Heidelberg (2002)
19. Jho, N.-S., Hwang, J.Y., Cheon, J.H., Kim, M.-H., Lee, D.H., Yoo, E.S.: One-way chain based broadcast encryption schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 559–574. Springer, Heidelberg (2005)
20. Lewko, A.B., Waters, B.: Fully secure hibe with short ciphertexts, http://eprint.iacr.org/2009/
21. Liu, Y.-R., Tzeng, W.-G.: Public key broadcast encryption with low number of keys and constant decryption time. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 380–396. Springer, Heidelberg (2008)
22. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
23. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM Conference on Computer and Communications Security, pp. 195–203 (2007)
24. Sahai, A., Waters, B.: Revocation systems with very small private keys, http://eprint.iacr.org/2008/309
25. Wang, P., Ning, P., Reeves, D.S.: Storage-efficient stateless group key revocation. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 25–38. Springer, Heidelberg (2004)
26. 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)
27. Yoo, E.S., Jho, N.-S., Cheon, J.H., Kim, M.-H.: Efficient broadcast encryption using multiple interpolation methods. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 87–103. Springer, Heidelberg (2005)

Shrinking the Keys of Discrete-Log-Type Lossy Trapdoor Functions

Xavier Boyen and Brent Waters*

Institut Montefiore, Université de Liège, Belgium
University of Texas at Austin, USA

Abstract. To this day, realizations in the standard-model of (lossy) trapdoor functions from discrete-log-type assumptions require large public key sizes, e.g., about $\Theta(\lambda^2)$ group elements for a reduction from the decisional Diffie-Hellman assumption (where λ is a security parameter). We propose two realizations of lossy trapdoor functions that achieve public key size of only $\Theta(\lambda)$ group elements in bilinear groups, with a reduction from the decisional Bilinear Diffie-Hellman assumption.

Our first construction achieves this result at the expense of a long common reference string of $\Theta(\lambda^2)$ elements, albeit reusable in multiple LTDF instantiations. Our second scheme also achieves public keys of size $\Theta(\lambda)$, entirely in the standard model and in particular without any reference string, at the cost of a slightly more involved construction.

The main technical novelty, developed for the second scheme, is a compact encoding technique for generating compressed representations of certain sequences of group elements for the public parameters.

1 Introduction

The notion of Lossy Trapdoor Function (LTDF) is a new fundamental public-key primitive that was recently introduced by Peikert and Waters [20], henceforth PW. This notion has generated interest for two main reasons:

1. LTDFs can be constructed from widely differing hardness assumptions. These include the two recent constructions of PW from the Discrete-Log-based DDH and a worst-case Lattice assumption [20]. In addition, the Damgård-Jurik [9] variant of the decade-old Paillier cryptosystem [19], which, as was independently pointed out in [23,5], happens to immediately give an LTDF from the Factoring-based assumption of composite residuosity.
2. LTDFs can in turn serve as black-box building blocks within more complex primitives, such as regular injective trapdoor functions, provably collision-resistant hashing, and public-key encryption with chosen-ciphertext security [20], as well as some new strong notions of security for deterministic public-key encryption [5]. We may also expect to see more such applications in the future.

* Supported by NSF CNS-0716199, Air Force Office of Scientific Research (AFOSR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO) and the U.S. DHS under Grant Number 2006-CS-001-000001.

Since LTDFs are general and admit several applications, an interesting pursuit is to seek how to realize them in the most efficient manner, from robust and hardness assumptions. Peikert and Waters (PW), who first proposed the notion, provide two LTDF constructions from appealing hardness assumptions [20]. All of their constructions provided built upon a public key that reflected a "matrix structure". One drawback of this approach is that it results in a rather large public key of $\Theta(\lambda^2)$ group elements and thus (for the DDH-based construction) grows cubically with the security parameter λ. As a concrete example, at the $\lambda = 128$ bit security level, if we assume an optimal elliptic-curve implementation with the smallest possible 256-bit element representation, this implies a public-key storage and transmission requirement in excess of $(3 \times 128)^2 \times 256$ bits, or about 36 Mb, for every single instance of the LTDF.

In subsequent work Boldyreva, Fehr, and O'Neill [5] and Rosen and Segev [23] provided constructions of Lossy Trapdoors from the composite residuosity assumption. These constructions provided greater efficiency. Their security depends upon the composite residuosity assumption, while our goal is to look for new and efficient lossy trapdoors that do not depend upon the difficulty of factoring.

Our goal is to work toward achieving efficient trapdoor functions based on discrete log assumptions. One compelling reason to find solutions in the discrete log setting is that there are potentially many concrete instances for any one constructions. For example, there are several different realizations of bilinear groups [13]. If one group them turned out to be insecure, a construction might still be viable under a different group. In contrast, the assumption of factoring is absolute.

Our Approach. In this work, we aim toward making LTDFs from discrete log assumptions realizable in practice. Our guiding principle is to try to *compress* the DDH-based LTDF construction from [20] in order to shrink the public key size from $\Theta(\lambda^2)$ to $\Theta(\lambda)$ group elements.

More precisely, we propose two new LTDF constructions that extend the PW DDH-based LTDF onto pairing-friendly curves. Since with a (symmetric) pairing the DDH assumption is generally false, we use the related decisional Bilinear Diffie-Hellman (DBDH) assumption. DBDH is weaker than DDH, which is why it may hold even in bilinear groups where DDH does not, but in counterpart our constructions do require a pairing.

On pairing-friendly curves, we will show how to exploit the bilinear map in various ways in order to remove much of the redundancy from the LTDF public key, and instead allow the user to reconstruct it efficiently as needed. Not only does this save a factor of λ in the public-key size, it also makes the LTDF computation more space- and time-efficient.

We begin by observing that in the original construction there is informational theoretically much redundancy in their public key. When constructing an injective key their setup algorithm chooses $r_1, \ldots, r_n, a_1, \ldots, a_n \in \mathbb{Z}_p$ for a trapdoor function of input length n. The (i, j)-th entry for $i \neq j$ is the element $g^{r_i \cdot a_j}$ and along the diagonal the key consists of $g^{r_i \cdot a_i} \cdot g$.

In order to take advantage of this redundancy our first idea is to use a bilinear group $\mathbb{G}$ with generator g. A natural approach is to consider publishing as part of the public key $g^{a_1}, \ldots, g^{a_n}, g^{r_1}, \ldots, g^{r_n}$ consisting of only $O(n)$ group elements. Then when evaluating the trapdoor, one can dynamically generate the PW matrix by taking the (i, j)-th element as $e(g^{r_i}, g^{a_j})$. While this first technique does indeed compress the matrix, it actually gives the attacker too much information. In particular, it gives the attacker the diagonal of a *lossy* key. This allows the attacker to trivially distinguish between a lossy and an injective key; undercutting the main security guarantee. Our challenge is create a public key that allows a user to generate all the matrix elements *except* along the diagonal, but while simultaneously keeping the key short ($\Theta(n)$ group elements).

In our first construction we amortize the cost of a creating the matrix by leveraging a (fixed, reusable) common reference string (CRS) to achieve the compression. The reference string consists of $\Theta(n^2)$ group elements, and is therefore as long as one public key in the PW scheme; however, each additional public key consists of just $O(n)$ group elements. This system works since the CRS provides information when taken along with the public key to generate everything except along the diagonal. The primary downsides to this approach are the reliance on a trusted third party to generate the reference string and the large size of the reference string itself.

In our second construction we develop a new method in order to achieve true key compression without CRS. To do this we find a somewhat surprising connection to the identity-based revocation scheme of Lewko, Sahai, and Waters [17]. In their work, they describe a "two equation" technique for identity-based revocation. They show how to encrypt to an identity ID^* such that a private key for any identity $\text{ID} \neq \text{ID}^*$ is able to decrypt.[1] Intuitively, we will apply a (high-level) version of this idea, not for revocating IBE users but for compressing an LTDF public key.

Conceptually we will map each row i of a ciphertext to a ciphertext that is associated with a "virtual identity" i and each column j with a "virtual key" for identity j. Using this approach one is able to decompress the encoding and obtain an LTDF public key matrix for every element (i, j), except on the diagonal where $i = j$. Along the diagonal, the system generates two dependent equations and the public key will provide these components separately. Taken all together, this gives us a Lossy TDF system with $\Theta(n)$ size public keys in the standard model.

The security of both schemes follows from the DBDH assumption, respectively in the common reference string model in the first case and in the standard model in the second case.

We note that, while our keys and ciphertext only require $\Theta(\lambda)$ group elements instead of $\Theta(\lambda^2)$, half of the new elements will live in the bilinear "target" group $\mathbb{G}_t$ instead of entirely in the "source" group $\mathbb{G}$. For known pairing and known attacks, the optimal choices of $\mathbb{G}$ and $\mathbb{G}_t$ are such that the representation size of

[1] The work of Lewko et al. [17] actually shows how to efficiently revoke several users at once, but for our purposes discussing a single revocation is sufficient.

$\mathbb{G}_t$-elements grows faster than that of $\mathbb{G}$-elements. This makes our technique less useful in the asymptote that one could have expected (based on current knowledge). Nevertheless, our framework is quite advantageous for practical values of the security parameter.

1.1 Related Work

The concept of trapdoor functions was first proposed by Diffie and Hellman [10]. In an (injective) trapdoor function $f()$ a party with a trapdoor can invert the function; however, inversion should be hard for any attacker. Trapdoor functions have several interesting applications in cryptography ranging from two party computation [26], Non-Interactive Zero Knowledge Proofs [4], and Chosen-Ciphertext Secure Encryption [18,11] among others.

The first trapdoor realization was given by Rivest, Shamir, and Adleman [22] with security based on what has become known as the RSA assumption. Other standard model construction include the factoring based one of Rabin [21] and that of Paillier [19]. We note that all these standard model constructions rely on the difficulty of factoring.

Using the random oracle heuristic it is possible to transform any secure public key encryption system into an injective trapdoor [3,2]. In addition, Bellare, Halevi, Sahai, and Vadhan gave generic standard model transformations from one way functions to highly *non-injective* trapdoor functions; however, the applications of these forms of trapdoors is rather limited. For example, they cannot be used to realize public key encryption. For this reason, we will often implicitly assume injectiveness when discussing useful trapdoor functions.

Until recently, the only known standard model realizations of trapdoor functions (LTDFs) relied on the difficulty of factoring. Recently, Peikert and Waters [20] introduced the concept of *Lossy Trapdoor Functions*. A Lossy Trapdoor system has the property that a publicly evaluable function f can be created to either be an injective function or highly non-injective; moreover, an adversary should not be able to distinguish what type of function f is given its description. They showed that Lossy TDFs implied standard trapdoor functions and gave several other applications of LTDFs including chosen-ciphertext secure encryption. In addition, they showed different realizations from both the hardness of the decisional Diffie-Hellman problem and certain lattice-based problems. One drawback of their construction is that uses a matrix of public key components that results in large public keys of $O(\lambda^2)$ group elements.

The PW lattice-based LTDF construction fares better asymptotically than the discrete-log one, since it requires $O(\lambda^2)$ elements of Z_q where q is relatively small[2]. However, the constants associated with the lattice-based construction might make it less efficient for security parameters in the foreseeable future, and for this paper we focus on the discrete-log setting (specifically, in a pairing-enabled context).

[2] Typically, q is set to be much less than 2^λ. See Gama and Nguyen [14] for a discussion of current lattice parameters.

One potential benefit of our work is that our public key compression techniques, developed here in a pairing context, might have future applications to the lattice setting. We note that lattice-based crypto analogues of pairing-based techniques have been used to construct Identity-Based Encryption in a lattice setting: e.g., Gentry et al. [15] recently gave a lattice-based trapdoor function system with interesting applications such as hash-and-sign signatures and identity-based encryption: at first in the random-oracle model (see [15]), and more recently in the standard model (see [1,7,8]).

More recently, Boldyreva, Fehr, and O'Neill [5] and Rosen and Segev [23] independently showed that the Damgård Jurik [9] extension of the Paillier [19] trapdoor was actually an efficient lossy trapdoor function and therefore inherits its applications. Freeman et al. [12] gave a number of LTDF constructions based on the quadratic and composite residuosity assumptions, or on d-Linear assumptions (see [6,24]). Hemenway and Ostrovsky [16] showed that LTDFs can also be constructed from smooth homomorphic hash proof systems.

2 Preliminaries

Before describing our constructions, it is useful to review and understand the original scheme, and also to see where it can be improved. (The paper [20] has two constructions, one from the DDH problem, the other from worst-case Lattice problems. We focus on the first. We refer to the paper for the full details of their construction.)

2.1 Simplified Definition of Lossy TDFs

First, we give a somewhat simplified definition of Lossy TDF, based on [20]. Let λ be a security prameter and $n(\lambda)$ be the length of the input on which the function is evaluated.

Definition 1. *A lossy trapdoor function (LTDF) is a collection of polynomial-time algorithms* $(S_{inj}, S_{loss}, F_{ltdf}, F_{ltdf}^{-1})$ *such that*

– S_{inj} *randomly generates an injective function along with an associated trapdoor,*
– S_{loss} *generates a lossy function (and no trapdoor),*
– F_{ltdf} *evaluates any function generated by either* S_{inj} *or* S_{loss} *on any input vector* $\mathbf{x} \in \{0,1\}^n$*,*
– F_{ltdf}^{-1} *recovers the original input* $\mathbf{x}$ *from the output* $\mathbf{y}$ *of an injective function using its trapdoor.*

Additionally, there is a security requirement that the injective and lossy functions respectively generated by S_{inj} *and* S_{loss} *be computationally indistinguishable.*

This simplified definition will cover all the cases of interest in this paper. We refer to [20] for a more precise and more complex definition. A few related notions (such as the amount of "lossiness" induced by the lossy mode of an LTDF) will be recalled as we need them.

3 Compact LTDFs from DBDH in the CRS Model

Our first LTDF construction with a compressed output is set in the common reference string (CRS) model, and uses bilinear maps.

The main interest of this construction is that it is very simple, yet produces a Lossy TDF with shorter public keys than any previous ones not based on factoring. It also features a security reduction from the decisional bilinear Diffie-Hellman assumption, which is one of the most-studied and weakest among all the very many bilinear-group assumptions that have been made to date.

Although the reference string in this scheme is about as long as the public key in the PW DDH scheme of the previous section, we stress that the CRS is reusable across users whereas public keys are clearly not.

3.1 Intuition

At a high level, this LTDF is a bit similar to the PW DDH-based PW. A generalized ElGamal scheme is used to encrypt a matrix message $\mathbf{M}$ that is either the identity matrix $\mathbf{I}$ or the zero matrix $\mathbf{0}$, with the same consequences on injectivity versus lossiness as before. The novelty lies in the distribution of the public key.

Recall that in the PW DDH-based LTDF, the public key comprises the actual ciphertext $\mathbf{C}$ of $\mathbf{M}$, and whose representation takes up to $n \times (n+1)$ group elements. Of course, there is a lot of redundancy in $\mathbf{C}$, due to the way it is constructed, but all of this redundancy had to remain computationally hidden in order for the injective and lossy modes to remain indistinguishable.

Our general goal here will be carefully to reveal a portion of this redundancy, so that it can be reconstructed at the time of use (explicitly or implicitly) without having to be transmitted, but without compromising the computational indistinguishability between an injective LTDF and a statistically lossy one.

Our approach in this first scheme starts with a decomposition of the ciphertext $\mathbf{C} = (\mathbf{C}_1 \| \mathbf{C}_2)$ linearly into two "additive" components: $\mathbf{C} = \mathbf{C}^{bulk} \oplus \mathbf{C}^{diag}$, where $\oplus$ is the element-wise matrix addition using $\mathbb{G}$'s own group operation (which we chose to write using a multiplicative notation):

1. The first component, $\mathbf{C}^{bulk}$, will correspond to $\mathbf{C}_1$ and all the elements of $\mathbf{C}_2$ off the diagonal: this is the "bulk" of the matrix $\mathbf{C}$, which stays the same in both injective and lossy modes.
2. The second component, $\mathbf{C}^{diag}$, will comprise just the elements on the diagonal of $\mathbf{C}_2$: there are only n of them, and the only ones whose construction changes between the two modes.

Since $\mathbf{C}^{diag}$ is already relatively compact, most of the gain will come from compressing $\mathbf{C}^{bulk}$. Observe in the PW scheme that the off-diagonal elements of $\mathbf{C}_2$ are of the form $g^{r_i z_j}$, where r_i is ephemeral and z_j belongs to the private key (if any); i.e., we have a product of two secret values in the exponent of a fixed group generator. Instead of publishing all the $g^{r_i z_j}$ elements *in extenso*, this *suggests* the publication of "precursor" elements g^{r_i} and g^{z_j} and the use of a pairing to reconstruct $e(g,g)^{r_i z_j} = e(g^{r_i}, g^{z_j})$ on the receiving end. From there, one could then implement the rest of the PW scheme in the group generated by $e(g,g)$ instead of $\mathbb{G}$.

Unfortunately, this idea does not quite work yet, because it also exposes the diagonal values $e(g^{r_i}, g^{z_j})$ for $i = j$, and from this information the injective and lossy modes become easy to distinguish. We need to forbid the reconstruction of $e(g^{r_i}, g^{z_j})$ when $i = j$, and allow it only when $i \neq j$. This is where the CRS comes into play: the CRS can be constructed in such a way that it contains values of the form $e(g^{r_i}, g^{z_j})$, or even $g^{r_i z_j}$, but only for $i \neq j$ and not for $i = j$.

Alas, there is still one problem: the private key elements z_j clearly cannot be part of a CRS that is meant to be reusable across multiple users. The solution is to have the CRS contain a reusable matrix of non-diagonal elements $g^{r_i a_j}$, and have each LTDF public key contain an independent vector of g^{b_j}. A bilinear pairing can be used to reconstruct $e(g^{r_i a_j}, g^{b_j}) = e(g,g)^{r_i a_j b_j}$ as needed, for $i \neq j$. The products $a_j b_j$ play the role of the z_j in the PW scheme (though only b_j is available as trapdoor information). The r_i are analogous to the secret ephemeral randomizers in the PW scheme, except that they have been immortalized in the CRS.

The main difficulties will be to show that inversion can still be performed only with the partial trapdoor b_j (in injective mode), and to prove that the r_i can be safely reused for all LTDF instances (even with different public keys) without putting their security in jeopardy.

3.2 Construction

Consider a pair of finite, abelian, bilinear groups $(\mathbb{G}, \times)$ and $(\hat{\mathbb{G}}, \times)$ of prime order $p = |\mathbb{G}| = |\hat{\mathbb{G}}|$, respectively generated $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$. Let $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_t$ be a non-degenerate, efficiently computable, bilinear pairing into a third abelian group $\mathbb{G}_t$ also of order $p = |\mathbb{G}_t|$ and thus generated by $e(g, \hat{g}) \in \mathbb{G}_t$. We use the multiplicative notation for the group operation in all three groups. There may or may not exist efficiently computable group homomorphisms between $\mathbb{G}$ and $\hat{\mathbb{G}}$; we make no requirement of this nature either way. We assume that the Decision Bilinear Diffie-Hellman (DBDH) problem is hard in $\mathbb{G} \times \hat{\mathbb{G}}$, meaning that no probabilistic polynomial-time algorithm $\mathbf{A}$ can distinguish $(g, g^a, g^b, \hat{g}, \hat{g}^a, \hat{g}^c, e(g,\hat{g})^{abc})$ from $(g, g^a, g^b, \hat{g}, \hat{g}^a, \hat{g}^c, e(g,\hat{g})^d)$ with probability non-negligibly greater than $\frac{1}{2}$, for randomly chosen $a, b, c, d \in \mathbb{Z}_p$.

As before, in all generality we must consider not a fixed pair of groups $\mathbb{G}$ and $\hat{\mathbb{G}}$, but an infinite family of such groups sampled by a PPT instance generator $\mathcal{G}$, where $\mathcal{G}$ on input 1^λ outputs a description of $(p, \mathbb{G}, g, \hat{\mathbb{G}}, \hat{g}, e)$ such that $\lfloor \log_2 p \rceil = \Theta(\lambda)$. The DBDH assumption, rigorously speaking, pertains to the family of groups induced by $\mathcal{G}$ for sufficiently large λ.

For simplicity of notation, in the description of the scheme and its proof below, we drop all "hats", thereby blurring the distinction between the two bilinear groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ (though $\mathbb{G}_t$ must remain distinct). For example, in this case the Decision-BDH problem becomes to distinguish $(g, g^a, g^b, g^c, e(g,g)^{abc})$ from $(g, g^a, g^b, g^c, e(g,g)^d)$.

Our compact LTDF from DBDH in the CRS model is constructed as follows:

CRS Setup: The universal setup algorithm, on input 1^λ, selects a bilinear group $(p, \mathbb{G}, g, e) \leftarrow \mathcal{G}(1^\lambda)$ and fixes $n > 3\log_2 p$. It then secretly chooses random $r_i \in \mathbb{Z}_p$ for $i \in [n]$ and random $a_j \in \mathbb{Z}_p$ for $j \in [n]$, and publishes a CRS that consists of the following:

- the description of $\mathbb{G} = \langle g \rangle$, and the pairing e;
- g^{r_i} for all $i \in [n]$;
- g^{a_j} for all $j \in [n]$;
- $g^{r_i a_j}$ for all $(i, j) \in [n]^2$ such that $i \neq j$;

Everyone can check that the CRS has been computed correctly (though of course not that the exponents r_i and a_i have been actually picked at random and forgotten).

Sampling algorithm, Injective mode: The injective function generator, denoted $S_{\text{inj}}(p, \mathbb{G}, g, e)$, randomly draws $b_j \in \mathbb{Z}_p$ for $j \in [n]$. It looks up the CRS to obtain g^{r_i} for $i \in [n]$ and g^{a_j} for $j \in [n]$.

- The LTDF function index, or "public key", is published as $2n$ elements of $\mathbb{G}$ and $\mathbb{G}_t$ in total, arranged in a single-row matrix $\mathbf{B}$ and a diagonal matrix $\mathbf{D}$ (where $1 = e(g, g)^0$),

$$\mathbf{B} = \begin{pmatrix} g^{b_1} & \cdots & g^{b_n} \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} e(g^{r_1}, g^{a_1})^{b_1} \cdot e(g, g) & 1 & \cdots & 1 \\ 1 & e(g^{r_2}, g^{a_2})^{b_2} \cdot e(g, g) & \cdots & 1 \\ \vdots & & \ddots & 1 \\ 1 & 1 & \cdots & e(g^{r_n}, g^{a_n})^{b_n} \cdot e(g, g) \end{pmatrix}$$

- The LTDF trapdoor, or "private key", consists of the n-vector $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{Z}_p^n$.

Sampling algorithm, Lossy mode: The lossy function generator, denoted $S_{\text{loss}}(p, \mathbb{G}, g, e)$, looks up the CRS and proceeds in the same fashion as S_{inj}, except that the public and private keys are created differently.

- The LTDF function index, or "public key", consists of two matrices $\mathbf{B}$ and $\mathbf{D}$, where,

$$\mathbf{B} = \begin{pmatrix} g^{b_1} & \cdots & g^{b_n} \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} e(g^{r_1}, g^{a_1})^{b_1} & 1 & \cdots & 1 \\ 1 & e(g^{r_2}, g^{a_2})^{b_2} & \cdots & 1 \\ \vdots & & \ddots & 1 \\ 1 & 1 & \cdots & e(g^{r_n}, g^{a_n})^{b_n} \end{pmatrix}$$

- There is no LTDF trapdoor, or "private key" (since in this case one is statistically unable to perform an inversion, whether one is given the b_j or not).

Evaluation algorithm: The evaluation algorithm, denoted F_{ltdf}, takes as input $((\mathbf{B}, \mathbf{D}), \mathbf{x})$, where $(\mathbf{B}, \mathbf{D})$ is a function index, and $\mathbf{x} \in \{0,1\}^n$ is an n-bit binary input string. The evaluation algorithm F_{ltdf} naturally also has access to the CRS. To facilitate the exposition, we present two different but functionally equivalent implementations of F_{ltdf}.

- *"Pedestrian" evaluation (requires $n^2 - n$ pairings):*
  To compute the desired output, the simplest way is first to reconstruct the analogous to the matrix $\mathbf{C}$ in the PW scheme, and then proceed with the function evaluation in the analogous way. One big difference with PW is that, here, the matrix $\mathbf{C}$ does not have all of its elements in the same group $\mathbb{G}$. Rather, the zero-th column, $\mathbf{C}_1$, has elements in $\mathbb{G}$, whereas the remaining n column, $\mathbf{C}_2$, have elements in $\mathbb{G}_t$.

  Let thus $\mathbf{C} = \mathbf{C}_1 \| \mathbf{C}_2$ with $\mathbf{C}_1 \in \mathbb{G}^{n\times 1}$ and $\mathbf{C}_2 \in \mathbb{G}_t^{n\times n}$ computed from $\mathbf{B}$, $\mathbf{D}$, and the CRS,

$$\mathbf{C}_1 = \begin{pmatrix} g^{r_1} \\ \vdots \\ g^{r_n} \end{pmatrix} \qquad \mathbf{C}_2 = \begin{pmatrix} D_1 & e(g^{r_1 a_2}, B_2) & \cdots & e(g^{r_1 a_n}, B_n) \\ \vdots & & \ddots & \vdots \\ e(g^{r_2 a_1}, B_1) & D_2 & \cdots & e(g^{r_2 a_n}, B_n) \\ \vdots & & \ddots & \vdots \\ e(g^{r_n a_1}, B_1) & e(g^{r_n a_2}, B_2) & \cdots & D_n \end{pmatrix}$$

  The output is the row-vector $\mathbf{y} = \mathbf{x} \cdot \mathbf{C}$, with 1 element in $\mathbb{G}$ and n elements in $\mathbb{G}_t$, given by,

$$\begin{aligned} \mathbf{y} = \mathbf{x} \cdot \mathbf{C} = \mathbf{x} \cdot (\mathbf{C}_1 \| \mathbf{C}_2) &= \begin{pmatrix} x_1 \dots x_n \end{pmatrix} \cdot \begin{pmatrix} c_{1,0} & c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \ddots & \vdots \\ c_{n,0} & c_{n,1} & \cdots & c_{n,n} \end{pmatrix} \\ &= \left(\prod_{i=1}^n c_{i,0}^{x_i} \ , \ \prod_{i=1}^n c_{i,1}^{x_i} \ , \ \cdots \prod_{i=1}^n c_{i,n}^{x_i} \right) \qquad \in \mathbb{G} \times \mathbb{G}_t^n \end{aligned}$$

- *"Shortcut" evaluation (requires n pairings):*
  A careful study of the above procedure suggests a faster but equivalent way to evaluate the function on the given inputs. Instead of expanding each element of $\mathbf{C}$ using a pairing, only to have them multiplied up together down the line, we keep the multiplication in mind from the start which allows us to do but a single pairing for each output element.

  The output must be a row-vector $\mathbf{y}$ of size $n + 1$, where $y_0 \in \mathbb{G}$ and each of the elements $y_j \in \mathbb{G}_t$ for $j \in [n]$ are directly computable from the respective expressions,

$$y_0 = \prod_{i=1}^{n} (g^{r_i})^{x_i} \quad \in \mathbb{G} \qquad , \qquad y_j = e\left(\prod_{i \in [n]\setminus\{j\}} (g^{r_i a_j})^{x_i} \ , \ B_j \right) \cdot D_j^{x_j} \quad \in \mathbb{G}_t$$

  It is easy to see that both methods give the same result. The pedestrian methods clearly shows the analogy to PW, while the shortcut method requires much less memory and is more efficient.

Inversion algorithm: The inversion algorithm F_{ltdf}^{-1} is input $(\mathbf{b}, \mathbf{y})$, where $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{Z}_p^n$ is the trapdoor from the injective sampling algorithm S_{inj}, and $\mathbf{y} = (y_0, y_1, \ldots, y_n) \in \mathbb{G}_t^{n+1}$ is the output of the evaluation algorithm F_{ltdf}. The algorithm F_{ltdf}^{-1} has access to the CRS.

The bits of the preimage $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ of $\mathbf{y}$ are output as follows,

$$x_j = \begin{cases} 0 & \text{if } y_j = e(y_0, g^{a_j})^{z_j} \cdot e(g,g)^0 \\ 1 & \text{if } y_j = e(y_0, g^{a_j})^{z_j} \cdot e(g,g)^1 \end{cases}$$

Naturally, $e(y_0, g^{a_j})^{z_j}$ is only computed once for each $j \in [n]$, so the entire inversion process requires n pairing evaluations — or $n+1$, strictly speaking, though $e(g,g)$ never changes.

3.3 Security

There are three properties that must be satisfied in order for our scheme to be an LTDF: (1) injectivity in injective mode; (2) enough lossiness in lossy mode; (3) computational indistinguishability of the two modes.

Theorem 1. *The preceding algorithms define a collection of* $(n, n - \log_2 p)$*-Lossy TDFs under the Decision Bilinear Diffie-Hellman assumption for* $\mathbb{G}$*, in the Common Reference String model.*

Proof. We need to prove each of the three listed properties: (1) injectivity in injective mode; (2) enough lossiness in lossy mode; (3) computational indistinguishability of the two modes.

Injectivity in injective mode follows from that, since the matrix $\mathbf{M}$ and thus $\mathbf{C}_2$ is invertible, the information from $\mathbf{x}$ will be preserved by the evaluation function, which by construction amounts to "multiplying" the input vector $\mathbf{x}$ by the matrix $\mathbf{C}_2$ (and also $\mathbf{C}_1$ which is important to compute the function inverse but not to show injectivity).

Lossiness in lossy mode follows from that, under the parameters generated by S_{loss}, the evaluation function $F_{\mathrm{ltdf}}(\mathbf{C}, \cdot)$ has at most p possible output values as the input vector $\mathbf{x}$ varies, for any given choice of randomness, and so the information about $\mathbf{x}$ contained in the output $\mathbf{y}$ is at most $\log_2 p$ bits. Since the domain of $\mathbf{x}$ is $\{0,1\}^n$ of size 2^n for $n > 3\log_2 p$, it follows that the lossiness (i.e., the information loss about $\mathbf{x}$ for a uniformly distributed $\mathbf{x}$) caused by $F_{\mathrm{ltdf}}(\mathbf{C}, \cdot)$ is equal to $k = \log_2(2^n) - \log_2 p = n - \log_2 p > n - n/3 = 2n/3$.

Indistinguishability is the only property that requires a complexity assumption, and whose proof makes use of the common reference string. The result is stated in Lemma 1.

Lemma 1. *Under the Decision Bilinear Diffie-Hellman assumption in* $\mathbb{G}$*, the distributions of public keys* $\mathbf{C}$ *generated by* S_{inj} *and by* S_{loss} *cannot be distinguished with non-negligible advantage by a probabilistic polynomial-time algorithm, in the CRS model.*

Proof (Proof.). The proof is based on a hybrid-game argument; that is, we gradually switch the LTDF from an injective-mode setup S_{inj} to a lossy-mode setup S_{loss}, one element of the diagonal at a time. That is, for $k \in [n]$, one k at a time, we change D_k from the injective definition (where $m_k = 1$) into the lossy definition (where $m_k = 0$). For each k, we then show that if the adversary can successfully distinguish the transition from H_k to H_{k+1}, then we can turn it into a BDH distinguisher. (And it must distinguish at least one such transition, if it is to distinguish injective from lossy.)

We begin by defining intermediate game H'_k where in game H'_k the first $k-1$ elements are in lossy mode, D_k is a random group element, and D_i for $i > k$ are chosen according to the injective setup.

Under the BDH assumption we can show that H_k is indistinguishable from H'_k. Our reduction algorithm takes in a d-BDH challenge tuple $g, g^a, g^b, g^c \in \mathbb{G}$ and $T \in \mathbb{G}_t$. For $i \neq k$ it chooses random a_i, b_i, r_i itself. It uses these to populate the CRS and the public key $\mathbf{B}$ at all positions except the k-th position. It then sets $g^{a_k} = g^a$, $g^{r_k} = g^c$, and $g^{b_k} = g^b$. Using these knowledge of b_i for $i \neq k$ the reduction can create $D_i = (e(g^{a_i}, g^{r_i}))^{b_i}$ for $i < k$ and for $D_i = (e(g^{a_i}, g^{r_i}))^{b_i} e(g, g)$ for $i > k$. It finally creates $D_k = Te(g, g)$. If T is a bilinear tuple we are in game H_k; otherwise we are in game H'_k.

A symmetrical argument can show that for all k it is difficult to distinguish between H'_k and H_{k+1}. Putting the sequence together completes our proof.

4 Compact LTDFs from DBDH in the Standard Model

We now present a second LTDF construction with public keys about as compact as our first construction, and likewise also based on the DBDH assumption in bilinear groups, but without CRS or any setup requirement.

4.1 Intuition

The general structure is similar to that of our first scheme, which is to say that it combines the PW LTDF strategy to achieve lossiness with some algebraic manipulations to remove as much redundancy as possible from the public key.

There is a paradox that we mentioned in the intuition of our first scheme: that, in order to achieve any meaningful compression of the elements of $\mathbf{C}$ without "outside help", we would need to expose the redundancy that was hidden in the matrix $\mathbf{C}$. But we could not do that, because it would allow an adversary to reconstruct not only its off-diagonal elements , but also the "forbidden" elements on the diagonal (those that determine whether the function is injective or lossy). Our solution was to use a (thankfully reusable) common reference string to reveal (the better part of) the off-diagonal elements without saying anything much at all about the diagonal ones.

Here, we will do much better, and resolve the paradox: we construct a special compact encoding that, without any outside help, will let anyone easily reconstruct the off-diagonal elements of $\mathbf{C}$, without exposing the diagonal ones (which are sent separately).

The idea is to construct a linear system of equations, with one equation per row i and one per column j, in such a way that each element at the intersection (i, j) can be obtained by solving the pair of the i-th row and j-column equations for two variables. The trick is that the equations are constructed in such a way that they become linearly dependent for $i = j$, rendering them useless on the diagonal — and only on the diagonal. As stated in the introduction, this can be viewed as a novel twist on Lewko's et al. [17] two equation-based revocation system, where we conceptually match up rows and columns of the Lossy TDF with ciphertexts and private keys of an broadcast revocation system.

Of course, we cannot give the equations in the clear, we use a technique that is quite common in Discrete-Log-hard groups, of encoding all the coefficients we need to hide as powers of some group generator. That is, instead of revealing $\alpha \in \mathbb{Z}_p$, we reveal $g^\alpha \in \mathbb{G}$. Because $\mathbb{Z}_p$ and $\mathbb{G}$ are homomorphic through exponentiation, we can still perform any additive operation that we like on the "α"s hidden in the exponents.

Then, since our stratagem with the pairs of equations require some multiplication at some point when trying to solve for the unknowns, we need to make use of a bilinear map in order to do this one multiplication "in the exponent".

We now describe our system. The crux of the construction lies in the denominator $\frac{1}{j-i}$ that we make appear along the computation path, and that will break things down when one tries it for $i = j$.

4.2 Construction

The notation regarding bilinear groups is the same as in Section 3. Likewise, to avoid clutter, we drop all "hats" in the notation, thereby implicitly assuming a symmetric pairing $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_t$ where $\mathbb{G} = \hat{\mathbb{G}}$. The system continues to work in the asymmetric setting, for any partition of the bilinear-group elements into $\mathbb{G}$ or $\hat{\mathbb{G}}$, as long as it is self-consistent.

Our compact LTDF from DBDH in the Standard Model is constructed as follows:

Sampling algorithm, Injective mode: The injective function generator S_{inj}, on input a security parameter 1^λ, selects a bilinear group $(p, \mathbb{G}, g, e) \leftarrow \mathcal{G}(1^\lambda)$, and fixes $n > 3 \log_2 p$. It also chooses two independent random generators $u \in \mathbb{G}$ and $h \in \mathbb{G}$ in addition to $g \in \mathbb{G}$. It then chooses random $r_i \in \mathbb{Z}_p$ for each $i \in [n]$, and random $z_j \in \mathbb{Z}_p$ for each $j \in [n]$.

- The LTDF function index, or "public key", is listed as $4n$ elements of $\mathbb{G}$ and n of $\mathbb{G}_t$:

$$\begin{array}{rcrr} \text{for each "row" index} i \in [n] & : & R_i = g^{r_i} \quad , & S_i = (h^i \cdot u)^{r_i} \\ \text{for each "column" index} j \in [n] & : & V_j = g^{z_j} \quad , & W_j = (h^j \cdot u)^{z_j} \\ \text{for each "diagonal" element} k \in [n] & : & & D_k = e(g, u)^{r_k z_k} \cdot e(g, g) \end{array}$$

- The LTDF trapdoor, or "private key", consists of the n-vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{Z}_p^n$.

Sampling algorithm, Lossy mode: The lossy function generator S_{loss}, on input a security parameter 1^λ, proceeds similarly as S_{inj} above, except for the key computation:

– The LTDF function index, or "public key", is listed as $4n$ elements of $\mathbb{G}$ and n of $\mathbb{G}_t$:

$$\begin{array}{rcccc} \text{for each "row" index} i \in [n] & : & R_i = g^{r_i} & , & S_i = (h^i \cdot u)^{r_i} \\ \text{for each "column" index} j \in [n] & : & V_j = g^{z_j} & , & W_j = (h^j \cdot u)^{z_j} \\ \text{for each "diagonal" element} k \in [n] & : & & D_k = \ e(g,h)^{r_k z_k} & \end{array}$$

– There is no LTDF trapdoor (since the evaluation output will be lossy even with the knowledge of all the secret exponents).

Evaluation algorithm: The evaluation algorithm F_{ltdf} takes as input an LTDF public key and an n-bit binary input string $\mathbf{x} \in \{0,1\}^n$. Again, there are at least two functionally equivalent ways to perform the computation; we start with the inefficient one for expository purposes.

• *"Pedestrian" evaluation (requires a total of n^2 pairings and double-pairings):* To compute the desired output, the evaluation algorithm starts by reconstructing a virtual matrix $\mathbf{C}$ very similar to that of the CRS scheme of Section 3 (in its "pedestrian" implementation), and then proceed with the actual function evaluation. With the present notation, here the virtual matrix $\mathbf{C} = \mathbf{C}_1 \| \mathbf{C}_2$ is written,

$$\mathbf{C}_1 = \begin{pmatrix} e(g,h)^{r_1} \\ \vdots \\ e(g,h)^{r_n} \end{pmatrix}$$

$$\mathbf{C}_2 = \begin{pmatrix} e(g,h)^{r_1 z_1} \cdot e(g,g)^{m_1} & e(g,h)^{r_1 z_2} & \cdots & e(g,h)^{r_1 z_n} \\ e(g,h)^{r_2 z_1} & e(g,h)^{r_2 z_2} \cdot e(g,g)^{m_2} & \cdots & e(g,h)^{r_2 z_n} \\ \vdots & \vdots & \ddots & \vdots \\ e(g,h)^{r_n z_1} & e(g,h)^{r_n z_2} & \cdots & e(g,h)^{r_n z_n} \cdot e(g,g)^{m_n} \end{pmatrix}$$

As before, the "message" $(m_1, \ldots, m_n)$ is a vector of all 0 or all 1, depending whether the LTDF is lossy or injective (which fact must remain unknown to F_{ltdf}). We now show how the evaluation algorithm can compute $\mathbf{C}$ given the information at its disposal.

The zero-th column of $\mathbf{C}$ consists of elements of $\mathbb{G}_t$ that are computed from the public key. For each index $i \in [n]$, the i-th element of $\mathbf{C}_1$ is computed as the pairing,

$$\begin{aligned} c_{i,0} \ &:= e(R_i, h) \\ &= e(g,h)^{r_i} \ \in \mathbb{G}_t \quad \text{which is indeed the required value for } c_{i,0} \end{aligned}$$

The diagonal elements of $\mathbf{C}$ are elements of $\mathbb{G}_t$ which are explicitly given in the public key. For $k \in [n]$, those are the given values D_k. Hence, the algorithm simply assigns, for $i \in [n]$,

$$\begin{aligned} c_{i,i} \ &:= D_i \\ &= e(g,h)^{r_i z_i} \cdot e(g,g)^{m_i} \ \in \mathbb{G}_t \quad \text{as required, with } m_i \in \{0,1\} \text{ unknown} \end{aligned}$$

The off-diagonal of $\mathbf{C}$, i.e., the values $c_{i,j}$ with indices $i, j \in [n]$ such that $i \neq j > 0$, must be computed explicitly. For each such pair (i, j), the algorithm F_{ltdf} computes a double-pairing, which is a product, or more precisely here, a ratio, of two pairings. (Such "double pairings" can be computed almost as fast as a single pairing.) It computes,

$$c'_{i,j} \ := \frac{e(R_i, W_j)}{e(V_j, S_i)} = \frac{e(g, h^j)^{r_i z_j} \cdot e(g,u)^{r_i z_j}}{e(g, h^i)^{r_i z_j} \cdot e(g,u)^{r_i z_j}} = \frac{e(g, h^j)^{r_i z_j}}{e(g, h^i)^{r_i z_j}} = \Big(e(g,h)^{r_i z_j}\Big)^{j-i}$$

and for each such value takes its $(j-i)$-th root,

$$c_{i,j} \ := \left(c'_{i,j}\right)^{\frac{1}{j-i}} = \Big(e(g,h)^{r_i z_j}\Big)^{\frac{j-i}{j-i}} = e(g,h)^{r_i z_j} \quad \text{as required for } c_{i,j}$$

Thus, for $i \neq j$ this gives the desired outcome, $e(g,h)^{r_i z_j}$, as the preceding derivations show. Crucially, this computation and the root-taking step in particular will succeed if and only if $i \neq j$. Hence, a malicious F_{ltdf} cannot use this method to compute $e(g,h)^{r_k z_k}$ on the diagonal, and, by comparison with D_k, infer whether the LTDF is lossy or injective.

The actual LTDF output is the row-vector $\mathbf{y} = \mathbf{x} \cdot \mathbf{C}$ with $n+1$ elements in $\mathbb{G}_t$, given by,

$$\mathbf{y} = \mathbf{x} \cdot \mathbf{C} = \mathbf{x} \cdot (\mathbf{C}_1 \| \mathbf{C}_2) = \left(\ \prod_{i=1}^n c_{i,0}^{x_i} \ , \prod_{i=1}^n c_{i,1}^{x_i} \ , \ \dots \prod_{i=1}^n c_{i,n}^{x_i} \ \right) \ \in \mathbb{G}_t^{n+1}$$

- *"Shortcut" evaluation (requires n double-pairings plus 1 pairing):*

  As in Section 3, it is possible to delay the pairing computation and do the group multiplications first. This reduces the number of pairings (or more precisely, two-pairing ratios) from n to 1, for each element of the output vector.

  Thus, to compute the output $\mathbf{y}$, a row-vector of $n+1$ elements, it suffices for the algorithm F_{ltdf} to compute y_j for $j = 0$ and $j \in [n]$ respectively as follows,

$$\begin{aligned} y_0 \ &:= e\Big(\prod_{i=1}^n R_i^{x_i} \ , \ h\Big) \\ &= \prod_{i=1}^n e(g,h)^{r_i x_i} \ \in \mathbb{G} \quad \text{which is the required value for the output } y_0 \end{aligned}$$

$$\begin{aligned}
y_j \; &:= \left(e\Big(\textstyle\prod_{i\neq j} R_i^{\frac{x_i}{j-i}} \,,\, W_j \Big) / e\Big(V_j \,,\, \textstyle\prod_{i\neq j} S_i^{\frac{x_i}{j-i}} \Big) \right) \cdot D_j^{x_j} \\
&= \frac{e\Big(\prod_{i\neq j} g^{\frac{r_i x_i}{j-i}} \;,\; (h^j \cdot u)^{z_j} \Big)}{e\Big(g^{z_j} \;,\; \prod_{i\neq j} (h^i \cdot u)^{\frac{r_i x_i}{j-i}} \Big)} \cdot D_j^{x_j} \;=\; \frac{\prod_{i\neq j} e(g,h)^{j \frac{r_i x_i}{j-i} z_j}}{\prod_{i\neq j} e(g,h)^{i \frac{r_i x_i}{j-i} z_j}} \cdot D_j^{x_j} \\
&= e(g,h)^{\sum_{i\neq j} \frac{j}{j-i} r_i x_i z_j} \cdot e(g,h)^{\sum_{i\neq j} \frac{-i}{j-i} r_i x_i z_j} \cdot D_j^{x_j} \\
&= e(g,h)^{\sum_{i\neq j} \frac{j-i}{j-i} r_i x_i z_j} \cdot D_j^{x_j} \;=\; \prod_{i\neq j} e(g,h)^{r_i x_i z_j} \cdot D_j^{x_j} \\
&= \prod_{i=1}^{n} e(g,h)^{r_i x_i z_j} \cdot e(g,g)^{m_j x_j} \;\in \mathbb{G}_t \quad \text{as required for } y_j
\end{aligned}$$

It is easy to see that this "Shortcut" evaluation gives the same result as the "Pedestrian" one, while requiring much less memory and being much faster.

Inversion algorithm: The inversion algorithm F_{ltdf}^{-1} is input $(\mathbf{z}, \mathbf{y})$, where $\mathbf{z} = (b_1, \ldots, b_n) \in \mathbb{Z}_p^n$ is the trapdoor output by the injective-mode S_{inj}, and $\mathbf{y} = (y_0, y_1, \ldots, y_n) \in \mathbb{G}_t^{n+1}$ is the output of the evaluation algorithm F_{ltdf}. The bits of the preimage $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ of $\mathbf{y}$ are output as follows,

$$x_j = \begin{cases} 0 & \text{if } y_j = y_0^{z_j} \cdot e(g,g)^0 \\ 1 & \text{if } y_j = y_0^{z_j} \cdot e(g,g)^1 \end{cases}$$

4.3 Security

Theorem 2. *The preceding algorithms define a collection of* $(n, n - \log_2 p)$*-Lossy TDFs under the Decision BDH assumption for* $\mathbb{G}$*.*

Proof. The theorem follows from the following Lemmas 2, 3, and 4.

Lemma 2. *For all LTDF parameters* $(\mathbf{C}, \mathbf{z})$ *produced by the injective-mode setup function* S_{inj}*, we have that* $\forall \mathbf{x} \in \{0,1\}^n$, $F_{ltdf}^{-1}(\mathbf{z}, F_{ltdf}(\mathbf{C}, \mathbf{x})) = \mathbf{x}$*, i.e., the LTDF output is invertible.*

Proof. By inspection of the algorithm specifications. Notice that in injective mode (i.e., for parameters sampled by S_{inj}), the exponents m_i on the diagonal are equal to 1, and thus preserve the information in the bit x_i, allowing the inversion algorithm to proceed.

Lemma 3. *For all LTDF parameters* $\mathbf{C}$ *produced by the lossy-mode setup function* S_{loss}*, the "lossiness" (or information loss for a uniform input vector* $\mathbf{x}$*) of the function* $F_{ltdf}(\mathbf{C}, \cdot)$ *is at least* $k = n - \log_2 p > \frac{2}{3} n$ *bits.*

Proof. By construction of the lossy-mode parameters generated by S_{loss}, the evaluation function $F_{\text{ltdf}}(\mathbf{C}, \cdot)$ has at most p possible output values as the input vector $\mathbf{x}$ varies, for any choice of randomness. Thus the information leakage about $\mathbf{x}$ is at most $\log_2 p$ bits. Since the domain of $\mathbf{x}$ is $\{0,1\}^n$ of size 2^n for $n > 3 \log_2 p$, it follows that the lossiness of $F_{\text{ltdf}}(\mathbf{C}, \cdot)$ is $k = \log_2(2^n) - \log_2 p = n - \log_2 p > n - n/3 = 2n/3$.

Lemma 4. *Under the DBDH assumption in $\mathbb{G}$, the distributions of public keys* $\mathbf{C}$ *generated by S_{inj} and by S_{loss} cannot be distinguished with non-negligible advantage by a probabilistic polynomial-time algorithm.*

Proof. The proof is based on a hybrid-game argument; that is, we gradually switch the LTDF from an injective-mode setup S_{inj} to a lossy-mode setup S_{loss}, one element of the diagonal at a time. That is, for $k \in [n]$, one k at a time, we change D_k from the injective definition (where $m_k = 1$) into the lossy definition (where $m_k = 0$). For each k, we then show that if the adversary can successfully distinguish the transition from H_k to H_{k+1}, then we can turn it into a BDH distinguisher. (And it must distinguish at least one such transition, if it is to distinguish injective from lossy.)

Here are the details of the hybrid argument.

We define a series of experiments $H_1, ..., H_{n+1}$, where, in the experiment H_k, the LTDF setup function sets $D_j = e(g,h)^{r_j z_j}$ for all $j < k$, and sets $D_j = e(g,h)^{r_j z_j} e(g,g)$ for all $j \geq k$. Recall that all the other components of the public key (i.e., other than the D_j) are defined identically in the injective mode and the lossy mode.

Observe that H_1 is an injective key with the correct distribution, and that H_{n+1} is a lossy key also with the correct distribution. It follows that if for no k can an adversary distinguish between H_k and H_{k+1}, then no adversary can distinguish the injective mode from the lossy mode.

Suppose then that for some k an adversary distinguishes H_k and H_{k+1}. Then we can use that adversary to solve a Decisional BDH challenge, thereby contradicting our assumption. Let thus $g, g^a, g^b, g^c \in \mathbb{G}$ and $T \in \mathbb{G}_t$ be our DBDH challenge, where T is either $e(g,g)^{abc}$ or a random element in $\mathbb{G}_t$.

For any fixed $k \in [n]$, the reduction for the transition from H_k to H_{k+1} works as follows.

For all $i \neq k$ and $j \neq k$, the simulator simply chooses $r_i \in \mathbb{Z}_p$ and $z_j \in \mathbb{Z}_p$ at random, and computes the corresponding public-key elements R_i and V_j as in the actual scheme.

For $i = k$ and $j = k$, the simulator instead chooses some random $y \in \mathbb{Z}_p$, and lets $h = (g^c)$ and $u = h^{-k} g^y$. It then sets $R_k = (g^a)$ and $V_k = (g^b)$ (thereby implictly setting $a = r_k$ and $b = z_k$, even though it does not know such values).

The simulator can now compute the public-key components S_i and W_j for all $i \neq k$ and $j \neq k$, since it knows the exponents r_i and z_j can thus compute $S_k = (g^a)^y$ and $W_k = (g^b)^y$. This works only because the contribution of $h = (g^c)$ vanishes from within S_k and W_k for this value of k, and one can check that this gives the correct values for S_k and W_k.

The simulator can also compute the public-key components D_i for all $i \neq k$. For consistency and continuity within the entire hybrid sequence of games, for $i < k$ the components D_i are set to be lossy (i.e., computed as in S_{loss}), while for $i > k$ the D_i are set to be injective (i.e., computed as in S_{inj}).

To set the one remaining public-key component, D_k, the simulator flips a coin $\beta \in \{0,1\}$ and defines $D_k = T \cdot e(g,g)^{\beta}$. The public key is given to the adversary, who must tell whether it is lossy or injective.

If the BDH instance was genuine, then $T = e(g, g)$ and the adversary will be in the situation of having to distinguish H_k (if $\beta = 1$) from H_{k+1} (if $\beta = 0$); it will thus be able to exploit its advantage, if any. However, if T is random then the adversary cannot have any advantage in this situation since the bit β is completely hidden from its view.

It follows that any advantage ϵ that the adversary has at distinguishing H_k from H_{k+1}, gives us a BDH distinguisher with advantage $\epsilon/2$.

To conclude the hybrid argument, considering the entire sequence of hybrid transitions from H_k to H_{k+1} for $k \in [n]$, we deduce that any adversary that can distinguish an injective key from a lossy key with advantage ϵ, must be able to distinguish H_k from H_{k+1} for at least one value of k with advantage ϵ/n, which in turn will give us a distinguisher for solving DBDH with advantage $\epsilon/(2n)$.

This concludes the proof of indistinguishability between the injective and lossy modes.

5 Conclusion

We have proposed two new Lossy Trapdoor Function schemes of the "discrete-log" type, that are significantly more efficient than earlier comparable constructions. We gave two schemes: with and withour common reference string. Both of them make use of pairings, and have efficient security reductions from the classic DBDH assumption.

The main advantage of "discrete-log-type" LTDF constructions, compared to more efficient ones of the "factoring" type (based on the Paillier trapdoor), is that the hardness of problems related to the discrete log generally depend on the choice of group, unlike those related to factoring that are, so to speak, absolute. In other words, even though Factoring is liable to be universally easy independently in all choices of context, there is a hope that Discrete Log and related problems can remain hard for certain choices of groups (e.g., the counterfactual generic groups [25]), provided that we can find and construct them.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110. Springer, Heidelberg (2010)
2. Bellare, M., Halevi, S., Sahai, A., Vadhan, S.P.: Many-to-one trapdoor functions and their relation to public-key cryptosystems. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 283–298. Springer, Heidelberg (1998)
3. 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)
4. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC (1988)
5. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)

6. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
7. Boyen, X.: Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056. Springer, Heidelberg (2010)
8. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110. Springer, Heidelberg (2010)
9. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
10. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
11. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. 30(2), 391–437 (2000); Preliminary version in STOC 1991
12. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. Cryptology ePrint Archive, Report 2009/590 (2009), http://eprint.iacr.org/
13. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
14. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)
16. Hemenway, B., Ostrovsky, R.: Lossy trapdoor functions from smooth homomorphic hash proof systems (manuscript, 2010), http://www.math.ucla.edu/~bretth/papers/uhp_ltdf.pdf
17. Lewko, A., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: Security and Privacy (2010)
18. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC, pp. 427–437 (1990)
19. 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)
20. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC, pp. 187–196 (2008)
21. Rabin, M.O.: Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology (1979)
22. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (1978)
23. Rosen, A., Segev, G.: Efficient lossy trapdoor functions based on the composite residuosity assumption. Cryptology ePrint Archive, Report 2008/134 (2008), http://eprint.iacr.org/
24. Shacham, H.: A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007), http://eprint.iacr.org/
25. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
26. Yao, A.C.-C.: Theory and applications of trapdoor functions (extended abstract). In: FOCS, pp. 80–91 (1982)

Trapdoor Sanitizable Signatures Made Easy

Dae Hyun Yum, Jae Woo Seo, and Pil Joong Lee

Electronic and Electrical Engineering, POSTECH, Republic of Korea
{dhyum,jwseo,pjl}@postech.ac.kr

Abstract. A sanitizable signature scheme allows a signer to partially delegate signing rights on a message to another party, called a sanitizer. After the message is signed, the sanitizer can modify pre-determined parts of the message and generate a new signature on the sanitized message without interacting with the signer. At ACNS 2008, Canard et al. introduced trapdoor sanitizable signatures based on identity-based chameleon hashes, where the power of sanitization for a given signed message can be delegated to possibly several entities, by giving a trapdoor issued by the signer at any time. We present a generic construction of trapdoor sanitizable signatures from ordinary signature schemes. The construction is intuitively simple and answers the basic theoretic question about the minimal computational complexity assumption under which a trapdoor sanitizable signature exists; one-way functions imply trapdoor sanitizable signatures.

1 Introduction

Message origin authentication can be achieved by digital signatures, since any alteration on signed messages nullifies the validity of digital signatures. However, some applications (e.g., releasing medical documents or classified government documents) require that some parts of a signed document be sanitized without invaliding the signature. This is often called the digital document sanitizing problem and many signature schemes for this problem have been proposed with various names: content extraction signature [1], redactable signature [2], and sanitizable signature [3]. Informally, we will use the term of sanitizable signature to refer to any signature scheme for the digital document sanitizing problem.

A sanitizable signature scheme allows a signer to delegate signing rights on a message to a sanitizer (or a censor) in a limited and controlled way. When generating a signature on a message, the signer selects a specific sanitizer who can later sanitize pre-determined parts of the message and update the signature without interacting with the signer. As pointed out by Ateniese et al. [3], there are many situations where re-signing by the original signer is undesirable, including: (1) the signer's key has expired, (2) the original signature was securely time-stamped, (3) the signer may not be reachable or available, and (4) each new signature would cost too much, either in terms of real expense or in terms of computation. In the literature, there are two different definitions of sanitizable signatures according to the degree of sanitizing capability. The first type is a sanitizable signature with pre-determined deletion that allows the sanitizer only to delete pre-determined parts of a signed message (e.g., [1,2,4,5,6]). The second type is a sanitizable signature with pre-determined modification that allows the sanitizer to modify pre-determined parts of the message and generate a new signature on the modified message (e.g., [3,7,8]). The capability of modification is valuable for many applications; especially for authenticated multicast, database outsourcing, and secure routing.

A simple sanitizable signature scheme can be constructed by applying a cryptographic hash function $\mathsf{hash}(\cdot)$ to the concatenation of each submessage with a random number [1]. To generate a signature on a message $m = m_1 \| m_2$ consisting of two submessages m_1 and m_2, the signer computes $\sigma = \mathcal{S}(h_1 \| h_2)$ for $h_i = \mathsf{hash}(m_i \| r_i)$ for $i = 1, 2$, where $\mathcal{S}(\cdot)$ is a signing algorithm of an ordinary signature scheme and r_i is a random number. The sanitizable signature on $m = m_1 \| m_2$ is (σ, r_1, r_2). If a sanitizer wants to delete m_1, he removes m_1 from m and replaces the signature with (σ, h_1, r_2). While verifies can check the validity of the sanitized version of the signed message, they cannot extract any information on the submessage m_1 from h_1 without knowing r_1. In this scheme, either (m_i, r_i) or h_i should be disclosed for verifiers to check the validity of (m, σ) and hence, an undeleted submessage can be sanitized by anyone.

A more sophisticated sanitizable signature scheme was introduced by Ateniese et al. [3], substituting a chameleon hash function $\mathsf{chash}(\cdot)$ for the hash function $\mathsf{hash}(\cdot)$. A chameleon hash [9,10] computed over a submessage m_i with randomness r_i, and under a public key pk is denoted by $\mathsf{chash}_{pk}(m_i, r_i)$. A chameleon hash function (or trapdoor commitment) has the same properties of any cryptographic hash function and, in particular, it provides collision resistance. However, the owner of the secret key sk corresponding to the public key pk can find collisions, i.e., m'_i and r'_i such that $h_i = \mathsf{chash}_{pk}(m_i, r_i) = \mathsf{chash}_{pk}(m'_i, r'_i)$. For given (m_i, r_i, m'_i), the unique randomness r'_i such that $\mathsf{chash}_{pk}(m_i, r_i) = \mathsf{chash}_{pk}(m'_i, r'_i)$ can be computed with the knowledge of sk. Chameleon hash functions are always probabilistic algorithms because of the randomness r_i in the input. For a given m_i, there are many hash values $h_i = \mathsf{chash}_{pk}(m_i, r_i)$ by using different randomness r_i. To verify the correctness of a computed chameleon hash value h_i, it is necessary to provide both the submessage m_i and the randomness r_i. If each sanitizer publishes a public key of the chameleon hash function, the signer can choose a specific sanitizer by using the sanitizer's public key. Only the sanitizer who knows the corresponding secret key can modify a signed message by finding collisions of the chameleon hash with a modified message. Several extensions of this approach was presented in [7].

While previous sanitizable signatures divide a message into submessages and then apply a cryptographic primitive (e.g., a hash function or a chameleon hash function) to each submessage, a new methodology utilizing a label was introduced in [11]. By marking the position of sanitizable submessages in a label, a sanitizable signature with constant length can be obtained.

At ACNS 2008, Canard et al. [12] introduced trapdoor sanitizable signatures, where the signer allows a specific user to modify pre-determined parts of a signed message by producing a piece of information (or trapdoor) that will help the user in sanitizing the message. The signer can choose to whom and when he will deliver the trapdoor information and hence, any user can act as a sanitizer; in other words, a sanitizer does not need any setup procedure such as generating a key pair. To implement the "trapdoor" requirement, Canard et al. [12] gave a construction based on the identity-based chameleon hash function [13]. Instead of the public key pk of chameleon hash functions, identity-based chameleon hash functions can use "identity strings" that are essentially any form of strings such as an e-mail address, a URL, a person's address, and any other unambiguous reference. A secret key sk_{ID} corresponding to an identity string ID is generated by an authority who knows a master secret key msk. To build a trapdoor sanitizable signature scheme, the signer keeps the master secret key msk of the identity-based chameleon hash function. The trapdoor information for sanitizing a message m is essentially the secret key sk_m with respect to the "identity string" m. Consequently, the signer can compute the trapdoor information at any time and give it to whomever he wants.

Our Contribution. We present a very simple and efficient approach to building trapdoor sanitizable signature schemes from ordinary signature schemes. Previous construction of trapdoor sanitizable signature [12] is based on the identity-based chameleon hash function (e.g., [13,14]), which, in turn, is based on the RSA signature or the bilinear maps. The identity-based chameleon hash function is evaluated for each sanitizable submessage. In contrast, our approach requires only one additional signature generation. When instantiated with an aggregate signature scheme (e.g., [15]), the trapdoor sanitizable signature consists of two group elements. The proposed construction answers the basic theoretic question about the minimal computational complexity assumption under which a trapdoor sanitizable signature exists; one-way functions imply trapdoor sanitizable signatures. If we borrow the terminology of [16], this shows that trapdoor sanitizable signature belongs to the "private cryptography world."

2 Preliminaries

If x is a string, then $|x|$ denotes its length, while if X is a finite set, $|X|$ denotes its size. If x and y are strings, $x\|y$ denotes the concatenation of x and y. If k is a positive integer, then $[1,k] = \{1,2,\ldots,k\}$. If $k \in \mathbb{N}$, then 1^k denotes the string of k ones. A function $f(k)$ is negligible if for all polynomial $p(k)$, $f(k) < 1/p(k)$ holds for all sufficiently large $k \in \mathbb{N}$. For a probability space P, $x \leftarrow P$ denotes the algorithm that samples a random element according to P. For a finite set X, $x \leftarrow X$ denotes the algorithm that samples an element uniformly at random from X. If $\mathcal{A}$ is a probabilistic polynomial time (PPT) algorithm, then $z \leftarrow \mathcal{A}(x, y, \ldots)$ denotes the operation of running $\mathcal{A}$ on inputs $x, y, \ldots$ and letting z be the output. If $p(\cdot,\cdot,\cdots)$ is a boolean function, then $\Pr[p(x_1, x_2, \cdots) \mid x_1 \leftarrow P_1,\ x_2 \leftarrow P_2,\ \cdots]$ denotes the probability that $p(x_1, x_2, \cdots)$ is true after executing algorithms $x_1 \leftarrow P_1, x_2 \leftarrow P_2, \cdots$.

Definition 1 (One-way Function). A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function, if there exists a polynomial time algorithm which computes $f(x)$ correctly for all x and the following probability is negligible in k for all PPT algorithm $\mathcal{A}$:

$$\Pr[f(x') = y \mid x \leftarrow \{0,1\}^k,\ y = f(x),\ x' \leftarrow \mathcal{A}(y, 1^k)] \qquad \Box$$

The existence of one-way functions is the most fundamental assumption of the complexity-based cryptography [17]. The definition of ordinary signature schemes is as follows.

Definition 2 (Digital Signature). A digital signature scheme $\mathcal{DS}$ is a 3-tuple of PPT algorithms ($\mathcal{K}$, $\mathcal{S}$, $\mathcal{V}$) that, respectively, generate keys for a user, sign a message, and verify the signature for a message.

- $\mathcal{K}$, the key generation algorithm, is a probabilistic algorithm that takes as input a security parameter and outputs a signing key and a verification key; $(SK, VK) \leftarrow \mathcal{K}(1^k)$.
- $\mathcal{S}$, the signing algorithm, is a probabilistic algorithm that takes as input a message $m \in \{0,1\}^*$ and a signing key SK. It outputs a signature or a special character $\perp$ indicating an error; $\sigma \leftarrow \mathcal{S}(m, SK)$.
- $\mathcal{V}$, the verification algorithm, is a deterministic algorithm that takes as input a message m, a signature σ, and a verification key VK. It outputs a single bit b indicating validity ($b = 1$) or invalidity ($b = 0$) of the signature; $b \leftarrow \mathcal{V}(m, \sigma, VK)$ where $b \in \{0,1\}$. $\Box$

We consider existential unforgeability under adaptive chosen message attacks, denoted by UF-CMA [18]. Let $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a signature scheme. The adversary $\mathcal{A}$ is given oracle access to the signing oracle $O^{\mathcal{S}}_{SK}(\cdot)$, i.e., $\mathcal{A}$ is allowed to query the signing oracle $O^{\mathcal{S}}_{SK}(\cdot)$ to obtain valid signatures $\sigma_1, \ldots, \sigma_\lambda$ of arbitrary messages $m_1, \ldots, m_\lambda$ adaptively chosen by $\mathcal{A}$. The adversary succeeds if it forges a valid signature σ of a message m that was not queried to $O^{\mathcal{S}}_{SK}(\cdot)$, i.e., $m \notin \{m_1, \ldots, m_\lambda\}$. An adversary $\mathcal{A}$ is said to (t, λ, ϵ)-break $\mathcal{DS}$, if $\mathcal{A}$ runs in time at most t, makes at most λ queries to the signing oracle, and succeeds in forgery with probability at least ϵ. The signature scheme $\mathcal{DS}$ is said to be (t, λ, ϵ)-secure, if no adversary can (t, λ, ϵ)-break it.

Definition 3 (Secure Signature). The adversary $\mathcal{A}$'s advantage against a signature scheme $\mathcal{DS}$ is defined by

$$\mathsf{Adv}^{\mathsf{uf\text{-}cma}}_{\mathcal{A},\mathcal{DS}}(k) = \Pr[\mathcal{V}(m, \sigma, VK) = 1 \mid (SK, VK) \leftarrow \mathcal{K}(1^k),\ (m, \sigma) \leftarrow \mathcal{A}^{O^{\mathcal{S}}_{SK}}(VK)]$$

where m should not be queried to the signing oracle. A signature scheme $\mathcal{DS}$ is secure if the advantage of any PPT adversary $\mathcal{A}$ is negligible in k. $\Box$

It is well known that a secure signature scheme can be built from a one-way function.

Theorem 1 ([19,20]). *Secure signatures exist if and only if one-way functions exist.* □

Boneh et al. [15] introduced an aggregate signature that allows incremental aggregation of signatures generated by multiple signers into one short signature based on bilinear groups. Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p and let g be a generater of $\mathbb{G}$. A bilinear map is a computable map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

– Bilinear: for all $g_1, g_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
– Non-degenerate: $e(g, g) \neq 1$.

A bilinear group is any group that possesses such a map e and on which CDH (Computational Diffie-Hellman) problem is hard; for given $g, g^a, h \in \mathbb{G}$, it is hard to compute $h^a \in \mathbb{G}$. Let $\mathsf{hash} : \mathbb{G} \times \{0,1\}^* \rightarrow \mathbb{G}$ be a cryptographic hash function. The bilinear aggregate signature is defined as follows [21].

Key Generation. For a particular user, pick random $x \leftarrow \mathbb{Z}_p$ and compute $v = g^x$. The user's signing key is (x, v) and the verification key is v.

Signing. For a particular user, given a message $m \in \{0,1\}^*$ and the signing key (x, v), compute $h = \mathsf{hash}(v \| m)$ and $\sigma = h^x$. The signature is $\sigma \in \mathbb{G}$.

Verification. Given a user's verification key v, a message m, and a signature σ, compute $h = \mathsf{hash}(v \| m)$; accept if $e(\sigma, g) = e(h, v)$.

Aggregation. Arbitrarily assign to each user whose signature will be aggregated an index i, ranging from 1 to λ. Each user i provides a signature $\sigma_i \in \mathbb{G}$ on a message $m_i \in \{0,1\}^*$. Compute $\sigma = \prod_{i=t}^{\lambda} \sigma_i$. The aggregate signature is $\sigma \in \mathbb{G}$.

Aggregation Verification. We are given an aggregate signature $\sigma \in \mathbb{G}$ for a set of users, indexed as before, and are given the messages $m_i \in \{0,1\}^*$ and verification keys $v_i \in \mathbb{G}$. To verify the aggregate signature σ, compute $h_i = \mathsf{hash}(v_i \| m_i)$ for $1 \leq i \leq \lambda$, and accept if $e(\sigma, g) = \prod_{i=1}^{\lambda} e(h_i, v_i)$ holds.

3 Trapdoor Sanitizable Signatures

3.1 Definition

A message m is composed of L submessages i.e., $m = m_1 \| \ldots \| m_L$. We assume that each submessage (implicitly or explicitly) includes an index to check that m_i is the i-th submessage; otherwise, we can simply insert an index (e.g., $m_i = i \| m_i$). Let $I_{\mathsf{adm}} \subset [1, L]$ be a set of indices of admissible submessages that can be modified by a sanitizer. We say that I_{adm} is consistent with (m, m') for an original message m and a modified message m' if $\{i \in [1, L] \mid m_i \neq m'_i\} \subset I_{\mathsf{adm}}$. To put it another way, I_{adm} is consistent with (m, m') if a modified message m' agrees with the signer's intention.

Definition 4 (Trapdoor Sanitizable Signature). A trapdoor sanitizable signature scheme TSS is a 6-tuple of PPT algorithms (Setup, KeyGen, Sign, $\mathsf{Trapdoor}$, $\mathsf{Sanitize}$, Vrfy) that, respectively, generate a public parameter, generate keys for a user, sign a message, generate a trapdoor for sanitization, sanitize a signed message, and verify the signature for a message.

- Setup, the parameter generation algorithm, is a probabilistic algorithm that takes as input a security parameter 1^k for $k \in \mathbb{N}$ and outputs a public parameter; $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$.
- KeyGen, the key generation algorithm, is a probabilistic algorithm that takes as input a public parameter and outputs a secret key and a public key; $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$.
- Sign, the signing algorithm, is a probabilistic algorithm that takes as input a message $m = m_1 \| m_2 \| \cdots \| m_L$ for $m_i \in \{0,1\}^*$ and $L \in \mathbb{N}$, a secret key sk, and a set of indices $I_{\mathsf{adm}} \subset [1, L]$. It outputs a signature or a special character $\perp$ indicating an error; $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}, I_{\mathsf{adm}})$.
- $\mathsf{Trapdoor}$, the trapdoor generation algorithm, is a probabilistic algorithm that takes as input a message m, a signature σ, a secret key sk, and a set of indices I_{adm}. It outputs a trapdoor or a special character $\perp$ indicating an error; $\mathsf{td} \leftarrow \mathsf{Trapdoor}(m, \sigma, \mathsf{sk}, I_{\mathsf{adm}})$.
- $\mathsf{Sanitize}$, the sanitization algorithm, is a probabilistic algorithm that takes as input a message m, a signature σ, a public key pk, a trapdoor td, a set of indices I_{adm}, and a modified message m'. It outputs a new signature on the modified message or a special character $\perp$ indicating an error; $\sigma' = \mathsf{Sanitize}(m, \sigma, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')$.
- Vrfy, the verification algorithm, is a deterministic algorithm that takes as input a message m, a signature σ, a public key pk, and a set of indices I_{adm}. It outputs a single bit b indicating validity ($b = 1$) or invalidity ($b = 0$) of the signature; $b \leftarrow \mathsf{Vrfy}(m, \sigma, \mathsf{pk}, I_{\mathsf{adm}})$ where $b \in \{0, 1\}$.

The parameter generation algorithm Setup is optional; in this case, Setup is an identity function (i.e., $\mathsf{param} = 1^k$). We omit param in the input of Sign, $\mathsf{Trapdoor}$, $\mathsf{Sanitize}$, and Vrfy, because param can be included in the secret key sk and the public key pk, if necessary. □

Correctness. A trapdoor sanitizable signature scheme should satisfy the standard correctness property of ordinary signature schemes, saying that a genuinely signed message is accepted by the verification algorithm; for any security parameter $k \in \mathbb{N}$, any public parameter $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$, any key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, any message $m = m_1 \| \cdots \| m_L$ for $L \in \mathbb{N}$, any set of indices $I_{\mathsf{adm}} \subset [1, L]$, and any signature $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}, I_{\mathsf{adm}})$, we have

$$\mathsf{Vrfy}(m, \sigma, \mathsf{pk}, I_{\mathsf{adm}}) = 1.$$

In a trapdoor sanitizable signature scheme, a signature sanitized with a valid trapdoor should also be accepted by the verification algorithm; for any security parameter $k \in \mathbb{N}$, any public parameter $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$, any key pair

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, any message $m = m_1 \| \cdots \| m_L$ for $L \in \mathbb{N}$, any set of indices $I_{\mathsf{adm}} \subset [1, L]$, any signature $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}, I)$, any trapdoor $\mathsf{td} \leftarrow \mathsf{Trapdoor}(m, \sigma, \mathsf{sk}, I_{\mathsf{adm}})$, and any sanitized signature $\sigma' = \mathsf{Sanitize}(m, \sigma, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')$ on a modified message $m' = m'_1 \| \cdots \| m'_L$ where $\{i \in [1, L] \mid m_i \neq m'_i\} \subset I_{\mathsf{adm}}$, we have

$$\mathsf{Vrfy}(m', \sigma', \mathsf{pk}, I_{\mathsf{adm}}) = 1.$$

Moreover, the sanitization process can be repeated; for any (original or sanitized) signature σ on a (original or sanitized) message m with $\mathsf{Vrfy}(m, \sigma, \mathsf{pk}, I_{\mathsf{adm}}) = 1$, a sanitized signature $\sigma' = \mathsf{Sanitize}(m, \sigma, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')$ on a modified message $m' = m'_1 \| \cdots \| m'_L$ where $\{i \in [1, L] \mid m_i \neq m'_i\} \subset I_{\mathsf{adm}}$ is valid;

$$\mathsf{Vrfy}(m', \sigma', \mathsf{pk}, I_{\mathsf{adm}}) = 1.$$

3.2 Security Model

Ateniese et al. [3] informally introduced five security requirements of sanitizable signature schemes.

- Unforgeability. No outsider should be able to forger the signer's or the sanitizer's signature.
- Immutability. The sanitizer should not be able to modify any part of the message that is not designated as sanitizable by the original signer.
- Transparency. Given a signed message with a valid signature, it should be infeasible to decide whether the message has been sanitized or not.
- Privacy. Sanitized messages and their signatures should not reveal the original data (i.e., the parts which have been sanitized).
- Accountability. In case of a dispute, the signer can prove to a trusted third party (e.g., court) that a certain message was modified by the sanitizer.

Note that unforgeability considers security against outsiders, while immutability against insiders (or sanitizers). These five security requirements were rigorously formalized according to game-based approaches by Brzuska et al. [8]. They distinguished between signer-accountability and sanitizer-accountability and investigated the relationship of the security requirements.

When it comes to trapdoor sanitizable signature schemes, Canard et al. [12] formalized two security requirements: unforgeability and indistinguishability. We rephrase the security model of [12] with game-based approaches. We first define the following oracles that are initialized with a key pair $(\mathsf{sk}, \mathsf{pk})$ and a random bit $b \in \{0, 1\}$.

- A signing oracle $O^{\mathsf{Sign}}_{\mathsf{sk}}(\cdot, \cdot)$ takes a message $m = m_1 \| \cdots \| m_L$, and a set of indices $I_{\mathsf{adm}} \subset [1, L]$ as input and outputs $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}, I_{\mathsf{adm}})$.
- A trapdoor generation oracle $O^{\mathsf{Trapdoor}}_{\mathsf{sk}}(\cdot, \cdot, \cdot)$ takes a message m, a signature σ, and a set of indices I_{adm} as input and outputs $\mathsf{td} \leftarrow \mathsf{Trapdoor}(m, \sigma, \mathsf{sk}, I_{\mathsf{adm}})$.
- A sanitization oracle $O^{\mathsf{Sanitize}}_{\mathsf{sk}}(\cdot, \cdot, \cdot, \cdot)$ takes as input a message m, a signature σ, a set of indices I_{adm}, and a modified message m'. It outputs $\sigma' \leftarrow \mathsf{Sanitize}(m, \sigma, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')$, where $\mathsf{td} \leftarrow \mathsf{Trapdoor}(m, \sigma, \mathsf{sk}, I_{\mathsf{adm}})$.

- A sanitization-or-signing oracle $O^{\mathsf{Sanitize/Sign}}_{\mathsf{sk},b}(\cdot,\cdot,\cdot)$ takes as input a message m, a set of indices I_{adm}, and a modified message m', where I_{adm} should be consistent with (m, m'). If $b = 0$, it outputs $\sigma' \leftarrow \mathsf{Sanitize}(m, \sigma, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')$ where $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}, I_{\mathsf{adm}})$ and $\mathsf{td} \leftarrow \mathsf{Trapdoor}(m, \sigma, \mathsf{sk}, I_{\mathsf{adm}})$. If $b = 1$, it outputs $\sigma' \leftarrow \mathsf{Sign}(m', \mathsf{sk}, I_{\mathsf{adm}})$.
- A left-or-right oracle $O^{\mathsf{LoR}}_{\mathsf{sk},b}(\cdot,\cdot,\cdot,\cdot)$ takes as input two messages m_1, m_2 composed of the same number of submessages, a set of indices I_{adm} and a modified message m', where I_{adm} should be consistent with both (m_1, m') and (m_2, m'). It outputs $\sigma' \leftarrow \mathsf{Sanitize}(m_b, \sigma_b, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')$ where $\sigma_b \leftarrow \mathsf{Sign}(m_b, \mathsf{sk}, I_{\mathsf{adm}})$ and $\mathsf{td} \leftarrow \mathsf{Trapdoor}(m_b, \sigma_b, \mathsf{sk}, I_{\mathsf{adm}})$.

In trapdoor sanitizable signature schemes, a signer can choose any user as a sanitizer just by giving the trapdoor information; thus no clear distinction exists between outsiders and insiders. Accordingly, unforgeability of trapdoor sanitizable signature schemes is defined to include the immutability requirement.

Definition 5 (Unforgeability [12]). A trapdoor sanitizable signature scheme TSS is unforgeable if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following game is negligible in the security parameter k:

1. A public parameter param and a key pair $(\mathsf{sk}, \mathsf{pk})$ are generated using $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$ and $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$. The public parameter param and the public key pk are given to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to a signing oracle $O^{\mathsf{Sign}}_{\mathsf{sk}}(\cdot,\cdot)$, a trapdoor generation oracle $O^{\mathsf{Trapdoor}}_{\mathsf{sk}}(\cdot,\cdot,\cdot)$, and a sanitization oracle $O^{\mathsf{Sanitize}}_{\mathsf{sk}}(\cdot,\cdot,\cdot,\cdot)$.
3. $\mathcal{A}$ outputs $(m^*, \sigma^*, I^*_{\mathsf{adm}})$ and succeeds if the following conditions hold.
   (a) $\mathsf{Vrfy}(m^*, \sigma^*, \mathsf{pk}, I^*_{\mathsf{adm}}) = 1$.
   (b) (m^*, σ^*) did not come from the signing oracle. In other words, $\mathcal{A}$ never queried $(m^*, \cdot)$ to the signing oracle.
   (c) (m^*, σ^*) did not come from the sanitization oracle. In other words, $\mathcal{A}$ never queried $(\cdot,\cdot,\cdot,m^*)$ to the sanitization oracle.
   (d) (m^*, σ^*) is not linked to a tuple (td, m, σ) from the trapdoor generation oracle. More precisely, for any message m being in the input of $O^{\mathsf{Trapdoor}}_{\mathsf{sk}}$, there is an index $i \notin I_{\mathsf{adm}}$ such that $m^*_i \neq m_i$, where I_{adm} corresponds to (m, σ). □

The indistinguishability requirement of [12] demands that the output distributions of the sanitization algorithm and the signing algorithm should be identical. That is, the following distributions $\mathcal{D}_{\mathsf{Sanitize}}$ and $\mathcal{D}_{\mathsf{Sign}}$ should be statistically indistinguishable for all key pairs $(\mathsf{sk}, \mathsf{pk})$ and messages $m, m' \in \{0,1\}^*$:

$$\begin{aligned}\mathcal{D}_{\mathsf{Sanitize}} = \{\sigma' \mid\ & \sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk}, I_{\mathsf{adm}}),\ \mathsf{td} \leftarrow \mathsf{Trapdoor}(m, \sigma, \mathsf{sk}, I_{\mathsf{adm}}),\\ & \sigma' \leftarrow \mathsf{Sanitize}(m, \sigma, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')\}\\ \mathcal{D}_{\mathsf{Sign}} = \{\sigma' \mid\ & \sigma' \leftarrow \mathsf{Sign}(m', \mathsf{sk}, I_{\mathsf{adm}})\}\end{aligned}$$

As the indistinguishability requirement of [12] considers statistical difference between the output distributions of the sanitization algorithm and the signing algorithm, the distinguisher is not given any other information (through oracle queries). By contrast, the transparency requirement of [8], which is defined with computational indistinguishability, captures the indistinguishability requirement of [12] with game-based approaches. We give the transparency requirement of trapdoor sanitizable signatures based on the definition of [8]. The adversary can probe the output distributions adaptively through oracle queries and have additional information by accessing the signing oracle, the sanitization oracle, and the trapdoor generation oracle.

Definition 6 (Transparency). A trapdoor sanitizable signature scheme TSS is transparent if the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible in the security parameter k:

1. A public parameter param and a key pair $(\mathsf{sk}, \mathsf{pk})$ are generated using $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$ and $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$. The public parameter param and the public key pk are given to $\mathcal{A}$.
2. A bit b is randomly chosen and the adversary $\mathcal{A}$ is given access to a signing oracle $O_{\mathsf{sk}}^{\mathsf{Sign}}(\cdot,\cdot)$, a trapdoor generation oracle $O_{\mathsf{sk}}^{\mathsf{Trapdoor}}(\cdot,\cdot,\cdot)$, a sanitization oracle $O_{\mathsf{sk}}^{\mathsf{Sanitize}}(\cdot,\cdot,\cdot,\cdot)$, and a sanitization-or-signing oracle $O_{\mathsf{sk},b}^{\mathsf{Sanitize/Sign}}(\cdot,\cdot,\cdot)$.
3. $\mathcal{A}$ outputs a guess b'.

We say that the adversary $\mathcal{A}$ succeeds if $b' = b$, and denote the probability of this event by $\Pr_{\mathcal{A},\mathsf{TSS}}^{\mathsf{trans}}[\mathsf{Succ}]$. The adversary's advantage is defined as $|\Pr_{\mathcal{A},\mathsf{TSS}}^{\mathsf{trans}}[\mathsf{Succ}] - 1/2|$. □

There are two equivalent ways in formalizing privacy for sanitizable signature schemes [12]; one approach follows semantic security of encryption schemes and the other is based on the indistinguishability notion for encryption. We define the privacy requirement of trapdoor sanitizable signature schemes with the more handy approach of the indistinguishability notion.

Definition 7 (Privacy). A trapdoor sanitizable signature scheme TSS is private if the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible in the security parameter k:

1. A public parameter param and a key pair $(\mathsf{sk}, \mathsf{pk})$ are generated using $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$ and $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$. The public parameter param and the public key pk are given to $\mathcal{A}$.
2. A bit b is randomly chosen and the adversary $\mathcal{A}$ is given access to a signing oracle $O_{\mathsf{sk}}^{\mathsf{Sign}}(\cdot,\cdot)$, a trapdoor generation oracle $O_{\mathsf{sk}}^{\mathsf{Trapdoor}}(\cdot,\cdot,\cdot)$, a sanitization oracle $O_{\mathsf{sk}}^{\mathsf{Sanitize}}(\cdot,\cdot,\cdot,\cdot)$, and a left-or-right oracle $O_{\mathsf{sk},b}^{\mathsf{LoR}}(\cdot,\cdot,\cdot,\cdot)$.
3. $\mathcal{A}$ outputs a guess b'.

We say that the adversary $\mathcal{A}$ succeeds if $b' = b$, and denote the probability of this event by $\Pr_{\mathcal{A},\mathsf{TSS}}^{\mathsf{priv}}[\mathsf{Succ}]$. The adversary's advantage is defined as $|\Pr_{\mathcal{A},\mathsf{TSS}}^{\mathsf{priv}}[\mathsf{Succ}] - 1/2|$. □

Brzuska et al. [8] showed that transparency implies privacy in sanitizable signatures. Their proof of implication can be straightforwardly translated into the case of trapdoor sanitizable signatures and hence, the same implication holds in trapdoor sanitizable signatures.

Theorem 2. *A transparent trapdoor sanitizable signature scheme is private.*

Proof. In a similar manner to [8], we convert an adversary $\mathcal{A}_{\mathsf{priv}}$ against privacy with a non-negligible advantage ϵ into $\mathcal{A}_{\mathsf{trans}}$ against transparency with a non-negligible advantage $\epsilon/2$ as follows.

```
$\mathcal{A}_{\mathsf{trans}}^{O_{\mathsf{sk}}^{\mathsf{Sign}}(\cdot,\cdot),O_{\mathsf{sk}}^{\mathsf{Trapdoor}}(\cdot,\cdot,\cdot),O_{\mathsf{sk}}^{\mathsf{Sanitize}}(\cdot,\cdot,\cdot),O_{\mathsf{sk},b}^{\mathsf{Sanitize/Sign}}(\cdot,\cdot,\cdot)}(\mathsf{param},\mathsf{pk})$
    $b^* \leftarrow \{0,1\}$
    $a \leftarrow \mathcal{A}_{\mathsf{priv}}^{O_{\mathsf{sk}}^{\mathsf{Sign}}(\cdot,\cdot),O_{\mathsf{sk}}^{\mathsf{Trapdoor}}(\cdot,\cdot,\cdot),O_{\mathsf{sk}}^{\mathsf{Sanitize}}(\cdot,\cdot,\cdot),O_{\mathsf{sk},b^*}^{\mathsf{LoR}}(\cdot,\cdot,\cdot)}(\mathsf{param},\mathsf{pk})$
        where $O_{\mathsf{sk},b^*}^{\mathsf{LoR}}(m_0,m_1,I_{\mathsf{adm}},m') = O_{\mathsf{sk},b}^{\mathsf{Sanitize/Sign}}(m_{b^*},I_{\mathsf{adm}},m')$ for every query on $(m_0,m_1,I_{\mathsf{adm}},m')$ that $\mathcal{A}_{\mathsf{priv}}$ sends to the left-or-right oracle.
    Return $b'=0$ if $a=b^*$, otherwise $b'=1$
```

$\mathcal{A}_{\mathsf{trans}}$ gets a public parameter param and a public key pk as input and has access to $O_{\mathsf{sk}}^{\mathsf{Sign}}(\cdot,\cdot)$, $O_{\mathsf{sk}}^{\mathsf{Trapdoor}}(\cdot,\cdot,\cdot)$, $O_{\mathsf{sk}}^{\mathsf{Sanitize}}(\cdot,\cdot,\cdot,\cdot)$, and $O_{\mathsf{sk},b}^{\mathsf{Sanitize/Sign}}(\cdot,\cdot,\cdot)$. The goal of $\mathcal{A}_{\mathsf{trans}}$ is to guess a random bit b with non-negligible advantage, i.e., to distinguish whether the message has been sanitized or not. $\mathcal{A}_{\mathsf{trans}}$ simulates the attack environment of $\mathcal{A}_{\mathsf{priv}}$ and utilizes $\mathcal{A}_{\mathsf{priv}}$ as a subroutine. For signing, trapdoor generation, and sanitizing queries of $\mathcal{A}_{\mathsf{priv}}$, $\mathcal{A}_{\mathsf{trans}}$ forwards them to its own oracles and hands the answers back to $\mathcal{A}_{\mathsf{priv}}$. For a left-or-right query of $\mathcal{A}_{\mathsf{priv}}$ on $(m_0,m_1,I_{\mathsf{adm}},m')$, $\mathcal{A}_{\mathsf{trans}}$ sends $(m_{b^*},I_{\mathsf{adm}},m')$ to the sanitization-or-signing oracle and hands the answer back to $\mathcal{A}_{\mathsf{priv}}$. Eventually, $\mathcal{A}_{\mathsf{priv}}$ outputs its guess a, and $\mathcal{A}_{\mathsf{trans}}$ outputs $b'=0$ iff $a=b^*$.

Now, we consider the success probability of $\mathcal{A}_{\mathsf{priv}}$. If $b=0$, then the simulation is identical to the actual attack environment of $\mathcal{A}_{\mathsf{priv}}$ with a random bit b^* and we have $\Pr[b'=0|b=0]=\Pr[\mathcal{A}_{\mathsf{priv}}=b^*]\geq\frac{1}{2}+\epsilon$. If $b=1$, then no information on the bit b^* is given to $\mathcal{A}_{\mathsf{priv}}$ and we have $\Pr[b'=1|b=1]=\frac{1}{2}$. Therefore, the advantage Adv of $\mathcal{A}_{\mathsf{trans}}$ is given as follows.

$$\begin{aligned}\mathsf{Adv} &= \left|\Pr[b'=b]-\frac{1}{2}\right| \\ &= \left|\Pr[b=0]\cdot\Pr[b'=0|b=0]+\Pr[b=1]\cdot\Pr[b'=1|b=1]-\frac{1}{2}\right| \\ &\geq \left|\frac{1}{2}\cdot\left(\frac{1}{2}+\epsilon\right)+\frac{1}{2}\cdot\frac{1}{2}-\frac{1}{2}\right| = \frac{\epsilon}{2}\end{aligned}$$ □

Accountability of sanitizable signatures requires that the origin of a (sanitized) signed message should be undeniable [3,8]. To recognize a sanitizer in any meaningful way, the sanitizer must have some secret information. However, there is no such information in trapdoor sanitizable signatures, because any user, just with a trapdoor, can act as a sanitizer. Therefore, accountability cannot be defined in trapdoor sanitizable signatures.

In summary, the security model of trapdoor sanitizable signatures considers unforgeability and transparency because the definition of unforgeability includes immutability, the transparency requirement implies privacy, and the accountability requirement cannot be defined in a meaningful way.

Definition 8 (Security). A trapdoor sanitizable signature scheme TSS is secure if it is unforgeable and transparent. $\square$

4 Generic Construction from Ordinary Signatures

The previous construction of trapdoor sanitizable signature schemes is based on identity-based chameleon hash functions to implement the trapdoor generation functionality [8]. Identity-based chameleon hash functions are not as efficient as cryptographic hash functions such as SHA-1; rather they are a kind of public-key primitive in terms of functionality and efficiency. Currently known identity-based chameleon hash functions (e.g., [13,14]) are based on the RSA signature or the bilinear maps. Consequently, they are not more efficient than ordinary signature schemes. Recall that the purpose of trapdoor is to delegate the power of signing on a message. Our approach to implementing the trapdoor generation functionality is to generate a temporary key pair $(\hat{SK}, \hat{VK})$ of an ordinary signature scheme and use the signing key $\hat{SK}$ as a trapdoor. This is simple and intuitive. We first give a stateful construction that stores $(\hat{SK}, \hat{VK})$ in a list and then discuss how to remove it.

Construction 1 (Stateful Version). Let $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be an ordinary signature scheme.

- Setup. The public parameter generation algorithm is an identity function. It takes a security parameter 1^k as input and sets $\mathsf{param} = 1^k$.
- KeyGen. The key generation algorithm takes $\mathsf{param} = 1^k$ as input and generates a long-term key pair by executing $(SK, VK) \leftarrow \mathcal{K}(1^k)$. The secret key is $\mathsf{sk} = SK$ and the public key is $\mathsf{pk} = VK$.
- Sign. The signing algorithm takes as input a message $m = m_1 \| m_2 \| \cdots \| m_L$, a secret key $\mathsf{sk} = SK$, and a set of indices $I_{\mathsf{adm}} \subset [1, L]$. It generates a temporary key pair $(\hat{SK}, \hat{VK}) \leftarrow \mathcal{K}(1^k)$ and forms $\hat{m}$ from m and I_{adm} as follows.
$$\hat{m} = \hat{VK} \| \hat{m}_1 \| \hat{m}_2 \| \cdots \| \hat{m}_L, \quad \text{where } \hat{m}_i = \begin{cases} \star & \text{if } \; i \in I_{\mathsf{adm}} \\ m_i & \text{if } \; i \notin I_{\mathsf{adm}} \end{cases} \tag{1}$$
where $\star$ is a special character for space holder. The signing algorithm computes $\sigma_0 \leftarrow \mathcal{S}(\hat{m}, SK)$ and $\sigma_1 \leftarrow \mathcal{S}(\hat{m} \| m, \hat{SK})$. The signature is $\sigma = (\hat{VK}, \sigma_0, \sigma_1)$. The temporary key pair $(\hat{SK}, \hat{VK})$ is kept in a list LIST for trapdoor generation.
- $\mathsf{Trapdoor}$. The trapdoor generation algorithm takes as input a message m, a signature σ, a secret key sk, and a set of indices I_{adm}. If σ is a valid signature on m and $\hat{VK}$ exists in LIST, it outputs a trapdoor $\mathsf{td} = \hat{SK}$ from LIST. Otherwise, it outputs $\perp$.

– Sanitize. The sanitization algorithm takes as input a message m, a signature $\sigma = (\hat{VK}, \sigma_0, \sigma_1)$, a public key $\mathsf{pk} = VK$, a trapdoor $\mathsf{td} = \hat{SK}$, a set of indices I_{adm}, and a modified message m'. If σ is not a valid signature on m, I_{adm} is not consistent with (m, m'), or $\hat{SK}$ is not a valid signing key with respect to $\hat{VK}$, then it outputs $\perp$. Otherwise, it forms $\hat{m}$ according to Equation (1) and computes $\sigma_1' \leftarrow \mathcal{S}(\hat{m} \| m', \hat{SK})$. The sanitized signature is $\sigma' = (\hat{VK}, \sigma_0, \sigma_1')$.
– Vrfy. The verification algorithm takes as input a message m, a signature $\sigma = (\hat{VK}, \sigma_0, \sigma_1)$, a public key $\mathsf{pk} = VK$, and a set of indices I_{adm}. It forms $\hat{m}$ according to Equation (1) and checks the validity of σ_0 and σ_1 by $b_0 \leftarrow \mathcal{V}(\hat{m}, \sigma_0, VK)$ and $b_1 \leftarrow \mathcal{V}(\hat{m} \| m, \sigma_1, \hat{VK})$. The verification algorithm outputs $b = b_0 \ \wedge \ b_1$. □

The signing algorithm Sign generates a signature σ_0 that is essentially a kind of "public-key certificate" of the verification key $\hat{VK}$. The content of the certificate (i.e. $\hat{m}$), which is similar to the label of [11], limits the usage of the key $\hat{VK}$; the corresponding signing key can only be used to generate signatures σ_1 with respect to messages m' where I_{adm} is consistent with (m, m'). This is natural because in a sense, a public-key certificate "delegates" a signing power to a user. The temporary key $\hat{VK}$ is also used for a message ID for $\hat{m} = \hat{VK} \| \hat{m}_1 \| \cdots \| \hat{m}_L$.

If we adopt an aggregate signature scheme, the values σ_0 and σ_1 in a signature $\sigma = (\hat{VK}, \sigma_0, \sigma_1)$ can be shortened into one value. For example, if we use the bilinear aggregate signature of Section 2, the signature $\sigma = (\hat{VK}, \sigma_0, \sigma_1)$ on a message m can be compressed into $\sigma = (\alpha, \beta) = (\hat{VK}, \sigma_{01}) = (v, h_0^x h_1^{\hat{x}})$ where $(\mathsf{sk}, \mathsf{pk}) = (x, v)$, $(\hat{SK}, \hat{VK}) = (\hat{x}, \hat{v})$, $h_0 = \mathsf{hash}(v \| \hat{m})$, and $h_1 = \mathsf{hash}(\hat{v} \| \hat{m} \| m)$. An updated signature σ' with a modified message m' and a trapdoor $\mathsf{td} = \hat{SK} = \hat{x}$ can be obtained by $\sigma' = (\alpha, \frac{\beta \sigma_1'}{\sigma_1})$ where $\sigma_1 = \mathsf{hash}(\hat{v} \| \hat{m} \| m)^{\hat{x}}$ and $\sigma_1' = \mathsf{hash}(\hat{v} \| \hat{m} \| m')^{\hat{x}}$. Note that sequential aggregate signatures (e.g., [22,23]) can be used in a similar manner, since σ_1 can always be computed after σ_0.

Theorem 3. *Construction 1 is a secure trapdoor sanitizable signature scheme if the underlying signature scheme is secure.*

Proof. Let $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a secure signature scheme and $\mathsf{TSS}_{\mathcal{DS}}$ be a trapdoor sanitizable signature scheme of Construction 1 instantiated with $\mathcal{DS}$. To prove the security of $\mathsf{TSS}_{\mathcal{DS}}$, we have to show that $\mathsf{TSS}_{\mathcal{DS}}$ is unforgeable and transparent.

UNFORGEABILITY. To show the unforgeability of $\mathsf{TSS}_{\mathcal{DS}}$, we build an adversary $\mathcal{A}$ against the signature scheme $\mathcal{DS}$ from an adversary $\mathcal{B}$ breaking the unforgeability of $\mathsf{TSS}_{\mathcal{DS}}$. Recall that $\mathcal{A}$ gets VK as input and has access to the signing oracle $O_{SK}^{\mathcal{S}}$, where (SK, VK) is a valid key pair of $\mathcal{DS}$. On the other hand, $\mathcal{B}$ gets $(\mathsf{param}, \mathsf{pk})$ as input and makes q_{sig} queries to the signing oracle $O_{\mathsf{sk}}^{\mathsf{Sign}}$, q_{tr} queries to the trapdoor generation oracle $O_{\mathsf{sk}}^{\mathsf{Trapdoor}}$, and q_{san} queries to the sanitization oracle $O_{\mathsf{sk}}^{\mathsf{Sanitize}}$. Since a query to the trapdoor generation oracle should contain a valid signature, we have $q_{sig} \geq q_{tr}$ by assuming that the adversary does not make two queries for the same trapdoor.

On input VK, the adversary $\mathcal{A}$ begins simulating the attack environment of $\mathcal{B}$. It first sets $\mathsf{param} = 1^k$ where $k = |VK|$ and generates a long-term key pair $(SK_0, VK_0) \leftarrow \mathcal{K}(1^k)$ and q_{sig} temporary key pairs $(SK_i, VK_i) \leftarrow \mathcal{K}(1^k)$ for $i \in [1, q_{sig}]$. It picks a random number $\mu \in [0, q_{sig}]$ and then sets keys as follows for $i \in [1, q_{sig}]$.

$$\mathsf{sk} = \begin{cases} SK_0 & \text{if } \mu \neq 0 \\ \perp & \text{if } \mu = 0 \end{cases} \qquad \hat{SK}_i = \begin{cases} SK_i & \text{if } \mu \neq i \\ \perp & \text{if } \mu = i \end{cases}$$

$$\mathsf{pk} = \begin{cases} VK_0 & \text{if } \mu \neq 0 \\ VK & \text{if } \mu = 0 \end{cases} \qquad \hat{VK}_i = \begin{cases} VK_i & \text{if } \mu \neq i \\ VK & \text{if } \mu = i \end{cases}$$

The random number μ is $\mathcal{A}$'s guess at the signing key for which $\mathcal{B}$ makes a forgery. If $\mathcal{A}$'s guess is correct, then $\mathcal{A}$ can break the security of $\mathcal{DS}$ from $\mathcal{B}$'s forgery. Otherwise, $\mathcal{A}$ aborts simulation without success. The adversary $\mathcal{A}$ stores all key pairs $(\hat{SK}_i, \hat{VK}_i)$ in LIST, gives $(\mathsf{param}, \mathsf{pk})$ to $\mathcal{B}$ as input, and answers the queries of $\mathcal{B}$ as follows.

– For the i-th signing query (m, I_{adm}), the adversary $\mathcal{A}$ forms $\hat{m}$ according to Equation (1). If $\mu \neq 0$, then $\mathcal{A}$ generates $\sigma_0 \leftarrow \mathcal{S}(\hat{m}, SK_0)$. Otherwise, $\mathcal{A}$ generates σ_0 by making a query $\hat{m}$ to its own signing oracle $O^{\mathcal{S}}_{SK}(\cdot)$. If $\mu \neq i$, then $\mathcal{A}$ generates $\sigma_1 \leftarrow \mathcal{S}(\hat{m}\|m, \hat{SK}_i)$. Otherwise, $\mathcal{A}$ generates σ_1 by making a query $\hat{m}\|m$ to $O^{\mathcal{S}}_{SK}(\cdot)$. The adversary $\mathcal{A}$ returns the signature $\sigma = (\hat{VK}_i, \sigma_0, \sigma_1)$.
– For the i-th trapdoor query $(m, \sigma, I_{\mathsf{adm}})$ where $\sigma = (\hat{VK}, \sigma_0, \sigma_1)$, the adversary $\mathcal{A}$ first checks the validity of σ. If $\mathsf{Vrfy}(m, \sigma, \mathsf{pk}, I_{\mathsf{adm}}) = 0$, then $\mathcal{A}$ returns $\perp$. If $\mathsf{Vrfy}(m, \sigma, \mathsf{pk}, I_{\mathsf{adm}}) = 1$ and $\hat{VK} = VK$, then $\mathcal{A}$ aborts the simulation. Otherwise, $\mathcal{A}$ returns $\hat{SK}$ corresponding to $\hat{VK}$ from List.
– For the i-th sanitization query $(m, \sigma, I_{\mathsf{adm}}, m')$ where $\sigma = (\hat{VK}, \sigma_0, \sigma_1)$, the adversary $\mathcal{A}$ first checks the validity of the query. If $\mathsf{Vrfy}(m, \sigma, \mathsf{pk}, I_{\mathsf{adm}}) = 0$ or I_{adm} is not consistent with (m, m'), then $\mathcal{A}$ returns $\perp$. Otherwise, $\mathcal{A}$ computes a sanitized signature as follows.
  - If $\hat{VK} \neq VK$, then $\mathcal{A}$ searches $\hat{SK}$ corresponding to $\hat{VK}$ from LIST and generates $\sigma_1' \leftarrow \mathcal{S}(\hat{m}\|m', \hat{SK})$.
  - Otherwise, $\mathcal{A}$ generates σ_1' by making a query $\hat{m}\|m'$ to its own signing oracle $O^{\mathcal{S}}_{SK}(\cdot)$.

  The adversary $\mathcal{A}$ returns a sanitized signature $\sigma' = (\hat{VK}, \sigma_0, \sigma_1')$.

Let $(m^*, \sigma^*, I^*_{\mathsf{adm}})$ be a successful forgery of $\mathcal{B}$, where $\sigma^* = (\hat{VK}^*, \sigma_0^*, \sigma_1^*)$. That is, $\mathsf{Vrfy}(m^*, \sigma^*, \mathsf{pk}, I^*_{\mathsf{adm}}) = 1$.

– If $\mu = 0$ and σ_0^* is a successful forgery with respect to VK, then $\mathcal{A}$ forms $\hat{m}^*$ from m^* and I^*_{adm} according to Equation (1) and returns $(\hat{m}^*, \sigma_0^*)$ as its own output.
– If $\mu = 0$ and σ_0^* is not a successful forgery with respect to VK, then $\mathcal{A}$ aborts.

– If $\mu \neq 0$ and $\hat{VK}^* = VK$, then $\mathcal{A}$ forms $\hat{m}^*$ from m^* and I^*_{adm} according to Equation (1) and returns $(\hat{m}^* \| m^*, \sigma^*_1)$ as its own output.
– Otherwise (i.e., $\mu \neq 0$ and $\hat{VK}^* \neq VK$), then $\mathcal{A}$ aborts.

If $\mathcal{A}$ does not abort (i.e., $\mu \in [0, q_{sig}]$ is a correct guess), then the simulation is perfect. As $q_{sig} \geq q_{tr}$, the event of $\mathcal{A}$'s not aborting happens with probability at least $1/(q_{sig}+1)$. Therefore, if $\mathcal{B}$ succeeds with a probability ϵ, the adversary $\mathcal{A}$ succeeds with probability at least $\epsilon/(q_{sig}+1)$.

Transparency. Recall that a sanitization-or-signing oracle $O^{\mathsf{Sanitize/Sign}}_{\mathsf{sk},b}(\cdot,\cdot,\cdot)$ takes as input a message m, a set of indices I_{adm}, and a modified message m', where I_{adm} should be consistent with (m, m'). If $b = 0$, it outputs $\sigma' \leftarrow \mathsf{Sanitize}(m, \sigma, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')$ and otherwise, it outputs $\sigma' \leftarrow \mathsf{Sign}(m', \mathsf{sk}, I_{\mathsf{adm}})$. The adversary's goal is to guess the random bit $b \in \{0,1\}$. For all messages m, m' and their consistent index set I_{adm}, the scheme $\mathsf{TSS}_{\mathcal{DS}}$ generates the signature $\sigma' = (\hat{VK}, \sigma_0, \sigma'_1)$ on message m' as follows.

$$\begin{aligned}
\hat{m}' &= \hat{VK} \| \hat{m}'_1 \| \hat{m}'_2 \| \cdots \| \hat{m}'_L, \quad \text{where } \hat{m}'_i = \begin{cases} \star & \text{if } i \in I_{\mathsf{adm}} \\ m'_i & \text{if } i \notin I_{\mathsf{adm}} \end{cases}, \\
\sigma_0 &\leftarrow \mathcal{S}(\hat{m}', SK), \\
\sigma'_1 &\leftarrow \mathcal{S}(\hat{m}' \| m', \hat{SK}).
\end{aligned}$$

This holds whether the signature σ' is given by $\sigma' \leftarrow \mathsf{Sanitize}(m, \sigma, \mathsf{pk}, \mathsf{td}, I_{\mathsf{adm}}, m')$ or $\sigma' \leftarrow \mathsf{Sign}(m', \mathsf{sk}, I_{\mathsf{adm}})$. In the $\mathsf{TSS}_{\mathcal{DS}}$ scheme, the sanitization-or-signing oracle $O^{\mathsf{Sanitize/Sign}}_{\mathsf{sk},b}(\cdot,\cdot,\cdot)$ works exactly in the same way irrespective of the random bit $b = \{0,1\}$. Therefore, any adversary, which is not necessarily a PPT algorithm, cannot guess the random bit b with a probability greater than 1/2, even with additional queries to the signing oracle, the trapdoor generation oracle, and the sanitization oracle. In other words, TSS is transparent in an information-theoretic sense. □

Our construction gives a simple answer to the basic theoretic question about the minimal computational complexity assumption under which a trapdoor sanitizable signature exists.

Theorem 4. *Secure trapdoor sanitizable signatures exist if and only if one-way functions exist.*

Proof. Theorem 1 and Theorem 3 proves that one-way functions imply secure trapdoor sanitizable signatures. For the other direction, we use a well-known technique of [20]. Let $\mathsf{TSS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Trapdoor}, \mathsf{Sanitize}, \mathsf{Vrfy})$ be a secure trapdoor sanitizable signature. We build a one-way function $f(\cdot)$ as follows; let $f(x)$ for $x \in \{0,1\}^k$ run $\mathsf{param} \leftarrow \mathsf{Setup}(x)$ and $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$ and output pk. Assume that there is an adversary $\mathcal{A}$ that can invert $f(\cdot)$. Then, given a public key pk, we would be able to obtain a secret key sk' with the property that sk' could generate signatures valid for pk. But this implies that TSS is insecure, which is a contradiction. □

Gertner et al. [16] minted the terms of "private cryptography world" and "public cryptography world" to divide cryptographic primitives into two groups. The former world consists of one-way function and all its equivalent primitives (e.g., pseudo-random generator, pseudo-random function, bit commitment, and digital signature) and the latter world consists of "harder" primitives such as key agreement, public key encryption, oblivious transfer, secure function evaluation, and trapdoor permutation. According to the terminology of Gertner et al. [16], Theorem 4 shows that trapdoor sanitizable signature belongs to the private cryptography world.

Stateless Version. There is a simple technique to remove the necessity of List in Construction 1. One can make $\mathcal{K}(\cdot)$, the key generation algorithm for temporary key pairs $(\hat{SK}, \hat{VK})$, deterministic. Let $\ell_1(\cdot)$ and $\ell_2(\cdot)$ be polynomials. Let $\mathsf{PRF}_k : \{0,1\}^{\ell_1(k)} \times \{0,1\}^* \rightarrow \{0,1\}^{\ell_2(k)}$ be a variable input-length pseudo-random function such as variable input-length MACs (message authentication codes). Let γ_k be the randomness used by $\mathcal{K}(1^k)$ where $|\gamma_k| = \ell_2(k)$. The randomness γ_k can be produced in a way that is deterministically dependent on $\hat{m}$. First, a random key $\kappa \in \{0,1\}^{\ell_1(k)}$ for PRF_k is generated and included in the secret key sk. To sign a message m, the randomness γ_k is computed by $\gamma_k = \mathsf{PRF}_k(\kappa, \hat{m})$. Then, the trapdoor (i.e., the temporary signing key $\hat{SK}$) can be re-computed from κ and $\hat{m}$ without consulting List. No PPT algorithm can distinguish between the stateless version and the stateful version because PRF_k is a pseudorandom function.

References

1. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
2. 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)
3. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
4. Izu, T., Kanaya, N., Takenaka, M., Yoshioka, T.: PIATS: A partially sanitizable signature scheme. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 72–83. Springer, Heidelberg (2005)
5. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions 88-A(1), 239–246 (2005)
6. Miyazaki, K., Hanaoka, G., Imai, H.: Invisibly sanitizable digital signature scheme. IEICE Transactions 91-A(1), 392–402 (2008)
7. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
8. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)

9. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS. The Internet Society (2000)
10. Ateniese, G., de Medeiros, B.: On the key exposure problem in chameleon hashes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 165–179. Springer, Heidelberg (2005)
11. Yum, D.H., Lee, P.J.: Sanitizable signatures reconsidered (2010) (unpublished manuscript)
12. Canard, S., Laguillaumie, F., Milhau, M.: Trapdoor sanitizable signatures and their application to content protection. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 258–276. Springer, Heidelberg (2008)
13. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
14. Zhang, F., Safavi-Naini, R., Susilo, W.: ID-based chameleon hashes from bilinear pairings. Cryptology ePrint Archive, Report 2003/208 (2003), http://eprint.iacr.org/
15. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
16. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: The 41st Annual Symposium on Foundations of Computer Science, pp. 325–335. IEEE, Los Alamitos (2000)
17. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity based cryptography (extended abstract). In: The 30th Annual Symposium on Foundations of Computer Science, pp. 230–235. IEEE Computer Society, Los Alamitos (1989)
18. 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)
19. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: The 21st Annual ACM Symposium on Theory of Computing, pp. 33–43. ACM, New York (1989)
20. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: The 22nd Annual ACM Symposium on Theory of Computing, pp. 387–394. ACM, New York (1990)
21. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: A survey of two signature aggregation techniques. CryptoBytes 6(2), 1–10 (2003)
22. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
23. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)

Generic Constructions for Verifiably Encrypted Signatures without Random Oracles or NIZKs

Markus Rückert[*], Michael Schneider, and Dominique Schröder[*]

Technische Universität Darmstadt, Germany
markus.rueckert@cased.de,
mischnei@cdc.informatik.tu-darmstadt.de,
schroeder@me.com

Abstract. Verifiably encrypted signature schemes (VES) allow a signer to encrypt his or her signature under the public key of a trusted third party, while maintaining public signature verifiability. With our work, we propose two generic constructions based on Merkle authentication trees that do not require non-interactive zero-knowledge proofs (NIZKs) for maintaining verifiability. Both are stateful and secure in the standard model. Furthermore, we extend the specification for VES, bringing it closer to real-world needs. We also argue that statefulness can be a feature in common business scenarios.

Our constructions rely on the assumption that CPA (even slightly weaker) secure encryption, "maskable" CMA secure signatures, and collision resistant hash functions exist. "Maskable" means that a signature can be hidden in a verifiable way using a secret masking value. Unmasking the signature is hard without knowing the secret masking value. We show that our constructions can be instantiated with a broad range of efficient signature and encryption schemes, including two lattice-based primitives. Thus, VES schemes can be based on the hardness of worst-case lattice problems, making them secure against subexponential and quantum-computer attacks. Among others, we provide the first efficient pairing-free instantiation in the standard model.

Keywords: Generic construction, Merkle tree, post-quantum, standard model.

1 Introduction

Boneh et al. introduced the concept of verifiably encrypted signatures (VES) at Eurocrypt 2003 as a means of covertly exchanging signatures, while maintaining their verifiability [8]. They include a passive, trusted third party, the adjudicator, which makes VES schemes fall into the category of optimistic fair exchange protocols [2,4].

[*] This work was supported by CASED (www.cased.de). Dominique Schröder is also supported by the Emmy Noether Programme Fi 940/2-1 of the German Research Foundation (DFG).

Signer, receiver, and adjudicator, interact as follows. The signer encrypts his or her signature σ for a document M as a verifiably encrypted signature ϖ. Given ϖ, the receiver can verify that it contains a valid signature for M, but is otherwise unable to extract σ. A commercially important application is *online contract signing* under the supervision of an escrow. As opposed to the classic (offline) scenario, where the escrow needs to be involved in every step of the process, verifiably encrypted signatures make signing contracts more cost-efficient due to the passiveness of the escrow and simultaneously allow the parties to negotiate online rather than meet in person. They simply exchange verifiably encrypted signatures on the negotiated contract, verify them, and finally exchange the corresponding regular signatures. Assume Alice acts honestly, while Bob tries to misuse (e.g., for negotiating a better deal elsewhere or simply for blackmail) the partially signed document without providing a signature himself. The adjudicator can step in and disclose Bob's signature. The contract becomes binding despite Bob's attempt to back out.

Boneh et al. propose the first construction [8], which is provably secure in the random oracle model (ROM), followed by a slightly more efficient construction by Zhang et al. [24]. Lu et al. present the first scheme in the standard model in [13]. Furthermore, they sketch a generic construction based on NIZKs. Another NIZK construction has been proposed by Dodis et al. in [11]. Using NIZKs, however, is generally very inefficient with respect to computational cost and signature size.

The previous efficient instantiations typically use pairings in order to achieve verifiability of El Gamal encrypted signatures. With pairings, however, security proofs have to rely on very special versions of the Diffie-Hellman problem, whose actual hardness is yet to be assessed [7]. Note that there is a second line of work on "verifiable encryption" in a more general scenario by Camenisch and Shoup [9] as well as Ateniese [3]. Their objectives differ from the one in [8] as Boneh et al. demand that transmitting and verifying an encrypted signature can be done in a single move. In particular, the verification process does not involve interactive ZK proofs as required in both [9] and [3]. Therefore, we focus on the line of research coming from Boneh et al.

Our Contribution

Generic construcions of cryptographic schemes help understand their complexity. A common approach is: take a message, encrypt it, and append a NIZK, proving that the encrypted value satisfies some property. Removing NIZKs is non-trivial. In the following, we discuss how they can be avoided in the context of VES.

VES with a Setup Phase. We extend the model of Boneh et al. in the sense that the signer's key may depend on the adjudicator's public key. More precisely, we allow signer and adjudicator to interact *once* during key generation. We believe that this is a good model of real-world business scenarios. To illustrate this, let's consider a notary, the adjudicator, that oversees fair online contract signing. In general, the notary wants to remain passive but he or she still wants to bill his or her services on a per signature-exchange basis. With our extension and the instantiations therein, we show how the (offline) notary can actually control the number of verifiably encrypted signature that his or her customers can securely exchange. The customer pays for a certain number of signatures in advance and the notary completes the customer's key pair accordingly. Interestingly, the underlying signature scheme can still be used in a black-box way. Thus, smart cards or other signing devices can be used and the secret signing key is not revealed. This is important for contract signing as laws and ordinances often require this for the contract to be legally binding.

Generic Construction. So far, there have been two construction principles for VES schemes: use pairings for efficiency or NIZKs for generic constructions from minimal cryptographic assumptions. Our construction fills the gap between those extremes as it can be considered both efficient (compared to NIZKs) and generic. We believe that our construction principles may also be helpful in finding NIZK-free generic constructions for other schemes. In detail, we show three things.

Firstly, generic constructions for VES schemes need not involve inefficient non-interactive zero-knowledge proofs. We propose two generic constructions in the standard model, which is encouraged by the work of Canetti, Goldreich, and Halevi [10]. Both are based on "random plaintext secure" (RPA) encryption[1], "maskable" existentially unforgeable (EU-CMA) signatures, and a collision-resistant hash function in a Merkle authentication tree [18] that can be handled efficiently. This allows us to scale the resulting scheme according to the individual needs of specific application scenarios.

Maskability is a property of the signature scheme that states that we can choose a random masking value α and mask a given signature σ for a message M by calling $(\tau, \beta) \leftarrow \mathsf{Mask}(\sigma, \alpha)$. The resulting masked signature τ is still valid for the same message under a modified verification procedure that uses some additional advice β. Given (τ, β), it is hard to recover σ. However, with the secret masking value α, one can call $\sigma' \leftarrow \mathsf{Unmask}(\tau, \beta, \alpha)$ and recover a valid (ordinary) signature for M.

Our first construction uses regular (many-time) signature schemes, the other solely requires the existence of a suitable one-time signature scheme. Both of our constructions are stateful and the key generation step depends, as always with tree-based constructions [18], on the desired signature capacity ℓ of the resulting scheme. Using Merkle trees for VES was first considered in [20] for the special case of RSA signatures. By formalizing this approach, we develop this technique to its full potential.

Secondly, pairings are not necessary for efficient VES schemes. In particular, we show the first pairing-free VES scheme in the standard model without NIZKs.

Thirdly, we introduce efficient VES schemes to the post-quantum era because we give lattice-based instantiations that withstand quantum computer and subexponential attacks. Previous instantiations will become insecure in the

[1] A weaker notion than CPA security, where the adversary has to distinguish the ciphertext for a *random* message from the encryption of the 0-string (see Section 3).

presence of quantum computers due to the work of Shor [23]. The full version contains a second instantiation from lattices and one from RSA.

Organization. We start by recalling the VES security model in Section 2. There, we also propose our extension. Then, we describe the required building blocks, including "maskability", in Section 3, followed by our generic constructions. In order to demonstrate their feasibility, we instantiate both constructions with various efficient primitives in Section 4 and the full version [21].

2 Verifiably Encrypted Signatures

Verifiably encrypted signature schemes support the encryption of signatures under the public key of a trusted third party, while simultaneously proving that the encryption contains a valid signature. They are built upon digital signature schemes $\mathsf{DSig} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$, defined via *Key Generation:* $\mathsf{Kg}(1^n)$ outputs a key pair $(\mathsf{ssk}, \mathsf{spk})$. ssk is a private signing key and spk the corresponding public verification key $\mathsf{spk} \in \mathcal{K}$ (public key space); *Signing:* $\mathsf{Sign}(\mathsf{ssk}, M)$ outputs a signature $\sigma \in \Sigma$ (signature space) on a message $M \in \mathcal{M}$ (message space) under ssk; *Signature Verification:* $\mathsf{Vf}(\mathsf{spk}, \sigma, M)$ outputs 1 iff σ is a valid signature on M under spk. Now, a verifiably encrypted signature scheme $\mathsf{VES} = (\mathsf{AdjKg}, \mathsf{AdjSetup}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$ consists of the following algorithms. Note that we generalize the model slightly by letting the key generation algorithm of the signer depend on the keys of the adjudicator (see $\mathsf{AdjSetup}$ below). In particular, we view the key generation process as an interactive algorithm between the adjudicator and the signer, i.e., the interaction takes place only once.

Adjudicator Key Generation: $\mathsf{AdjKg}(1^n)$ outputs a key pair $(\mathsf{ask}, \mathsf{apk})$, where ask is the private key and apk the corresponding public key.

Adjudication Setup: The adjudicator provides an algorithm $\mathsf{AdjSetup}(\mathsf{ask}, \mathsf{pk})$ whose input is the private key of the adjudicator ask and a public key pk of the signer. It returns a key pk'.

Key Generation: The key generation algorithm $\mathsf{Kg}(1^n)$ may interact with the adjudicator via the oracle $\mathsf{AdjSetup}(\mathsf{ask}, \cdot)$ to produce the key pair $(\mathsf{sk}, \mathsf{pk})$.

Signing and Verification: Same as in a digital signature scheme

VES Creation: $\mathsf{Create}(\mathsf{sk}, \mathsf{apk}, M)$ takes as input a secret key sk, the adjudicator's public key apk, and a message $M \in \mathcal{M}$. It returns a verifiably encrypted signature ϖ for M.

VES Verification: $\mathsf{VesVf}(\mathsf{apk}, \mathsf{pk}, \varpi, M)$ takes as input the adjudicator's public key apk, a public key pk, a verifiably encrypted signature ϖ, and a message M. It returns a bit.

Adjudication: $\mathsf{Adj}(\mathsf{ask}, \mathsf{apk}, \mathsf{pk}, \varpi, M)$ takes as input the key pair $(\mathsf{ask}, \mathsf{apk})$ of the adjudicator, the public key of the signer pk, a verifiably encrypted signature ϖ, and a message M. It extracts an ordinary signature σ for M.

A VES scheme is *complete* if for all adjudication key pairs $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{AdjKg}(1^n)$ and for all signature key pairs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}^{\mathsf{AdjSetup}(\mathsf{ask},\cdot)}(1^n)$ the following holds: $\mathsf{VesVf}(\mathsf{apk}, \mathsf{pk}, \mathsf{Create}(\mathsf{sk}, \mathsf{apk}, M), M) = 1$ and $\mathsf{Vf}(\mathsf{pk}, \mathsf{Adj}(\mathsf{ask}, \mathsf{apk}, \mathsf{pk}, \mathsf{Create}(\mathsf{sk}, \mathsf{apk}, M)), M) = 1$ for all $M \in \mathcal{M}$.

```
Experiment $\mathsf{VesForge}_{\mathcal{A}}^{\mathsf{VES}}(n)$
    $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{AdjKg}(1^n)$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}^{\mathsf{AdjSetup}(\mathsf{ask},\cdot)}(1^n)$
    $(M^*, \varpi^*) \leftarrow \mathcal{A}^{\mathfrak{C}(\mathsf{sk},\mathsf{apk},\cdot),\mathfrak{A}(\mathsf{ask},\mathsf{apk},\mathsf{pk},\cdot,\cdot),\mathfrak{S}(\mathsf{ask},\cdot)}(\mathsf{pk}, \mathsf{apk})$
    Return 1 iff $\mathsf{VesVf}(\mathsf{apk}, \mathsf{pk}, \varpi^*, M^*) = 1$ and
        $\mathcal{A}$ has never queried $M^*$ to $\mathfrak{C}(\mathsf{sk}, \mathsf{apk}, \cdot)$
        or $\mathfrak{A}(\mathsf{ask}, \mathsf{apk}, \mathsf{pk}, \cdot, \cdot)$.
```

```
Experiment $\mathsf{Opac}_{\mathcal{A}}^{\mathsf{VES}}(n)$
    $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{AdjKg}(1^n)$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}^{\mathsf{AdjSetup}(\mathsf{ask},\cdot)}(1^n)$
    $(M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathfrak{C}(\mathsf{sk},\mathsf{apk},\cdot),\mathfrak{A}(\mathsf{ask},\mathsf{apk},\mathsf{pk},\cdot,\cdot),\mathfrak{S}(\mathsf{ask},\cdot)}(\mathsf{pk}, \mathsf{apk})$
    Return 1 iff $\mathsf{Vf}(\mathsf{pk}, \sigma^*, M^*) = 1$ and
        $\mathcal{A}$ has never queried $M^*$ to $\mathfrak{A}(\mathsf{ask}, \mathsf{apk}, \mathsf{pk}, \cdot, \cdot)$.
```

```
Experiment $\mathsf{Extract}_{\mathcal{A}}^{\mathsf{VES}}(n)$
    $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{AdjKg}(1^n)$
    $(M^*, \varpi^*, \mathsf{pk}^*) \leftarrow \mathcal{A}^{\mathfrak{A}(\mathsf{ask},\mathsf{apk},\cdot,\cdot,\cdot),\mathfrak{S}(\mathsf{ask},\cdot)}(\mathsf{apk})$
    Let $\sigma^* \leftarrow \mathsf{Adj}(\mathsf{ask}, \mathsf{apk}, \mathsf{pk}^*, \varpi^*, M^*)$
    Return 1 iff $\mathsf{VesVf}(\mathsf{apk}, \mathsf{pk}^*, \varpi^*, M^*) = 1$
        and $\mathsf{Vf}(\mathsf{pk}^*, \sigma^*, M^*) = 0$.
```

```
Experiment $\mathsf{Collusion}_{\mathcal{A}}^{\mathsf{VES}}(n)$
    $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{AdjKg}(1^n)$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}^{\mathsf{AdjSetup}(\mathsf{ask},\cdot)}(1^n)$
    $\mathsf{state} \leftarrow (\mathsf{apk}, \mathsf{ask}, \mathsf{pk})$
    $(M^*, \varpi^*) \leftarrow \mathcal{A}^{\mathfrak{C}(\mathsf{sk},\mathsf{apk},\cdot)}(\mathsf{state})$
    Return 1 iff $\mathsf{VesVf}(\mathsf{apk}, \mathsf{pk}, \varpi^*, M^*) = 1$ and
        $\mathcal{A}$ has never queried $\mathfrak{C}(\mathsf{pk}, \mathsf{apk}, \cdot)$ about $M^*$.
```

Fig. 1. Overview over the different security experiments

2.1 Security Model

Security of verifiably encrypted signatures is defined by unforgeability, opacity [8], extractability, and collusion-resistance [22].[2] *Unforgeability* requires that it is hard to forge a verifiably encrypted signature, *opacity* implies that it is difficult to extract an ordinary signature from an encrypted signature, *extractability* guarantees that the adjudicator can always extract a regular signature from a valid verifiably encrypted signature, and *collusion-resistance* prevents signer and adjudicator from successfully colluding in order to produce a verifiably encrypted signature on behalf of another party, provided that the collusion happens in the online phase and *not* during key registration. The security requirement can be interpreted as a stronger form of unforgeability.

Since we allow the key generation algorithm to depend on the interaction with the adjudicator, we also give the adversary access to the corresponding oracle. Unforgeability and opacity are formalized in experiments, where the adversary is given the public keys of the signer and of the adjudicator. Moreover, the adversary has access to three oracles: $\mathfrak{C}$ returns verifiably encrypted signatures for a given message, $\mathfrak{A}$ extracts a regular signature from a verifiably encrypted signature, and $\mathfrak{S}$ allows the adversary to perform setup queries. In the extractability experiment, the adversarial signer is given access to an adjudication oracle and wins if he or she can output an encrypted signature that is hidden irrecoverably. Finally, the collusion-resistance experiment gives the adversary direct access to the adjudicator's private key. The goal is to forge a signature for another party. All experiments are defined in Figure 1.

As usual, n is the security parameter. VES is *secure* if the following holds for any efficient adversary $\mathcal{A}$:

Unforgeability: VES is unforgeable if $\mathsf{VesForge}_{\mathcal{A}}^{\mathsf{VES}}(n)$ outputs 1 with negligible probability.

[2] Note that in [22] this was called "abuse-freeness". Here, however, we prefer "collusion-resistance" because abuse-freeness already has a slightly different meaning in the context of fair-exchange, which creates confusion.

Opacity: VES is opaque if $\mathsf{Opac}_{\mathcal{A}}^{\mathsf{VES}}(n)$ outputs 1 with negligible probability.

Extractability: VES is extractable if $\mathsf{Extract}_{\mathcal{A}}^{\mathsf{VES}}(n)$ outputs 1 with negligible probability.

Collusion-Resistance: VES is *collusion-resistant* if $\mathsf{Collusion}_{\mathcal{A}}^{\mathsf{VES}}(n)$ outputs 1 with negligible probability. In this experiment the adversary gets as input the adjudication key pair and a signer public key pk. The adversary gets access to a VES creation oracle $\mathfrak{C}(\mathsf{sk}, \mathsf{apk}, \cdot)$. It does not need the other oracles because it has ask.

A scheme is $(t, q_{\mathfrak{C}}, q_{\mathfrak{A}}, q_{\mathfrak{S}}, \epsilon)$-secure, if no adversary, running in time at most t, making at most $q_{\mathfrak{C}}$ verifiably encrypted signature oracle queries, at most $q_{\mathfrak{A}}$ adjudication oracle queries, and at most $q_{\mathfrak{S}}$ key registration queries, can succeed with probability at least ϵ in the $\mathsf{VesForge}$, Opac, $\mathsf{Extract}$ (with $q_{\mathfrak{C}} = 0$), or $\mathsf{Collusion}$ (with $q_{\mathfrak{A}} = q_{\mathfrak{S}} = 0$) experiment.

Since we have extended the model slightly, we have to re-prove the relations among the different properties. The relations in [22] still hold in our setting as phrased in the following propositions. We defer the details to the full version.

Proposition 1. *If unforgeable VES schemes exist, there is also an unforgeable VES scheme that is not extractable.*

Proposition 2. *If unforgeable and extractable VES schemes exist, there is also an unforgeable and extractable VES scheme that is not collusion-resistant.*

In addition, there is a new relation that can be quite useful when proving a VES secure. It states that a VES is unforgeable if it is collusion-resistant. To see this, observe that giving ask to $\mathcal{A}$ in the collusion-resistance experiment enables the adversary to simulate the oracles $\mathfrak{S}$ and $\mathfrak{A}$ itself. Thus, if $\mathcal{A}$ is successful in the unforgeability experiment it can also break collusion-resistance.

Proposition 3. *If VES is collusion-resistant, it is also unforgeable.*

2.2 Discussion

As already discussed in the introduction, giving the key generation algorithm access to the adjudicator corresponds to the natural case where we have an initial setup phase. Note that the oracle $\mathfrak{S}$ only takes as input the public key and *not* the private key. Thus, this phase cannot be compared with the models that require the signer to prove knowledge of the secret key (e.g., KOSK). Moreover, this phase only takes place once, during key generation and *not* during each signature creation process. The adjudicator remains offline, i.e., our modification is suitable for fair exchange protocols with a *passive* adjudicator. Via $\mathsf{AdjSetup}$, the adjudicator may define parts of signer keys. Giving the adjudicator too much control over, however, is discouraged as it affects collusion-resistance.

In [22], the authors show that "abuse-freeness" is already implicit as long as the underlying signature scheme is unforgeable and treated as a black box (key independence, Definition 2 below). Since we consider a slightly stronger definition, we have to re-prove this result for collusion-resistance (Lemma 2).

Note that our constructions satisfy these properties. Thus, it is sufficient to prove opacity and extractability because key-independent and extractable schemes are automatically unforgeable (Lemma 1).

3 Generic Construction

We propose two efficient generic constructions based on CPA-secure (even slightly weaker) encryption, "maskable" digital signature schemes, and collision resistant hash functions. As our instantiations in Section 4 and in the full version demonstrate, this does not overly restrict the possible choices. In order to ensure that verifiably encrypted signatures can always be decrypted (cf. extractability), we build a Merkle authentication tree from a collision-resistant hash function. While our first construction uses regular signature schemes, our second construction reduces the assumptions even further by merely relying on one-time signatures.

3.1 Building Blocks

Let $\mathsf{G} : \{0,1\}^* \to \{0,1\}^n$ be a collision resistant hash function and let $\{x_i\}_1^\ell$ be the ordered set of values $x_1, \ldots, x_\ell$. Furthermore, we use $x \leftarrow_\$ S$, when choosing an $x \in S$ uniformly at random and $x \overset{\Delta}{\leftarrow} S$ if x is chosen according to a distribution Δ over S. For our generic constructions, we need to define "maskable" signature schemes, secure encryption schemes, and Merkle authentication trees.

Digital Signature Scheme. The definition of digital signature schemes $\mathsf{DSig} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ follows the well-known and established definition due to [12].

For our generic construction, we need a signature scheme that is "maskable". Generally speaking, signatures can be hidden by a masking value, such that we can still verify them. Furthermore, we have to be able to recover a valid signature from a valid masked one. We formalize this in the following definition.

Definition 1 (Maskability). *Let* DSig *be a signature scheme with public-key space* $\mathcal{K}$*, signature space* Σ*, and message space* $\mathcal{M}$*. It is* maskable *if there is a corresponding masking scheme* $\mathsf{MS}_{\mathsf{DSig}} = (\mathsf{Advice}, \mathsf{Mask}, \mathsf{Unmask}, \mathsf{Vf})$ *with the following specification:*

Sets: Let $\mathcal{S}$ be a set of masking values and let Δ be a distribution over $\mathcal{S}$. Furthermore, let $\mathcal{V}$ be the set of advice strings for verifying masked signatures and $\mathcal{T}$ be the space of masked signatures.

Advice: On input spk, α, Advice outputs an advice $\beta \in \mathcal{V}$.

Mask: On input spk, σ, α, M, the algorithm Mask outputs a masked signature $\tau \in \mathcal{T}$. Notice that we do not require a perfect masking scheme but allow the scheme to output the special symbol $\tau = \bot$.

Unmask: On input τ, β, α, M, the algorithm Unmask outputs a signature $\sigma \in \Sigma$.

Verification: On input spk, τ, β, and M the algorithm $\mathsf{MS}_{\mathsf{DSig}}.\mathsf{Vf}$ outputs a bit, indicating the validity of the masked signature. If $\tau = \bot$, it returns 0.

```
**Experiment** $\mathsf{Recover}_{\mathcal{A}}^{\mathsf{MS}_{\mathsf{DSig}}}(n)$
    $(\mathsf{ssk}, \mathsf{spk}) \leftarrow \mathsf{DSig.Kg}(1^n)$
    $\alpha \overset{\Delta}{\leftarrow} \mathcal{S}$
    $\beta \leftarrow \mathsf{Advice}(\mathsf{spk}, \alpha)$
    $\sigma^* \leftarrow \mathcal{A}^{\mathfrak{M}(\mathsf{ssk},\mathsf{spk},\alpha,\cdot),\mathsf{DSig.Sign}(\mathsf{ssk},\cdot)}(\mathsf{spk}, \beta)$
    Let $\{M_i\}_1^\ell$ be the queries to $\mathsf{DSig.Sign}$ and let $M^*$ be the query to $\mathfrak{M}$.
    Return 1 iff $M^* \notin \{M_i\}_1^\ell$ and $\mathsf{DSig.Vf}(\mathsf{spk}, \sigma^*, M^*) = 1$.
```

Fig. 2. Experiment for the hiding property of a masking scheme

Validity: We require $\mathsf{MS}_{\mathsf{DSig}}.\mathsf{Vf}(\mathsf{spk}, \tau, \beta, M) = 1 \Longrightarrow \mathsf{DSig.Vf}(\mathsf{spk}, \sigma', M) = 1$, where $\sigma' = \mathsf{Unmask}(\tau, \beta, \alpha, M)$ and $\beta = \mathsf{Advice}(\mathsf{spk}, \alpha)$, for all keys, masked signatures, messages, and masking values. Note that β is honestly created.

Hiding: The masking scheme needs to hide the signature σ (for M) in (τ, β) such that no adversary can recover a valid signature for M without knowing the masking value α. This must even hold if the adversary can query an oracle $\mathfrak{M}$ *once* that returns a masked signature for an adversely chosen message and a randomly chosen α: $\mathfrak{M}(\mathsf{ssk}, \mathsf{spk}, \alpha, M) = [\sigma \leftarrow \mathsf{DSig.Sign}(\mathsf{ssk}, M); \tau \leftarrow \mathsf{Mask}(\mathsf{spk}, \sigma, \alpha, M); \text{Return } \sigma;]$. Furthermore, the adversary can make arbitrary queries to an ordinary signature oracle. MS is (t, ϵ)-*hiding* if there is no adversary, running in time t, that wins the Experiment in Figure 2 with probability at least ϵ.

Notice that the above definition can be trivially satisfied by an encryption scheme. Then, the output of Advice is a zero-knowledge proof, demonstrating that it was honestly encrypted. Here, the masking value α would comprise the public and secret keys for the encryption scheme.

As we are interested in efficient instantiations, we propose that somewhat homomorphic signature schemes can provide the same functionality. We demonstrate this with the following example.

Example 1. Take the RSA signature scheme with full-domain hash function H and public key (N, v). The verification function for a signature σ on a message M checks whether $0 \leq \sigma < N$ and $\sigma^v = \mathsf{H}(M)$ over $\mathbb{Z}_N$. We let $\Sigma = \mathcal{V} = \mathbb{Z}_N$ and $\mathcal{S} = \mathbb{Z}_N^*$. Δ is the uniform distribution. $\mathsf{Mask}((N, v), \sigma, \alpha, M)$ outputs $\sigma\alpha \bmod N$ and $\mathsf{Advice}((N, v), \alpha)$ returns $\alpha^v \bmod N$. Thus, $\mathsf{Unmask}(\tau, \beta, \alpha, M)$ has to compute $\sigma \leftarrow \tau\alpha^{-1} \bmod N$. The modified verification algorithm $\mathsf{MS.Vf}(\mathsf{spk}, \tau, \beta, M)$ checks whether $0 \leq \tau < N$ and $\tau^v = \mathsf{H}(M)\beta$. Observe that validity and the hiding property are satisfied in the random oracle model.

Remark 1. Given the above example, it is easy to see that one can forge a masked signature (τ, β) that passes $\mathsf{MS.Vf}$, unless α and $\beta = \mathsf{Advice}(\mathsf{pk}, \alpha)$ are well-formed. One could simply compute $\beta \leftarrow \tau^v / \mathsf{H}(M)$ over $\mathbb{Z}_N$ for arbitrary M and τ. The result (τ, β, M) would be valid because $\tau^v \equiv \mathsf{H}(M)\beta$. However, in our constructions, the attacker is not able to choose β freely. It is chosen during a trusted setup procedure and then authenticated with a hash tree. This authentication mechanism yields an implicit rejection of adversely chosen β.

Notice that $\mathsf{AdjSetup}$ and the setup oracle $\mathfrak{S}$ are always controlled by the experiments in Figure 1 to make the setup procedure trusted. A straightforward extension of our security model would be to remove this trusted setup procedure.

Random Plaintext Attacks (RPA). Let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme, defined via *Key Generation:* $\mathsf{Kg}(1^n)$ outputs a key pair $(\mathsf{esk}, \mathsf{epk})$. epk is the public encryption key and esk is the secret decryption key; *Encryption:* $\mathsf{Enc}(\mathsf{epk}, M)$ outputs a ciphertext c for M; *Decryption:* $\mathsf{Dec}(\mathsf{esk}, c)$ attempts to decrypt c and returns the enclosed message, or $\perp$ upon failure. We define a weaker notion of security for encryption schemes that we call *random plaintext attacks* that is similar to a key encapsulation mechanism (KEM). The idea is that the adversary obtains a randomly chosen string s and a ciphertext c. The task is to determine whether c encrypts s or the 0-string.

We say that PKE is indistinguishable under random plaintext attacks (IND-RPA) if no efficient algorithm $\mathcal{A}$ can associate a randomly generated plaintext with its ciphertext. The adversary wins if it is able to guess b with probability $> 1/2$. PKE is *RPA secure* if for any efficient $\mathcal{A}$ and a negligible $\epsilon = \epsilon(n)$

$$\left|\mathrm{Prob}\left[b = \mathcal{A}(\mathsf{epk}, C) : (\mathsf{esk}, \mathsf{epk}) \leftarrow \mathsf{Kg}(1^n); b \leftarrow_{\$} \{0,1\}; M_0 \leftarrow_{\$} \mathcal{M}; M_1 \leftarrow 0^{|M_0|}; C \leftarrow_{\$} \mathsf{Enc}(\mathsf{epk}, M_b)\right] - \tfrac{1}{2}\right| < \epsilon\,.$$

CPA $\implies$ RPA. We claim that the notion of random plaintext attacks is strictly weaker than chosen plaintext attacks, in the sense that any CPA scheme is also RPA, but not vice-versa.

Proposition 4. *A CPA secure scheme is also RPA secure. If an RPA secure scheme exists then there is also an RPA secure scheme that is not CPA secure.*

The first part is obvious. As for the second part, the basic idea is letting $\mathsf{Enc}(1\|0^{n-1})$ in the modified RPA scheme output esk. Clearly, the scheme is not CPA secure but it is still RPA secure.

Merkle Authentication Trees. Merkle presented a tree structure that can be used to authenticate big amounts of data using only a single hash value [18]. Originally his idea was to create digital signatures out of one-time signature schemes, but many other applications of Merkle trees appeared in the past. With our constructions, we add verifiable encryption to this list of applications.

A Merkle tree is a complete binary tree of height h that is built from the bottom up to the root such that the leaves define the whole tree. The leaves are numbered consecutively from left to right. Inner nodes are constructed using the following rule: a node's value is the hash value of the concatenation of its children left and right: node $= \mathsf{G}(\mathsf{left}\|\mathsf{right})$, where $\mathsf{G} : \{0,1\}^* \to \{0,1\}^n$ is a collision resistant hash function. The root of the tree is used to authenticate the leaf values. For the authentication process, additional tree nodes are required.

These nodes form the *authentication path* of a leaf. Consider the path from the leaf with index $\varphi \in [1, 2^h]$ to the root. The siblings of the nodes on this path form the authentication path of this leaf. Using this path and the construction rule $\mathsf{G}(\mathsf{left}\|\mathsf{right})$, the root of the tree can be reconstructed. If the calculated root matches the original one, the leaf data is correctly authenticated.

Using an adversary that is able to replace a leaf value, such that the replaced leaf is still correctly authenticated by the tree, one can find collisions in the underlying hash function G. For an overview of techniques, results, and references, we refer the reader to [5, Chapter 3].

3.2 Generic Construction

The general idea is to use a maskable signature scheme with one-time masking values and encrypt these masking values under the adjudicator's public key. The *validity* property of the masking scheme ensures completeness, and opacity will be guaranteed by the *hiding* property. In general, we take an ordinary σ and hide it by applying Mask, using one of ℓ predefined one-time masking values α. If, for any reason, the masking scheme returns an invalid masked signature, the process is repeated with the next α. This allows for a broader range of (imperfect) masking schemes. The corresponding advice β for verification is also precomputed. Then, β and an encryption γ of α are used to build a Merkle authentication tree that allows a verifier to efficiently check whether β and γ correspond. The adjudicator forms the tree during the initial registration phase and signs its root under a certification key pair $(\mathsf{csk}, \mathsf{cpk})$ in order to prevent malicious signers from cheating in the extractablity experiment.

Construction 1. *Let* DSig *be a maskable signature scheme with masking scheme* $\mathsf{MS}_{\mathsf{DSig}}$*,* PKE *be a public key encryption scheme and* $\mathsf{G} : \{0,1\}^* \mapsto \{0,1\}^n$ *be a collision resistant hash function. Choose an adequate* $h \in \mathbb{N}$*, such that the resulting scheme admits* $\ell = 2^h$ *signatures.* $\mathsf{VES}_1 = (\mathsf{AdjKg}, \mathsf{AdjSetup}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$ *is defined as follows.*

Adjudicator Key Generation: *Call* $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{PKE.Kg}(1^n)$*,* $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \mathsf{DSig.Kg}(1^n)$ *and output* $((\mathsf{ask}, \mathsf{csk}), (\mathsf{apk}, \mathsf{cpk}))$.

Adjudication Setup: *On input* $((\mathsf{ask}, \mathsf{csk}), \mathsf{spk})$*, perform the following steps: 1. Choose* $\alpha_i \overset{\Delta}{\leftarrow} \mathcal{S}$ *and set* $\beta_i \leftarrow \mathsf{Advice}(\mathsf{spk}, \alpha_i)$*,* $\gamma_i \leftarrow \mathsf{Enc}(\mathsf{apk}, \alpha_i)$ *for* $i = 1, \ldots, \ell$*; 2. Construct a Merkle tree* T *using* G*, i.e., with leaves* $\mathsf{G}\Big(\mathsf{G}(\beta_i)||\mathsf{G}(\gamma_i)\Big)$ *that fully define the root* ρ*; 3. Compute the signature* $\sigma_\rho \leftarrow \mathsf{DSig.Sign}(\mathsf{csk}, \rho)$*; 4. Output* $(\{\alpha_i\}_1^\ell, \{\gamma_i\}_1^\ell, \rho, \sigma_\rho)$.

Key Generation: *Perform the following steps: 1. Call* $(\mathsf{ssk}, \mathsf{spk}) \leftarrow \mathsf{DSig.Kg}(1^n)$*; 2. Call* $(\{\alpha_i\}_1^\ell, \{\gamma_i\}_1^\ell, \rho, \sigma_\rho) \leftarrow \mathsf{AdjSetup}((\mathsf{ask}, \mathsf{csk}), \mathsf{spk})$*; 3. Initialize a signature counter* $c \leftarrow 0$*; 4. Output* $\mathsf{pk} = (\mathsf{spk}, \rho, \sigma_\rho)$ *and* $\mathsf{sk} = (\mathsf{ssk}, c, \{\alpha_i\}_1^\ell, \{\gamma_i\}_1^\ell)$.

Sign, Verify: *As defined in the underlying signature scheme* DSig.

Create: *On input* $\mathsf{sk}, \mathsf{pk}, M$*, the algorithm* Create *works in three steps: 1. Increment the counter c:* $c \leftarrow c + 1$*; 2. Sign the message* M *using the underlying signature scheme:* $\sigma \leftarrow \mathsf{DSig.Sign}(\mathsf{ssk}, M)$*; 3. Mask the signature* σ *with the secret* *value α_c: $\tau \leftarrow \mathsf{Mask}(\mathsf{spk}, \sigma, \alpha_c, M)$; 4. If $\mathsf{MS}_{\mathsf{DSig}}.\mathsf{Vf}(\mathsf{spk}, \tau, \beta_c, M) = 0$ increase c and go to 3. The output is $\varpi = (\tau, \beta_c, \gamma_c, \pi_c)$, where $\beta_c \leftarrow \mathsf{Advice}(\mathsf{spk}, \alpha_c)$ and π_c is the authentication path for leaf c.*

VES Verification: *On input $((\mathsf{apk}, \mathsf{cpk}), (\mathsf{spk}, \rho, \sigma_\rho), (\tau, \beta, \gamma, \pi), M)$, VesVf outputs 1 iff 1. $\mathsf{DSig}.\mathsf{Vf}(\mathsf{cpk}, \sigma_\rho, \rho) = 1$; 2. π is correct for β and γ with respect to ρ; 3. $\mathsf{MS}_{\mathsf{DSig}}.\mathsf{Vf}(\mathsf{spk}, \tau, \beta, M) = 1$.*

Adjudication: *On input $((\mathsf{ask}, \mathsf{csk}), (\mathsf{apk}, \mathsf{cpk}), (\mathsf{spk}, \rho, \sigma_\rho), (\tau, \beta, \gamma, \pi), M)$, Adj verifies the input using VesVf. If it is correct, it decrypts $\alpha' \leftarrow \mathsf{Dec}(\mathsf{ask}, \gamma)$, calls $\sigma' \leftarrow \mathsf{Unmask}(\tau, \beta, \alpha', M)$, and outputs σ'.*

3.3 Generic Construction Using One-Time Signatures

Since we already need a Merkle authentication tree for Construction 1, we can as well use a suitable one-time signature instead of a regular one. These signatures are potentially easier to achieve, i.e., they may be secure under milder assumptions. The following construction demonstrates that the second tree that is needed to turn a one-time signature scheme into a "many-time" signature scheme via the Merkle transformation can be easily merged with the first one.

Construction 2. *With OTS we denote a maskable one-time signature scheme with masking scheme $\mathsf{MS}_{\mathsf{OTS}}$. We define VES_2 as follows:*

Adjudicator Key Generation: *Call $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{PKE}.\mathsf{Kg}(1^n)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \mathsf{DSig}.\mathsf{Kg}(1^n)$ and output $((\mathsf{ask}, \mathsf{csk}), (\mathsf{apk}, \mathsf{cpk}))$.*

Adjudication Setup: *On input $((\mathsf{ask}, \mathsf{csk}), \{\mathsf{spk}_i\}_1^\ell)$, perform the following steps: 1. Choose $\alpha_i \overset{\Delta}{\leftarrow} \mathcal{S}$ and set $\beta_i \leftarrow \mathsf{Advice}(\mathsf{spk}, \alpha_i)$, $\gamma_i \leftarrow \mathsf{Enc}(\mathsf{apk}, \alpha_i)$ for $i = 1, \ldots, \ell$; 2. Construct a Merkle authentication tree T using the hash function G, where the leaves are of the form $\mathsf{G}\Big(\mathsf{G}(\beta_i)||\mathsf{G}(\gamma_i)||\mathsf{G}(\mathsf{spk}_i)\Big)$. Denote the root node with ρ; 3. Compute the signature $\sigma_\rho \leftarrow \mathsf{DSig}.\mathsf{Sign}(\mathsf{csk}, \rho)$; 4. Output $(\{\alpha_i\}_1^\ell, \{\gamma_i\}_1^\ell, \rho, \sigma_\rho)$.*

Key Generation: *Run the following steps: 1. Call $(\mathsf{ssk}_i, \mathsf{spk}_i) \leftarrow \mathsf{OTS}.\mathsf{Kg}(1^n)$ for $i = 1, \ldots \ell$; 2. Call $(\{\alpha_i\}_1^\ell, \{\gamma_i\}_1^\ell, \rho, \sigma_\rho) \leftarrow \mathsf{AdjSetup}((\mathsf{ask}, \mathsf{csk}), \{\mathsf{spk}_i\}_1^\ell)$; 3. Initialize a signature counter $c \leftarrow 0$; 4. Output $\mathsf{pk} = (\rho, \sigma_\rho)$ and $\mathsf{sk} = (\{\mathsf{ssk}_i\}_1^\ell, \{\mathsf{spk}_i\}_1^\ell, c, \{\alpha_i\}_1^\ell, \{\gamma_i\}_1^\ell)$.*

Sign, Verify: *As defined in OTS.*

Create: *On input $\{\mathsf{ssk}_i\}_1^\ell, \{\mathsf{spk}_i\}_1^\ell, c, \{\alpha_i\}_1^\ell, \{\gamma_i\}_1^\ell, M$, Create works in four steps: 1. Increment c: $c \leftarrow c + 1$; 2. Sign M: $\sigma \leftarrow \mathsf{OTS}.\mathsf{Sign}(\mathsf{ssk}_c, M)$; 3. Mask σ: $\tau \leftarrow \mathsf{Mask}(\mathsf{spk}_c, \sigma, \alpha_c, M)$; 4. If $\mathsf{MS}_{\mathsf{DSig}}.\mathsf{Vf}(\mathsf{spk}_c, \tau, \beta_c, M) = 0$ go to 1. The output is $\varpi = (\tau, \beta_c, \gamma_c, \pi_c, \mathsf{spk}_c)$, where $\beta_c \leftarrow \mathsf{Advice}(\mathsf{spk}, \alpha_c)$ and π_c is the authentication path for leaf c.*

VES Verification: *On input $((\mathsf{apk}, \mathsf{cpk}), (\rho, \sigma_\rho), (\tau, \beta, \gamma, \pi, \mathsf{spk}), M)$, VesVf outputs 1 iff 1. $\mathsf{DSig}.\mathsf{Vf}(\mathsf{cpk}, \sigma_\rho, \rho) = 1$; 2. ρ can be reconstructed using π, β, γ, and spk; 3. $\mathsf{MS}_{\mathsf{DSig}}.\mathsf{Vf}(\mathsf{spk}, \tau, \beta, M) = 1$.*

Adjudication: *On input $((\mathsf{ask}, \mathsf{csk}), (\mathsf{apk}, \mathsf{cpk}), (\rho, \sigma_\rho), (\tau, \beta, \gamma, \pi, \mathsf{spk}), M)$, Adj verifies the input using VesVf. If it is correct, it decrypts $\alpha' \leftarrow \mathsf{Dec}(\mathsf{ask}, \gamma)$, calls $\sigma' \leftarrow \mathsf{Unmask}(\tau, \beta, \alpha', M)$, and outputs σ'.*

3.4 Proof of Security

We show in this section that VES_1 satisfies the desired security requirements. Security of Construction 2 is proven analogously, the assumptions are just weaker, i.e., we only need a maskable one-time signature scheme instead of a regular one. We show extractability, unforgeability, collusion-resistance, and opacity. The proofs for VES_1 and VES_2 are essentially the same, we focus on VES_1.

Theorem 1 (Extractability). *If* G *is collision resistant,* DSig *is unforgeable, and* $\mathsf{MS}_{\mathsf{DSig}}$ *satisfies validity then* VES_1 *(*VES_2*) is extractable.*

Proof. The reduction plays against unforgeability of DSig and uses the validity of $\mathsf{MS}_{\mathsf{DSig}}$ and the collision-resistance of G and in the analysis. The unforgeability ensures that the adversary has to call $\mathsf{AdjSetup}$ to create its public key and validity guarantees that an extracted signatures is valid if computed from an honestly masked signature. Most importantly, the collision-resistance of G prevents the adversary from altering the leaves of the authentication tree, i.e., from being able to dishonestly mask a signature.

The reduction chooses the adjudication key honestly during the simulation and has access to a signature oracle for DSig and to the verification key spk. Thus, the adversary's environment can be perfectly simulated. The adversarial user $\mathcal{A}$ outputs a public key $(\mathsf{pk}^*, \rho^*, \sigma_\rho^*)$ and a pair $(M^*, (\tau^*, \alpha^*, \gamma^*, \pi^*))$ for which VesVf outputs 1. Furthermore, we let σ' be the result of the adjudication algorithm for $(\tau^*, \beta^*, \gamma^*, \pi^*)$.

Towards contradiction, assume that extraction fails, i.e., $\mathsf{DSig.Vf}(\mathsf{spk}, \sigma', M^*) = 0$. From VesVf, we know that ρ^* was previously created by the simulator together with a signature σ_ρ^*, using the external signature oracle. Otherwise, we would have an existential forgery that refutes unforgeability of DSig. Assume that ρ^* was formed using $\{\alpha_i\}_1^\ell, \{\beta_i\}_1^\ell, \{\gamma_i\}_1^\ell$.

VesVf guarantees that π^* is an authentication path for the leaf $\mathsf{G}(\mathsf{G}(\beta^*)||\mathsf{G}(\gamma^*))$ w.r.t. ρ. Thus, there is an index $i \in \{1, \ldots, \ell\}$ such that $\beta^* = \beta_i = \mathsf{Advice}(\mathsf{spk}, \alpha_i)$ and $\gamma^* = \gamma_i$. Otherwise, we would have at least one collision in the hash tree, which refutes collision resistance of G.

Finally, VesVf ensures that $\mathsf{MS}_{\mathsf{DSig}}.\mathsf{Vf}(\mathsf{spk}, \tau^*, \beta^*, M^*) = 1$, which implies the contradiction $\mathsf{DSig.Vf}(\mathsf{spk}, \sigma', M^*) = 1$ because of the *validity* of $\mathsf{MS}_{\mathsf{DSig}}$. □

In order to prove unforgeability, we need to observe that both constructions apply signature and encryption keys separately because $\mathsf{DSig.Sign}$ is called as a black box and the result is encrypted, or masked in our context. More precisely, they satisfy the following definition of key-independence.

Definition 2 (Key-Independence [22]). *Let the signer's private key* sk *consist of two independent elements* $(\mathsf{kisk}, \mathsf{ssk})$ *and let* $\mathsf{pk} = (\mathsf{kipk}, \mathsf{spk})$ *be the corresponding public key.* VES *is* key-independent *if there is an efficient encryption algorithm* $\mathsf{KI\text{-}Enc}$ *such that* $\mathsf{KI\text{-}Enc}(\mathsf{apk}, \mathsf{kipk}, \mathsf{kisk}, \mathsf{DSig.Sign}(\mathsf{ssk}, M), M) \equiv \mathsf{VES.Create}(\mathsf{sk}, \mathsf{apk}, M)$ *for all* $M \in \mathcal{M}$*, where* DSig *is employed signature.*

Lemma 1 ([22]). *Let* VES *be extractable and key-independent.* VES *is unforgeable if the underlying signature scheme* DSig *is unforgeable.*

Regarding our novel security requirement against collusion of the adjudicator and a user, we prove the following useful lemma.

Lemma 2. *Let* VES *be extractable and key-independent.* VES *is collusion-resistant if the underlying signature scheme* DSig *is unforgeable.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ that successfully breaks collusion-resistance with non-negligible probability $\epsilon(n)$ after at most q queries to the oracle $\mathfrak{C}$. We show how to forge ordinary signatures in DSig, running $\mathcal{A}$ as a black-box, with q queries to the signature oracle. The reduction $\mathcal{B}$, playing against unforgeability of DSig, receives a public verification key spk and has access to a signing oracle $\mathsf{DSig.Sign}(\mathsf{ssk}, \cdot)$. It generates an adjudication key pair $(\mathsf{ask}, \mathsf{apk}) \leftarrow \mathsf{VES.AdjKg}(1^n)$ and runs the remaining part of $\mathsf{VES.Kg}$, including $\mathsf{AdjSetup}$, to obtain a VES key pair $(\mathsf{sk}, \mathsf{pk})$. This is possible because VES is key-independent. Afterwards, $\mathcal{B}$ sets $\mathsf{state} \leftarrow (\mathsf{ask}, \mathsf{apk}, \mathsf{pk})$ and runs $\mathcal{A}(\mathsf{state})$ as a black-box. Whenever $\mathcal{A}$ queries M to $\mathfrak{C}$, $\mathcal{B}$ calls its external signing oracle $\sigma \leftarrow \mathsf{Sign}(\mathsf{ssk}, M)$ and computes $\varpi \leftarrow \mathsf{KI\text{-}Enc}(\mathsf{apk}, \mathsf{kipk}, \mathsf{kisk}, \sigma, M)$. Finally, $\mathcal{A}$ stops and outputs (M^*, ϖ^*). $\mathcal{B}$ extracts the corresponding signature $\sigma^* \leftarrow \mathsf{VES.Adj}(\mathsf{ask}, \mathsf{apk}, \mathsf{pk}, \varpi^*, M^*)$ and returns (M^*, σ^*). Observe that the environment of $\mathcal{A}$ is perfectly simulated and all oracle queries are simulated efficiently. By definition, $\mathcal{A}$ has not queried M^* to $\mathfrak{C}$. Thus, $\mathcal{B}$ has not queried M^* to its signature oracle. Moreover, the resulting (M^*, ϖ^*) yields an ordinary message-signature pair (M^*, σ^*) because VES is extractable. As a consequence, $\mathcal{B}$'s attack is legitimate and it succeeds with probability $\epsilon(n)$ after q queries to the signature oracle. □

Since VES_1 and VES_2 are extractable and key-independent, unforgeability follows from Lemma 1, and Lemma 2 guarantees collusion-resistance.

Corollary 1 (Unforgeability). *If* DSig *is unforgeable and* VES_1 *(*VES_2*) is extractable and key-independent, then* VES_1 *(*VES_2*) is unforgeable.*

Corollary 2 (Collusion-resistance). *If* DSig *is unforgeable and* VES_1 *(*VES_2*) is extractable and key-independent, then* VES_1 *(*VES_2*) is collusion-resistant.*

Concerning Opacity, we show the following:

Theorem 2 (Opacity). *If* DSig *is unforgeable,* PKE *is RPA secure,* $\mathsf{MS}_{\mathsf{DSig}}$ *is hiding, and* G *is collision resistant then* VES_1 *(*VES_2*) is opaque.*

Proof. An adversary breaking opacity can succeed in two different ways. First, by forging the underlying signature scheme, and second, by decrypting a given verifiably encrypted signature. We say that an algorithm $\mathcal{A}$ is a

1. type-1 adversary ($\mathcal{A}_1$), if it outputs a message-signature pair (M^*, σ^*) such that it *has never queried* M^* to $\mathfrak{C}$, or if it *invokes* $\mathfrak{A}$ on M' without having queried M' to $\mathfrak{C}$ before.
2. type-2 adversary ($\mathcal{A}_2$), if it outputs a message-signature pair (M^*, σ^*) such that it *has queried* M^* to $\mathfrak{C}$ and it *has never invoked* $\mathfrak{A}$ on M' without having queried M' to $\mathfrak{C}$ before.

$\mathcal{A}_1$ can be directly used to forge signatures in DSig. The reduction has control over the adjudicator's private key and can therefore extract ordinary signatures (forgeries) from $\mathcal{A}_1$'s output. We refer the reader to the full version.

Type-2 Attacker. We perform a change to the simulation of $\mathcal{A}_2$'s environment and argue that each does not change $\mathcal{A}_2$'s success probability but for a negligible amount. Let ϵ be $\mathcal{A}_2$'s success probability in the (unmodified) opacity experiment. We change the algorithm $\mathsf{AdjSetup}$.

Adjudication Setup: The algorithm $\mathsf{AdjSetup}'$ selects the elements α_i, β_i as before and chooses a random index $c^* \leftarrow_\$ \{0, \ldots, \ell\}$. It computes all $\gamma_{i \neq c^*}$ as before but $\gamma_{c^*} \leftarrow \mathsf{Enc}(\mathsf{apk}, 0^n)$. It outputs the corresponding tree, root ρ, and signature σ_ρ as before.

Due to the RPA security of the encryption scheme, this only changes $\mathcal{A}_2$'s success probability by a negligible ϵ'. The next change to $\mathsf{AdjSetup}$ allows us the reduction to use $\mathcal{A}_2$ to refute the hiding property of $\mathsf{MS}_{\mathsf{DSig}}$.

Adjudication Setup: The algorithm $\mathsf{AdjSetup}''$ works like $\mathsf{AdjSetup}'$, but receives β_{c^*} from $\mathsf{Recover}$ and embeds it into the the leaf at index c^*.

The success probability of $\mathcal{A}_2$ does not change because β_{c^*} is distributed as before. Also, knowledge of α_{c^*} is not necessary to build the modified public key.

The remaining oracles, $\mathfrak{C}$ and $\mathfrak{A}$, are perfectly simulated for all indices $\neq c^*$ because the reduction has access to all masking values (except α_{c^*}) and can therefore answer all adjudication queries. In particular, this is the reason why we do not require some form of CCA secure encryption: all plaintexts are known and authenticated. Also, using these masking values together with the signature oracle in the $\mathsf{Recover}$ experiment, enables the reduction to answer queries to $\mathfrak{C}$.

Eventually, $\mathcal{A}_2$ outputs a message-signature pair (M^*, σ^*). If it is valid for the index c^*, the reduction outputs σ^* to refute the hiding property. Otherwise, it aborts. The reduction's success probability is noticeable if ϵ is noticeable. □

4 Efficient Instantiations

We show that the assumptions in our generic constructions are sound and that *maskability* does not overly restrict the choice of signature schemes. We aim at providing VES schemes based on a broad range of cryptographic assumptions, including post-quantum ones. Here, we present the first efficient *pairing-free* VES in the *standard model*. The full version contains a instantiations of Construction 1 from lattices and RSA in the random oracle model.

4.1 An Instantiation Based on Worst-Case Lattice Problems in Ideal Lattices (Construction 2)

We propose an instantiation based on the hardness of lattice problems for Construction 2 in the *standard model*. The impact of this instantiation is significant. Not only are lattice-based constructions immune to quantum computer attacks, but they are also desirable in the classic scenario. Computations in lattices are efficient (mostly basic linear algebra) and cryptographic hardness can be based on worst-case assumptions by Ajtai's worst-case to average-case reduction [1].

Lattices. A full-rank lattice in $\mathbb{R}^m$ is a set $\Lambda = \{\sum_{i=1}^{m} x_i \, \mathbf{b}_i \mid x_i \in \mathbb{Z}\}$, where $\mathbf{b}_1, \ldots, \mathbf{b}_m$ are linearly independent over $\mathbb{R}$. The matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_m]$ is a *basis* of the lattice Λ and we write $\Lambda = \Lambda(\mathbf{B})$. The number of linearly independent vectors in the basis is the dimension of the lattice. *Ideal lattices* are a special class of lattices. Given a monic, irreducible (over $\mathbb{Z}$) polynomial f of degree n, the ring $R = \mathbb{Z}_p[x]/\langle f \rangle \cong \mathbb{Z}_p^n$, and an ideal I of R, the ideal lattice Λ is the set of coefficient vectors of the polynomials in I. In other words, $\Lambda = \{\mathbf{a} \in \mathbb{Z}^n : \sum_{i=0}^{n-1} a_i x^i \in I\}$. The main computational problem in lattices is the approximate shortest vector problem (SVP^∞), where an algorithm is given a basis of a lattice Λ and is supposed to find a sufficiently short vector $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$ with respect to the ℓ_∞ norm. The SVP^∞ in ideal lattices is called ISVP^∞. We write $\mathsf{ISVP}^\infty(f, \nu)$ for the problem of finding a vector $\mathbf{v}$ with $\|\mathbf{v}\|_\infty \leq \nu\lambda_1$ (λ_1 is the minimum distance in the lattice) in *all* lattices corresponding to ideals in the ring R.

$\mathcal{H}_{R,m}$ is the family of compression functions that map elements from R^m to R. Functions $\mathsf{h} \in \mathcal{H}_{R,m}$ are module homomorphisms, especially $\mathsf{h}(a + b) = \mathsf{h}(a) + \mathsf{h}(b)$. The problem $Col(\mathsf{h}, d)$ asks to find two distinct $\mathbf{x}$ and $\mathbf{x}'$ with $\max\{\|\mathbf{x}\|_\infty, \|\mathbf{x}'\|_\infty\} \leq d$ such that $\mathsf{h}(\mathbf{x}) = \mathsf{h}(\mathbf{x}')$. A polynomial time algorithm that solves $Col(\mathsf{h}, d)$ for $d = 10\phi p^{1/m} n \log^2(n)$ can be used to solve $\mathsf{ISVP}^\infty(f, \nu)$, where $\nu = \tilde{O}(\phi^5 n^2)$. Here, ϕ is a small ring constant defined in [15].

LM-OTS signatures. We use the one-time signature scheme of Lyubashevsky and Micciancio [15]. Its security is based on the problem of finding short vectors in ideal lattices, which is conjectured to be intractable by quantum computers.

Let $p = \tilde{\Theta}((\phi n)^4)$, $m = \lceil \log n \rceil$. Define the constant $D = 10\phi p^{1/m} n \log^2(n)$. The LM-OTS scheme is a tuple $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$, defined via *Key Generation:* $\mathsf{Kg}(1^n)$ outputs a signing key $(\mathbf{k}, \mathbf{l}) \in R^m \times R^m$ with $\|\mathbf{k}\|_\infty$ and $\|\mathbf{l}\|_\infty$ bounded. The verification key is $(\mathsf{h}, \mathsf{h}(\mathbf{k}), \mathsf{h}(\mathbf{l}))$, where $\mathsf{h} \leftarrow_\$ \mathcal{H}_{R,m}$; *Signing:* $\mathsf{Sign}((\mathbf{k}, \mathbf{l}), M)$, $M \in R$, $\|M\|_\infty \leq 1$, returns $\sigma = \mathbf{k}M + \mathbf{l} \in R_D^m$, where R_D^m restricts the coefficients in R to the range $[-D, D]$; *Verification:* $\mathsf{Vf}((\mathsf{h}, \mathsf{h}(\mathbf{k}), \mathsf{h}(\mathbf{l})), \sigma, M)$ returns 1 iff $\sigma \in R^m_{2\psi mnD - D}$ and $\mathsf{h}(\sigma) = \mathsf{h}(\mathbf{k})M + \mathsf{h}(\mathbf{l})$, for some constant ψ. The above scheme is a slight modification compared to [15] with regard to the admissible signature length. Observe that honestly generated signatures are in R_D^m, whereas signatures in $R^m_{2\psi mnD - D}$ may still be valid. The scheme remains secure under the stronger assumption that $Col(\mathsf{h}, 2\psi mnD - D)$, instead of $Col(\mathsf{h}, D)$, is hard. This change is required for the masking scheme to be hiding. If there is a successful adversary against unforgeability of this modified signature scheme, then one can find a collision $(\mathbf{x}, \mathbf{x}')$ under h such that $\max\{\|\mathbf{x}\|_\infty, \|\mathbf{x}'\|_\infty\} \leq D$. This can be used to solve ISVP^∞ in the worst case.

Instantiation. Let $\psi \in \mathbb{N}_{>0}$ be a small constant. The following table summarizes the instantiation using Construction 2 and the masking scheme $\mathsf{MS}_{LM\text{-}OTS}$.

pk	Σ	$\mathcal{T}$	$\mathcal{S}$	$\mathcal{V}$	Advice	Mask	Mask.Vf	Unmask
$(\mathsf{h}, \mathsf{h}(\mathbf{k}), \mathsf{h}(\mathbf{l}))$	$R^m_{2\psi mnD-D}$	$R^m_{\psi mnD-D}$	$R^m_{\psi mnD}$	R	$\mathsf{h}(\alpha)$	$\sigma + \alpha$	$\mathsf{h}(\tau) \equiv \mathsf{h}(\mathbf{k})M + \mathsf{h}(\mathbf{l}) + \beta,\ \tau \in \mathcal{T}$	$\tau - \alpha$

The masking distribution Δ is component-wise uniform. Signatures σ are masked via $\tau \leftarrow \sigma + \alpha$. Therefore, the verification function is easy to adapt because h is a module homomorphism. Notice that $\tau \leftarrow \sigma + \alpha$ may lie outside $R^m_{2\psi mnD-D}$. In this case, Mask returns the special symbol $\perp$ and Mask.Vf fails. Fortunately, we have $\sigma + \alpha \in R^m_{2\psi mnD-D}$ with probability $\approx e^{-1/\psi}$ (a generalization of Lemma 5.1 in [14]). There are two ways to deal with this completeness error. The first is to prepare $\omega(\log(n))$ many masking values for each leaf of the tree for a negligible error. However, this would waste time and space as a negligible completeness error is not necessary. By a simple trial-and-error approach, we may to discard some of the α's in VES.Create and move on to the next leaf. In practice, the number of failures is small. Choosing $\psi = 2$, for example, yields a success probability > 0.6 and it can be brought arbitrarily close to 1 by allowing larger ψ. Thus, ψ allows a tradeoff between completeness and size/security.

The following propositions show that $\mathsf{MS}_{LM\text{-}OTS}$ is applicable.

Proposition 5. $\mathsf{MS}_{LM\text{-}OTS}$ *supports validity.*

Proof. Let τ be a masked signature for M with advice $\beta = \mathsf{h}(\alpha)$. If τ is valid under Mask.Vf, then $\mathsf{h}(\tau) = \mathsf{h}(\mathbf{k})M + \mathsf{h}(\mathbf{l}) + \beta$. But if this equation holds, then it implies that $\mathsf{h}(\tau - \alpha) = \mathsf{h}(\mathbf{k})M + \mathsf{h}(\mathbf{l})$. Observe that $\tau - \alpha \in R^m_{2\psi mnD-D}$ because $\|\tau - \alpha\|_\infty \leq \|\tau\|_\infty + \|\alpha\|_\infty \leq 2\psi mnD - D$. Thus, we obtain a signature $\sigma = \tau - \alpha$ that passes $LM\text{-}OTS$.Vf for M as required. □

Proposition 6. $\mathsf{MS}_{LM\text{-}OTS}$ *is hiding if $Col(\mathsf{h}, 2\psi mnD - D)$ is hard.*

Proof. The proof is done in two steps. First, we show that two alternative LM-OTS signatures $\sigma \neq \sigma'$ for a given message M, which always exist because $|R^m_D| \gg |R|$ makes h compressing, are indistinguishable when masked according to the above rejection procedure. Second, we show that a successful attacker that wins in the Recover experiment can be used to find a collision under h.

The first part is showing that the statistical distance $\mathsf{SD}(\sigma + \alpha, \sigma' + \alpha)$, conditioned on $\sigma + \alpha, \sigma' + \alpha \in \mathcal{T}$, is zero over the choice of α. Notice that $\mathsf{SD}(\sigma+\alpha, \sigma'+\alpha) = 1/2 \sum_{t\in\mathcal{T}} |\operatorname{Prob}[\alpha = t - \sigma] - \operatorname{Prob}[\alpha = t - \sigma']| = 1/2 \sum_{t\in\mathcal{T}} |\prod_{i=1}^{mn} \operatorname{Prob}[\alpha_i = t_i - \sigma_i] - \prod_{i=1}^{mn} \operatorname{Prob}[\alpha_i = t_i - \sigma'_i]|$. The index i specifies a coefficient of the vectors in $R^m \cong \mathbb{Z}_q^{mn}$. Now, $\operatorname{Prob}[\alpha_i = t_i - \sigma_i] = 1/|R^m_{\psi mnD}|$ because $t_i - \sigma_i \in \mathcal{S}$ and $\alpha \leftarrow_\$ \mathcal{S}$. Thus, the distance is zero. The second part is a reduction from the collision problem. It chooses its own signature key with public key $(\mathsf{h}(\mathbf{k}), \mathsf{h}(\mathbf{l}))$ to simulate the oracle $\mathfrak{M}$, not the additional signature oracle because LM-OTS is one-time. On input M^*, it masks a signature $\sigma \in R^m_D$. The adversary's output will be $\sigma^* \in R^m_{2\psi mnD-D}$ with $\mathsf{h}(\sigma^*) = \mathsf{h}(\mathbf{k})M^* + \mathsf{h}(\mathbf{l}) = \mathsf{h}(\sigma)$. Since two alternative signatures are indistinguishable when masked, we have $\sigma^* \neq \sigma$ with probability 1/2 and we obtain the desired collision. □

Encryption. One could use ring-LWE [17] for encryption and furthermore use SWIFFT [16] for collision resistant hashing. This would base security entirely on worst-case ideal lattice problems.

5 Conclusions

With our work, we have extended the model of Boneh et al. by allowing an initial setup phase that is common in real-world scenarios. Moreover, we have proposed two novel generic constructions for verifiably encrypted signatures. Both rely on a certain class of signature schemes, a weaker-than-CPA secure encryption scheme, and a collision-resistant hash function. Both work without NIZKs or random oracles. To demonstrate their feasibility, we have instantiated them with a range of primitives, including post-quantum ones.

Acknowledgments

We thank the reviewers of ACNS 2010 for valuable comments. In particular, we are indebted to Willy Susilo.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC, pp. 99–108 (1996)
2. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. IEEE Journal on Selected Areas in Communications 18(4), 593–610 (2000)
3. Ateniese, G.: Verifiable encryption of digital signatures and applications. ACM Trans. Inf. Syst. Secur. 7(1), 1–20 (2004)
4. Bao, F., Deng, R.H., Mao, W.: Efficient and practical fair exchange protocols with off-line ttp. In: IEEE Symposium on Security and Privacy, pp. 77–85. IEEE Computer Society, Los Alamitos (1998)
5. Bernstein, D.J., Buchmann, J.A., Dahmen, E. (eds.): Post-Quantum Cryptography. Springer, Heidelberg (2008)
6. Biham, E. (ed.): EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
7. Boneh, D.: A brief look at pairings based cryptography. In: FOCS, pp. 19–26. IEEE Computer Society, Los Alamitos (2007)
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham [6], pp. 416–432
9. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
10. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
11. Dodis, Y., Lee, P.J., Yum, D.H.: Optimistic fair exchange in a multi-user setting. In: Okamoto, Wang [19], pp. 118–133
12. 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)
13. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
14. Lyubashevsky, V.: Towards Practical Lattice-Based Cryptography. PhD thesis (2008)

15. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)
16. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: Swifft: A modest proposal for fft hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008)
17. Lyubashevsky, V., Regev, O., Peikert, C.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110. Springer, Heidelberg (2010)
18. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
19. Okamoto, T., Wang, X. (eds.): PKC 2007. LNCS, vol. 4450. Springer, Heidelberg (2007)
20. Rückert, M.: Verifiably encrypted signatures from RSA without NIZKs. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 363–377. Springer, Heidelberg (2009)
21. Rückert, M., Schneider, M., Schröder, D.: Generic constructions for verifiably encrypted signatures without random oracles or NIZKs. Cryptology ePrint Archive, Report 2010/200 (2010), http://eprint.iacr.org/
22. Rückert, M., Schröder, D.: Security of verifiably encrypted signatures and a construction without random oracles. In: Shacham, H. (ed.) Pairing 2009. LNCS, vol. 5671, pp. 17–34. Springer, Heidelberg (2009)
23. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (1997)
24. Zhang, F., Safavi-Naini, R., Susilo, W.: Efficient verifiably encrypted signature and partially blind signature from bilinear pairings. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 191–204. Springer, Heidelberg (2003)

Redactable Signatures for Tree-Structured Data: Definitions and Constructions

Christina Brzuska, Heike Busch, Oezguer Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder

Technical University of Darmstadt
Center for Advanced Security Research Darmstadt (CASED)

Abstract. Kundu and Bertino (VLDB 2008) recently introduced the idea of structural signatures for trees which support public redaction of subtrees (by third-party distributors) while pertaining the integrity of the remaining parts. An example is given by signed XML documents of which parts should be sanitized before being published by a distributor not holding the signing key. Kundu and Bertino also provide a construction, but fall short of providing formal security definitions and proofs. Here we revisit their work and give rigorous security models for the redactable signatures for tree-structured data, relate the notions, and give a construction that can be proven secure under standard cryptographic assumptions.

1 Introduction

The XML data format is increasingly used to store and organize data. This development is most notable in the context of XML databases, which store the entire content in XML files. In some applications, both the integrity and the authenticity of the stored data must be ensured; this can in principle be achieved by signing the tree with a conventional cryptographic signature. In some scenarios, the content of the tree is privacy sensitive and an access control mechanism determines which part of the tree may be accessed by a specific user. The database management system must therefore be able to prune a tree upon access, so that those parts of the tree that the user is not allowed to see are removed prior to access. Still, it should be possible to prove the authenticity of the remaining data with respect to the original signer, without having to re-sign the document.

This can in principle be resolved by applying sanitizable signatures [2], which allow to overwrite certain parts of the data with a special null symbol while retaining the integrity of the data's signature. Such schemes also guarantee that the signature does not allow the recovery of deleted parts. Unfortunately, pure sanitization of data is insufficient to guarantee privacy: the recipient of the data clearly sees (due to the presence of the null symbol) that some data has been removed from the tree. This mere fact may already be an unwanted privacy leak.

Consider the following example taken from [6]. An XML file describes the health records of a single person. The root node's successors encode visits to a medical institution, whereas their successors encode results of medical tests performed at those institutions. Since nodes in XML files are ordered (and the tree structure may be publicly known), the recipient of the sanitized tree will be able to associate positions of null symbols in his tree with medical checks performed by the institution. For example, if a null symbol appears at a tree position associated with an HIV test, the recipient knows that such a test was performed, even without knowing its outcome. This information is already a privacy intrusion. Even worse breaches occur when several null symbols appear in positions associated with checkups for a certain disease: the recipient then quite accurately reconstructs the diagnosis.

To avoid such problems, it is therefore necessary to have a sanitizable signature that hides any performed santizations. Kundu and Bertino [6] proposed such a signature scheme for trees. However, they did not formally define the desired security properties; consequently, they were unable to formally prove the security of their scheme. Indeed, we show in this paper that their construction does not meet a strong security requirement.

Our Results. In this paper, we revisit the problem and give precise definitions for the privacy requirements. That is, besides the unforgeability of structural tree signatures, we also use game-based definitions to describe the notions of privacy (deleted data cannot be recovered) and transparency (recipients cannot even determine whether parts have been redacted). Our definitions are furthermore strong in the sense that they, for instance, allow the adversary to adaptively ask for multiple signatures for chosen tree structures.

Given our definitions of the desired security properties, we then formally relate these notions, showing that transparency implies privacy, whereas the converse is not true. We also show that the scheme of Kundu and Bertino [6] does not achieve transparency, even if the adversary may only ask for a single signature.

We then provide a secure construction of a tree signature scheme for ordered trees supporting redaction of subtrees. Our construction can be implemented with any EU-CMA signature scheme and provides reasonable efficiency. While for general trees with n nodes and large out-degree in nodes, it requires $\mathcal{O}(n^2)$ signature generations, for trees with bounded out-degree the number of signatures is linear in n. Our construction also permits incremental signing of trees, i.e., if a leaf is added to a signed tree, the signatures on the remaining tree can be re-used and the new signature generated in $\mathcal{O}(n)$ time. We leave it as an open problem to find schemes with better efficiency, still meeting our security notions.

Related Work. Deleting parts of a document while maintaining the integrity and authenticity of the remaining data is an issue that has been approached under different setup assumptions and goals. There have been various approaches to designing redactable signatures [9,10,8], where only linearly ordered documents and the deletion of substrings are considered. Still, they do not require hiding the amount of data removed from the document, i.e., one is able to derive the lengths of the removed strings, or where these were removed from. The former aspect has been addressed in [4] and [3], where the privacy requirement also includes the length of the hidden portions. A solution furthermore hiding their positions is sketched in [4] and this idea is also used as a building block in our construction for tree.

Further works by Ateniese et al. [1] and the extension to sanitizable signatures due to Brzuska et al. [2], where one can modify authenticated data in a controlled way, are influential to our security models. However, sanitization in such contexts requires the input of a secret key, whereas we allow for data to be manipulated by public means. Moreover, sanitizable signatures usually do not hide the amount of sanitized data.

Organization. The paper is organized as follows. After introducing the necessary notation in Section 2 we formally define the functionality of structural signatures for trees in Section 3. We discuss several formal security notions together with their relations in Section 4, and propose a provably secure construction in Section 5. Finally, we show in Appendix A that the scheme by Kundu and Bertino [6] does not achieve our notion of security.

2 Preliminaries

Trees. A *tree* T is a connected graph $G = (V_T, E_T)$ which consists of a nonempty finite set $V_T = \{v_1, \ldots, v_r\}$ of vertices, a set $E_T = \{e_1, \ldots, e_s\}$ of edges and does not contain cycles. We simply write V (resp. E) instead of V_T (resp. E_T) if the context is clear. Edges are denoted $e = (v_i, v_j) \in V \times V$. A tree T_ρ is *rooted* if one vertex $\rho \in V$ (the *root*) is distinguished from the others. The path-distance from node $v \in V$ to the root node ρ is called the *depth* of v. If $e = (v_i, v_j)$ is an edge, then the node that is closer to ρ is called the *parent* of the other node, while the latter is called a *child* of the former. If two vertices have the same parent, then these two vertices are called *siblings*. A *leaf* L is a vertex with no children. The root is the only node without parents. If the children of each vertex in T_ρ are ordered in respect to some linear order relation, then the tree is called *ordered*. Since this paper only concerns trees that are both rooted and ordered, we consider in the following all trees as rooted and ordered. We further assume that all edges $e = (v_i, v_j)$ are directed away from the root, i.e. v_i is parent of v_j. If two trees T and T' are isomorphic (where the isomorphism also maintains the root and the node order), we write $T \simeq T'$ (or $T = T'$). By $T \backslash L$, we denote the tree resulted after cutting leaf L from T; thus the vertex and edge sets of $T \backslash L$ are $V_T \backslash \{L\}$ and $\{(v_i, v_j) \in E_T \mid v_j \neq L\}$. Furthermore, we write $T' \prec T$ for trees T and T' if either $T' \simeq T \backslash L$, or $T' \prec (T \backslash L)$ for some leaf L of T. Consequently, we denote by $T' \preceq T$ the case where $T' \prec T$ or $T' \simeq T$. Note that writing $T' \preceq T$ means saying that T' is a rooted ordered subtree of T with the same root.

Signature Schemes. A signature scheme DS is a tuple $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ of efficient algorithms, where the key generation algorithm $\mathsf{Kg}(1^\lambda)$ returns a key pair (sk, pk); the signing algorithm $\mathsf{Sign}(sk, m)$ takes as input a signing key sk and a message $m \in \{0,1\}^\lambda$, and returns a signature σ; and the verification algorithm

$\mathsf{Vf}(pk, m, \sigma)$ takes public key pk, message m and signature σ, and returns 0 or 1. We assume that the signature scheme is complete, i.e. for any $(sk, pk) \leftarrow \mathsf{Kg}(1^\lambda)$, any message $m \in \{0,1\}^\lambda$, and any $\sigma \leftarrow \mathsf{Sign}(sk, m)$, we have: $\mathsf{Vf}(pk, m, \sigma) = 1$. Note that it is always possible to sign messages of arbitrary length by applying a collision-resistant hash function $h : \{0,1\}^* \mapsto \{0,1\}^\lambda$ to the message prior to signing. The security of signature schemes $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ is defined following [5], as usual. In this model, an adversary may adaptively invoke a signing oracle and is successful if it manages to compute a signature on a *new* message.

Definition 1 (Unforgeability). *A signature scheme* DS *is* unforgeable under adaptive chosen message attacks *(EU-CMA) if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Forge}_{\mathcal{A}}^{\mathsf{DS}}$ *evaluates to 1 is negligible (as a function of* λ*), where*

```
Experiment $\mathsf{Forge}_{\mathcal{A}}^{\mathsf{DS}}(\lambda)$
    $(sk, pk) \leftarrow \mathsf{Kg}(1^\lambda)$
    $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(pk)$
    Return 1 iff $\mathsf{Vf}(pk, m^*, \sigma^*) = 1$ and $\mathcal{A}$ has never queried $\mathsf{Sign}(sk, \cdot)$ on $m^*$.
```

The probability is taken over all coin tosses of Kg*,* Sign*, and* $\mathcal{A}$*.*

3 Structural Signatures for Trees

Kundu and Bertino proposed in [6] special signatures for trees, where parts of the tree can be cut off without invalidating the signature on the rest of the tree and without having to re-sign using the private key. To make formal security claims, we first formally define structural signature schemes for trees. These schemes sign trees and also support one public operation on signed trees: any user may remove parts of the tree and derive a signature for the pruned tree without access to the private key. We define such schemes for the operation of cutting single leaves only; iterating the cutting operation then allows for the removal entire subtrees.

Definition 2 (Structural Signature Scheme for Trees). *A structural signature scheme for trees* $\mathsf{strucSig}$ *consists of four efficient algorithms* $(\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *such that:*

Key Generation. *The key generation algorithm* $\mathsf{sKg}(1^\lambda)$ *outputs a private key* sk *and a corresponding public key* pk*:*

$$(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda).$$

Signing. *Algorithm* $\mathsf{sSign}(sk, T)$ *takes as input a secret key* sk *and a tree* T*. It outputs a structural signature* σ *(with* $T' = T$*):*

$$(T', \sigma) \leftarrow \mathsf{sSign}(sk, T).$$

Verification. *The verification algorithm* sVf *outputs a bit* $d \in \{0,1\}$ *verifying that* σ *is a valid structural signature on a tree* T *with respect to a public key pk:*

$$d \leftarrow \mathsf{sVf}(pk, T, \sigma).$$

Cutting. *The input of the algorithm* $\mathsf{sCut}(pk, T, \sigma, L)$ *is a public key pk, a tree* T*, a signature* σ*, as well as a leaf* L *of* T*. It returns the tree* $T' = T \backslash L$ *and a signature* σ'*:*

$$(T', \sigma') \leftarrow \mathsf{sCut}(pk, T, \sigma, L).$$

We say that a structural signature scheme is correct if:

Signing Correctness. For any $\lambda \in \mathbb{N}$, any key pair $(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda)$, any tree T, and any $(T', \sigma) \leftarrow \mathsf{sSign}(sk, T)$ we have $\mathsf{sVf}(pk, T, \sigma) = 1$.

Cutting Correctness. For any $\lambda \in \mathbb{N}$, any key pair $(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda)$, any tree T, any σ with $\mathsf{sVf}(pk, T, \sigma) = 1$, any leaf L of T, and any pair $(T', \sigma') \leftarrow \mathsf{sCut}(pk, T, \sigma, L)$, we require $\mathsf{sVf}(pk, T', \sigma') = 1$.

Again note that iterative leaf-cutting results in the removal of entire subtrees. It is obvious that any subtree T' of T which can be generated by successive executions of sCut satisfies $T' \preceq T$, and vice versa.

Note that our cutting algorithm relies only on the public key of the signer. In the medical example above, this allows the database to generate authentic tree parts without accessing the private key of the medical personnel.

4 Security of Structural Signature

We define in this section the security properties of structural signature schemes via unforgeability, privacy, and transparency. Informally, these security requirements state:

Unforgeability. No one should be able to compute a valid signature on a tree without having access to the secret key. That is, even if an outsider can request signatures on different trees, it remains impossible to forge a signature. This is analogous to the standard unforgeability requirement for signature schemes.

Privacy. No one should be able to gain any knowledge about parts cut off the tree from its structural signature without having access to these parts. Our definition is similar to the standard indistinguishability notion for encryption schemes.

Transparency. Nobody should be able to decide whether a signature of a tree has been created from scratch, or through an sCut. This means that a party who receives a signed tree cannot tell whether he received a freshly signed tree or a subtree of a signed tree where some parts have already been cut off.

In the following we will define these notions formally. We note that our definitions resemble the ones of Brzuska et al. [2] for sanitizable signatures which, in turn, refine previous notions [1,10] for sanitizable and redactable signatures. Yet, our notion here takes into account the (tree) structure of documents and allows *public* sanitizations.

4.1 Unforgeability

The unforgeability definition for structural signatures is defined analogously to the standard security requirement for signature schemes. Informally, it states that no one should be able to compute a valid signature σ on a tree T without having access to the secret key *sk*. This condition must hold even if the adversary can request signatures on q (possibly adaptively chosen) other trees. The forgery must be non-trivial in the sense that it is not the result of a sequence of cutting operations on a tree for which the adversary has previously requested a signature (recall that the cut algorithm operates on public data only).

Definition 3 (Unforgeability). *A structural signature scheme* $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *is* unforgeable under adaptively chosen tree attacks *if for any efficient algorithm $\mathcal{A}$ the probability that the experiment* $\mathsf{Unforgeability}_{\mathcal{A}}^{\mathsf{strucSig}}$ *evaluates to 1 is negligible (as a function of λ), where*

```
Experiment $\mathsf{Unforgeability}_{\mathcal{A}}^{\mathsf{strucSig}}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{sKg}(1^\lambda)$
    $(T, \sigma) \leftarrow \mathcal{A}^{\mathsf{sSign}(sk,\cdot)}(pk)$
        for $i = 1, 2, \ldots, q$, denote by $T_i$ resp. $\sigma_i$ the
        queries to, resp. answers from, the oracle sSign
    return 1 iff
        $\mathsf{sVf}(pk, T, \sigma) = 1$ and
        for all $i = 1, 2, \ldots, q$ we have $T \not\preceq T_i$
```

The probability is taken over all coin tosses of $\mathsf{sKg}, \mathsf{sSign}$, *and* $\mathcal{A}$.

4.2 Hiding Properties

Preventing leakage of information roughly means that it should be infeasible for anybody to recover further information on the cut parts of the tree from the structural signature. Here we propose two different notions, the first definition being weaker than the second one. Intuitively, the first notion hides the contents of the cut parts, but not necessarily the cut operations themselves, whereas our stronger notion also hides whether cutting operations have been performed.

Privacy. A basic requirement in the medical data example is that a party cannot gain any information on the parts of the tree that were cut off. This is formalized by demanding that, given a subtree with a signature and two possible source trees, one cannot decide from which source tree the subtree stems from. Intuitively, it follows that one cannot derive any information about the cut parts.

The definition of privacy for structural signatures is based on the indistinguishability definition for encryption schemes: an adversary $\mathcal{A}$ can choose two pairs $(T_0, L_0), (T_1, L_1)$ of trees and leaves such that $T_0 \backslash L_0 \simeq T_1 \backslash L_1$, i.e., removal of the leaves results in isomorphic trees. Furthermore, $\mathcal{A}$ has access to a *left-or-right* oracle, which, given those two trees, consistently either returns a cut signature for the left pair ($b = 0$) or for the right pair ($b = 1$). The scheme offers privacy, if no adversary can decide whether the oracle returns the left or the right cut tree.

Definition 4 (Privacy). *A structural signature scheme* $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *is* private *if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{LeakPriv}_{\mathcal{A}}^{\mathsf{strucSig}}$ *evaluates to 1 is negligibly close to* $1/2$ *(as a function of* λ*), where*

```
Experiment $\mathsf{LeakPriv}_{\mathcal{A}}^{\mathsf{strucSig}}(\lambda)$
    $(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda)$
    $b \leftarrow \{0,1\}$
    $d \leftarrow \mathcal{A}^{\mathsf{sSign}(sk,\cdot),\mathsf{SignCut}(\cdot,\cdot,\cdot,\cdot,sk,b)}(pk)$
    return 1 if $d = b$.

$\mathsf{SignCut}(T_{j,0}, L_{j,0}, T_{j,1}, L_{j,1}, sk, b)$
if $T_{j,0} \backslash L_{j,0} \not\cong T_{j,1} \backslash L_{j,1}$ abort
$(T_{j,b}, \sigma_{j,b}) \leftarrow \mathsf{sSign}(sk, T_{j,b})$
return $(T'_b, \sigma'_b) \leftarrow \mathsf{sCut}(pk, T_{j,b}, \sigma_{j,b}, L_{j,b})$
```

The probability is taken over all coin tosses of b, $\mathsf{sKg}, \mathsf{sSign}, \mathsf{SignCut}$ *and* $\mathcal{A}$. *(Note that for trees the graph isomorphism problem can be decided in polynomial time.)*

Note that, similar to the case of encryption, a hybrid argument shows that allowing the adversary to perform multiple cutting operations per oracle call is equivalent to the case in which only a single cutting operation is performed.

Transparency. The above notion of privacy does not prevent the following information leakage in the medical example: a party may learn that data about the patient's psychological treatment has been deleted from his subtree, although he cannot deduce the actual data. To capture a stronger notion of leakage prevention, we present a definition which not only protects the structure of the tree, but also the operations that may have been performed on it. Intuitively, an adversary should be unable to decide whether he is given a signed tree whose signature has been derived by an sCut operation, or a freshly signed tree. Let $(T', \sigma_{T'}) \leftarrow \mathsf{sCut}(pk, T, \sigma, L)$ be a signed tree derived from the signed tree (T, σ) by application of the leaf-cutting algorithm, and let $(T', \sigma_S) \leftarrow \mathsf{sSign}(sk, T')$ be a signature of T', generated from scratch (without doing any leaf-cutting). The task for the adversary is to distinguish both cases.

Definition 5 (Transparency). *A structural signature scheme* $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *is* transparent *if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{LeakTrans}_{\mathcal{A}}^{\mathsf{strucSig}}$ *evaluates to 1 is negligibly close to* $1/2$ *(as a function of* λ*), where*

```
Experiment $\mathsf{LeakTrans}_{\mathcal{A}}^{\mathsf{strucSig}}(\lambda)$
    $(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda)$
    $b \leftarrow \{0,1\}$
    $d \leftarrow \mathcal{A}^{\mathsf{sSign}(sk,\cdot),\mathsf{SignOrCut}(\cdot,\cdot,sk,b)}(pk)$
return 1 if $d = b$.

$\mathsf{SignOrCut}(T, L, sk, b)$
if $b = 0$ :
    $(T, \sigma) \leftarrow \mathsf{sSign}(sk, T)$
    $(T', \sigma') \leftarrow \mathsf{sCut}(pk, T, \sigma, L)$
if $b = 1$ :
    $T' = T \backslash L$
    $(T', \sigma') = \mathsf{sSign}(sk, T')$
return $(T', \sigma')$
```

The probability is taken over all coin tosses of $b, \mathsf{sKg}, \mathsf{sSign}, \mathsf{SignOrCut}$, *and* $\mathcal{A}$.

As for privacy, a hybrid argument again shows that this notion is robust in the sense that it already implies security against adversaries that pass several cut operations in a single oracle call instead of only one.

Note further that it is easy to see that the construction of Kundu and Bertino [6] does not satisfy this strong definition of transparency; for an analysis see Appendix A.

As mentioned, transparency provides strong hiding guarantees and is desirable in many cases. However, for various application examples privacy is in fact sufficient, namely in all cases, where the receiver already expects partly sanitized documents. This is the case for e.g. all anonymization procedures where, a party's data (patient's name) is removed. Therefore, privacy is a sufficient requirement for some applications and by using a private, non-transparent scheme, one thereby gains in efficiency. Thus, both security requirements deserve a formal treatment.

4.3 Relationships of the Security Requirements

In this section we show that transparency is strictly stronger than privacy. We first prove formally that transparency implies privacy. Then we separate the notions by turning a structural signature scheme that offers privacy into one which still has this property, but which violates transparency.

It is clear that unforgeability does not follow from privacy (and thus not from transparency). Take, for example, the trivial scheme which outputs constants as signatures, say, $\sigma = 0$; the cut algorithm for this scheme prunes the three and also outputs $\sigma = 0$. This scheme is clearly transparent, but easily forgeable. Vice versa, it holds that unforgeability implies neither privacy, nor transparency (e.g., take an unforgeable scheme and modify the cut algorithm to append the original tree to the output signature).

Proposition 1 (Transparency⇒Privacy). *Any transparent structural signature scheme is also private.*

Proof. Assume towards contradiction that there exists a transparent structural signature scheme $\mathsf{strucSig}$ which is not private, i.e., there exists an efficient adversary $\mathcal{A}$ that breaks the privacy of $\mathsf{strucSig}$ with non-negligible probability $1/2 + 1/\mathsf{poly}(\lambda)$ for some polynomial $\mathsf{poly}(\lambda)$. We derive a contradiction showing how to construct a successful algorithm $\mathcal{B}$ against transparency. The input of $\mathcal{B}$ is a public key *pk*. It runs a black-box simulation of $\mathcal{A}$ on input *pk* and picks a random bit b^*. Whenever $\mathcal{A}$ invokes its signing oracle $\mathsf{strucSig}$ on a tree T and some leaf L, then $\mathcal{B}$ answers this query with its sSign oracle. For every query $(T_0, L_0), (T_1, L_1)$ that $\mathcal{A}$ sends to its $\mathsf{SignCut}$ oracle, $\mathcal{B}$ forwards (T_{b^*}, L_{b^*}) to its external $\mathsf{SignOrCut}$ oracle and sends the answer to $\mathcal{A}$. Eventually, $\mathcal{A}$ stops outputting a decision bit d. Algorithm $\mathcal{B}$ outputs $a^* = 0$ iff $d = b^*$.

For the analysis first observe that $\mathcal{B}$ is efficient because $\mathcal{A}$ runs in polynomial time and handling all queries can also be done efficiently. We now look at the probability of $\mathcal{B}$ being successful:

– Given that $b = 0$, then the $\mathsf{SignOrCut}$ oracle always signs and applies the cutting algorithm afterwards. Thus, the simulation from $\mathcal{A}$'s point of view is identical to the attack against privacy (with random bit $b^* = 0$). Hence,

$$\text{Prob}[\,a^* = 0 \mid b = 0\,] = \text{Prob}[\,\mathcal{A} = b^* \mid b = 0\,] \geq 1/2 + 1/\mathsf{poly}(\lambda).$$

In other words, the probability of success is lower-bounded by $\mathcal{A}$'s success probability.

– Given on the other hand $b = 1$, then $\mathsf{SignOrCut}$ signs the modified tree T' directly. Bit b^* is information theoretically hidden from $\mathcal{A}$. This follows because the privacy experiment demands that the modified trees have to be identical. Thus, the input of the signing algorithm is independent of b^*:

$$\text{Prob}[\,a^* = 1 \mid b = 1\,] = \frac{1}{2}.$$

The overall success probability of $\mathcal{B}$ is now at least

$$\begin{aligned}\text{Prob}[\,\mathcal{B} = b] &= \text{Prob}[\,b = 0] \cdot \text{Prob}[\,\mathcal{B} = 0 \mid b = 0\,] \\ &\quad + \text{Prob}[\,b = 1] \cdot \text{Prob}[\,\mathcal{B} = 1 \mid b = 1] \\ &\geq \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{1}{\mathsf{poly}(\lambda)}\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{2 \cdot \mathsf{poly}(\lambda)},\end{aligned}$$

which is non-negligibly larger than 1/2. □

The following proposition separates both notions by showing that not all private structural signature schemes are also transparent:

Proposition 2 (Privacy$\not\Rightarrow$Transparency). *Suppose that there exists a private structural signature scheme. Then there exists a private scheme which is not transparent.*

Proof. To prove this separation, we modify a structural signature scheme that provides privacy in such a way that it does leak the information what kind of operation has been performed. To do so, we append a bit to the signature indicating whether a sign ($b = 0$) or cut ($b = 1$) operation has been performed.

More precisely, let $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ be a secure structural signature scheme that preserves privacy. We then define the scheme $\mathsf{strucSig}' = (\mathsf{sKg}', \mathsf{sSign}', \mathsf{sVf}', \mathsf{sCut}')$ as follows:

```
$\mathsf{sKg}'(1^\lambda)$
  return $\mathsf{sKg}(1^\lambda)$

$\mathsf{sVf}'(pk, T, \sigma)$
  parses $\sigma = (\sigma' \| b)$
      (with $b \in \{0,1\}$)
  return $\mathsf{sVf}(pk, T, \sigma')$

$\mathsf{sSign}'(sk, T)$
  return $\mathsf{sSign}(sk, T) \| 1$

$\mathsf{sCut}'(pk, T, \sigma, L)$
  parses $\sigma = (\sigma' \| b)$
      (with $b \in \{0,1\}$)
  return $\mathsf{sCut}(pk, T, \sigma', L) \| 0$
```

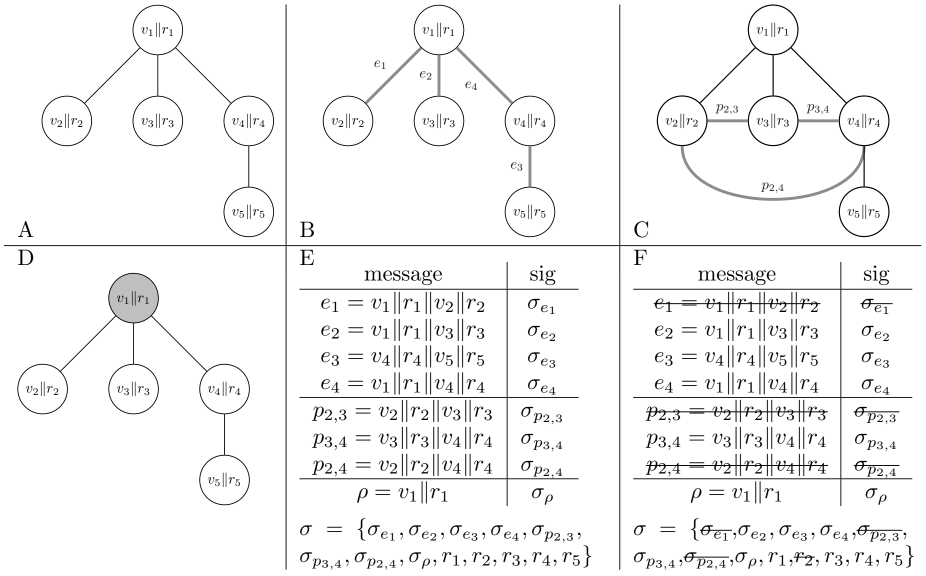
It follows easily from the construction that $\mathsf{strucSig}'$ is efficient and preserves privacy. The scheme, however, is clearly not transparent, because the last bit of a signature directly indicates which operation has been performed. The algorithm $\mathcal{A}$ breaking transparency queries its $\mathsf{SignOrCut}$ oracle on an arbitrary tree and some leaf of the tree. It then parses $\sigma = \sigma' \| b$ and outputs $d = b$. Obviously, this attacker breaks transparency with probability 1. $\square$

5 Constructing Secure Structural Signatures

In this section, we present our structural signature scheme for ordered trees which is unforgeable, transparent and private.

5.1 Construction

The idea of our construction is as follows (see Figure 1): We use an ordinary EU-CMA signature scheme to sign all edges in the tree. In order to avoid match-and-mix attacks between several trees, we endow each vertex with a fresh random number (A) and sign for each edge the contents (which could be, in the above-mentioned XML scenario, the XML tags and attributes) of its adjacent vertices (B) together with the associated random numbers.



E

message	sig
$e_1 = v_1\|r_1\|v_2\|r_2$	σ_{e_1}
$e_2 = v_1\|r_1\|v_3\|r_3$	σ_{e_2}
$e_3 = v_4\|r_4\|v_5\|r_5$	σ_{e_3}
$e_4 = v_1\|r_1\|v_4\|r_4$	σ_{e_4}
$p_{2,3} = v_2\|r_2\|v_3\|r_3$	$\sigma_{p_{2,3}}$
$p_{3,4} = v_3\|r_3\|v_4\|r_4$	$\sigma_{p_{3,4}}$
$p_{2,4} = v_2\|r_2\|v_4\|r_4$	$\sigma_{p_{2,4}}$
$\rho = v_1\|r_1$	σ_ρ

$\sigma = \{\sigma_{e_1}, \sigma_{e_2}, \sigma_{e_3}, \sigma_{e_4}, \sigma_{p_{2,3}}, \sigma_{p_{3,4}}, \sigma_{p_{2,4}}, \sigma_\rho, r_1, r_2, r_3, r_4, r_5\}$

F

message	sig
~~$e_1 = v_1\|r_1\|v_2\|r_2$~~	~~σ_{e_1}~~
$e_2 = v_1\|r_1\|v_3\|r_3$	σ_{e_2}
$e_3 = v_4\|r_4\|v_5\|r_5$	σ_{e_3}
$e_4 = v_1\|r_1\|v_4\|r_4$	σ_{e_4}
~~$p_{2,3} = v_2\|r_2\|v_3\|r_3$~~	~~$\sigma_{p_{2,3}}$~~
$p_{3,4} = v_3\|r_3\|v_4\|r_4$	$\sigma_{p_{3,4}}$
~~$p_{2,4} = v_2\|r_2\|v_4\|r_4$~~	~~$\sigma_{p_{2,4}}$~~
$\rho = v_1\|r_1$	σ_ρ

$\sigma = \{$~~σ_{e_1}~~$, \sigma_{e_2}, \sigma_{e_3}, \sigma_{e_4},$~~$\sigma_{p_{2,3}}$~~$, \sigma_{p_{3,4}},$~~$\sigma_{p_{2,4}}$~~$, \sigma_\rho, r_1,$~~$r_2$~~$, r_3, r_4, r_5\}$

Fig. 1. This figure demonstrates a simple application of the algorithms: (A) randomizing vertices, (B) signing edges, (C) signing order of siblings, (D) signing the root, (E) assembling the final signature, and (F) computing the signature when cutting the leftmost leaf. A more detailed description of the steps is given in Section 5.1.

As we consider ordered trees, we also have to protect the order of siblings of a node; we do this by signing elements of the linear order relation between siblings of a node (C). Finally, the root node and its random value need to be signed (D), as trees containing only a single node do not have any edges. The security of the presented construction relies only on the existence of a standard signature scheme, and no further cryptographic assumptions are required.

We start with defining some further notation. We use the notation v interchangeably to denote a node and its content. Let $\mathcal{P}$ denote the set of all parent nodes having more than one child and let $\mathcal{V}_P = (v_{P,1}, \ldots, v_{P,q})$ be the ordered sequence of child nodes of a node $P \in \mathcal{P}$. We write $\mathcal{R}_P \subseteq \mathcal{V}_P \times \mathcal{V}_P$ for the linear order relation on $\mathcal{V}_P$, i.e., $(v_{P,i}, v_{P,j}) \in \mathcal{R}_P$ if and only if $i < j$. We often denote the elements of $\mathcal{R}_P$ as $J := (v_{P,J_1}, v_{P,J_2})$. Similarly, we write r_{P,J_1} and r_{P,J_2} to denote the random values we will assign to v_{P,J_1} and v_{P,J_2}. Furthermore, we write r_ρ for the randomness associated to the root node.

Let $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ be a signature scheme. We construct a structural signature scheme $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ as follows:

Key Generation. On input the security parameter 1^λ, sKg runs the signature scheme's key generation algorithm and outputs $(sk, pk) \leftarrow \mathsf{Kg}(1^\lambda)$.

Signing. The signing algorithm works as follows:

```
$\mathsf{sSign}(sk, T)$ :
  // $T$ is given as graph $G = (V, E)$
  For each vertex $v \in V$:
      $r_v \leftarrow \{0,1\}^\lambda$
  $S :=$ ""
  // sign all edges
  perform a post-order traversal of the tree:
  for each edge $e := (v, w) \in E$ do
      $m_e = v \| r_v \| w \| r_w$
      $\sigma_e \leftarrow \mathsf{Sign}(sk, 0 \| m_e)$
      $S := \sigma_e \| S$
  // sign the order of child nodes of a vertex
  perform a post-order traversal of the tree:
  for each vertex $P \in \mathcal{P}$ and all $J \in \mathcal{R}_P$ do
      $m_{P,J} := v_{P,J_1} \| r_{P,J_1} \| v_{P,J_2} \| r_{P,J_2}$
      $\sigma_{P,J} \leftarrow \mathsf{Sign}(sk, 1 \| m_{P,J})$
      $S := \sigma_{P,J} \| S$
  // sign the root node
      $\sigma_\rho \leftarrow \mathsf{Sign}(sk, 2 \| \rho \| r_\rho)$
  return $(T, \sigma_\rho \| S \| r_{v_1} \| \ldots \| r_{v_{|V|}})$
```

Verification. The verification algorithm works as follows:

```
$\mathsf{sVf}(pk, T, \sigma)$ :
  parse $\sigma$ as $\sigma = \sigma_\rho \| S \| r_{v_1} \| \ldots \| r_{v_{|V|}}$
  // check signature on the root node
  if $\mathsf{Vf}(pk, 2 \| \rho \| r_\rho, \sigma_\rho) = 0$ return 0
  // check signatures over the order of child nodes
  // $\mathcal{V}_P$ of a parent node $P$
  perform a post-order traversal of the tree:
  for each vertex $P \in \mathcal{P}$ in reverse order do
      for all $J \in \mathcal{R}_P$
          parse $S$ as $\sigma_{P,J} \| S'$
          $m_{P,J} := v_{P,J_1} \| r_{P,J_1} \| v_{P,J_2} \| r_{P,J_2}$
          if $\mathsf{Vf}(pk, 1 \| m_{P,J}, \sigma_{P,J}) = 0$ return 0
          $S = S'$
  // check signature for each edge $e \in E$
  perform a post-order traversal of the tree:
  for each edge $e = (v, w) \in T$ in reverse order do
      parse $S$ as $\sigma_e \| S'$
      $m_e = v \| r_v \| w \| r_w$
      if $\mathsf{Vf}(pk, 0 \| m_e, \sigma_e) = 0$ return 0
      $S = S'$
  if $S =$ "" return 1 else return 0
```

Cutting. The cutting algorithm takes a tree T and its valid structural signature σ_T as well as a leaf node $L \in V$ as input. sCut outputs $T' := T \setminus L$ as well as a redacted signature σ'_T, which is constructed from σ_T by removing the signatures σ_e for $e = (P, L) \in E$, $\sigma_{P,J}$ for $J \in \mathcal{R}_P$ with $J = (v, L)$ or $J = (L, v)$ as well as σ_ρ, if $L = \rho$. In addition, r_L is removed from the signature.

It is obvious that the construction provides both signing and cutting correctness, as defined in Section 3.

Efficiency. The complexity of our structural signature scheme is linear in the number of nodes and quadratic in the number of siblings per node. For binary trees, the scheme remains linear in the number of nodes $|V|$, where exactly $\frac{3}{2}(|V|-1)+1$ signature operations are needed; for a tree with a bounded out-degree it also remains linear. We note that the construction in [6] is linear in the number of nodes as well, but is not provably transparent (see Appendix A).

We remark that the idea of signing all pairs of siblings to achieve transparency has been already sketched in [4] *for linear ordered documents.* There, the authors also present a scheme for linear ordered documents with a linear number of signature generations, denoted $\mathcal{RSS}$, which is based on redactable signatures for (non-ordered) sets. If this underlying scheme for sets provides transparency, then so does $\mathcal{RSS}$, and we can then use $\mathcal{RSS}$ in our construction to achieve a transparent scheme with improved efficiency.[1]

5.2 Proof of Security

We show in this section that our construction is unforgeable, transparent, and private.

Theorem 1. *The structural signature scheme* $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *defined above is unforgeable, transparent, and private.*

We prove this theorem via the following propositions.

Proposition 3. *If* $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ *is an unforgeable signature scheme, then the above construction is an unforgeable structural signature scheme.*

Proof. Let $\mathcal{A}$ be a successful adversary against *unforgeability* of structural signatures. Then, we build a successful adversary $\mathcal{B}$ breaking *EU-CMA* of the underlying signature scheme. The simulation works as follows. When $\mathcal{A}$ queries a tree T to its oracle $\mathsf{sSign}(sk, \cdot)$, then $\mathcal{B}$ draws distinct, but random numbers r_v for each vertex $v \in V$, sends the queries $0\|m_e$, $1\|m_{P,J}$, $2\|\rho\|r_\rho$ for $e \in E$, $P \in \mathcal{P}$ and $J \in \mathcal{R}_P$ to its signing oracle $\mathsf{Sign}(sk, \cdot)$, retrieves all signatures, combines them as in the signing algorithm, and returns the tree signature to $\mathcal{B}$. In the end, $\mathcal{A}$ returns (T, σ). We show that if this is a forgery, then a forgery for the underlying signature scheme has occurred. Furthermore, this forgery can be computed efficiently by an extraction algorithm. $\mathcal{B}$ returns the output of the subsequently defined algorithm $\mathsf{Extract}$ to the game. The remaining part of the proof will show that if $\mathcal{A}$ is successful, then so is $\mathcal{B}$.

The crucial idea is to prove that if (T, σ) is a successful forgery, then T contains one of the following elements: a root ρ that was not a root of any previously asked T_i; a signature over an edge m_e not contained in any query tree T_i; or a signature over a siblings' order relation $m_{P,L}$ not contained in any T_i. In the following we provide an extraction algorithm, which extracts such a forgery from the tree signature. We will show afterwards that if (T, σ) is a valid forgery of a tree

[1] The authors in [4] also claim a version of a more efficient scheme, called $\mathcal{SRSS}$, to be transparent, but we were unable to verify this claim.

signature, then one of the three cases must occur and the algorithm successfully outputs a forgery of the underlying signature scheme.

In the algorithm we use the following notation: For $d \in \mathbb{N}^+$, we denote by T_d the tree obtained from T by removing all nodes of depth larger than d; let $V_{d,T}$ be the set of vertices of depth d in tree T and $E_{d,T}$ be the set of edges, such that one node is at depth d and the other, at depth $d-1$. For a node v, denote the edge to its parent node by e_v. Denote by $\mathcal{V}_P^T$ the ordered sequence of child nodes of a single parent node P in tree T, and write $\mathcal{R}_P^T$ for its linear order.

Extraction. The extraction algorithm Extract on input $(pk, (T, \sigma), (T_1, \sigma_1), ..., (T_n, \sigma_n))$ works as follows:

```
if $0 \leftarrow \mathsf{sVf}(pk, (T, \sigma))$, return failure
else if $\forall i\ \rho_T \neq \rho_{T_i}$, return $(2\|\rho\|r_\rho, \sigma_\rho)$
else if $\rho_T = \rho_{T_i}$, then $I := i$
  for $d$ from $1$ to depth of $T_I + 1$ do
      if $E_{d,T} \not\subseteq E_{d,T_I}$
      find $e \in E_{d,T} \setminus E_{d,T_I}$ and return $(0\|m_e, \sigma_e)$
      else if $E_{d,T} \subseteq E_{d,T_I}$
            if $\exists P \in V_{d-1,T}$ such that $\mathcal{R}_P^T \not\subseteq \mathcal{R}_P^{T_I}$
            find $P \in V_{d-1,T}$ and $J \in \mathcal{R}_P^T \setminus \mathcal{R}_P^{T_I}$
            return $(1\|m_{P,J}, \sigma_{P,J})$
return failure
```

Lemma 1. *On inputs pk, $(T_1, \sigma_1), ..., (T_n, \sigma_n)$ and a valid forgery (T, σ) of a tree signature,* Extract *returns a valid forgery against* EU-CMA *security of* $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$.

A proof of the lemma can be found in Appendix B. This proves the proposition. □

Proposition 4. *The structural signature scheme as defined above is transparent.*

Proof. Transparency follows from a simple investigation of distributions: As on identical inputs, the distribution of the output of SignOrCut with $b = 0$ is identical to the distribution of the outputs of SignOrCut with $b = 1$, transparency follows. □

Note that transparency even holds for unbounded adversaries such that the redacted data remains confidential information-theoretically.

The following corollary follows directly from Propositions 1 and 4.

Corollary 1. *The structural signature scheme described above is private.*

5.3 Dynamic Update of Signed Trees

Our structural signature scheme for trees allows for the easy addition of new leaf nodes (and consequently new subtrees) by the signer. This update can be performed efficiently in the sense that the signer does not need to refresh any of the existing signatures (on edges and sibling order) that constitute the structural signature. The signer only has to sign new edges and new sibling order relationships resulting from the inclusion of some leaf node (or subtree) into the original tree, and update the structural signature with these signatures. In the following we provide a pseudo-code of the signature update algorithm that adds a new leaf L into some existing tree-signature pair (T, σ_T) and updates the signature. Note that this algorithm can be used to iteratively update T and σ_T with subtrees containing more than one leaf.

```
sSignUpd$(sk, T, \sigma_T, P, v_P, L)$ :
  // $T$ is given as graph $G = (V, E)$, $\sigma_T$ is the signature of $T$, $P$ is a node
  // in $T$, $v_P$ is either a child node of $P$ or $\perp$, $L$ is a new leaf node that
  // should be inserted in $T$ as a sibling node following $v_P$ or as the new
  // first child node of $P$ if $v_P = \perp$
  parse $\sigma_T$ as $\sigma_\rho \| S_s \| S_e \| r_{v_1} \| \ldots \| r_{v_{|V|}}$
  // here $S_s$ contains concatenated signatures on the order of siblings in $T$
  // and $S_e$ contains concatenated signatures of edges in $T$
  // for every vertex $v_i$, $r_{v_i}$ denotes the randomness for that vertex
  update $T := (V \cup \{L\}, E \cup \{(P, L)\})$ with $L$
  $r_L \leftarrow \{0,1\}^\lambda$
  // sign the order of all siblings of new leaf node $L$ and add them to $S_s$
  update $\mathcal{R}_P$ to $\mathcal{R}_P^*$ using new relationships $\{(L, v_{P,i})\}_i$ and $\{(L, v_{P,j})\}_j$
  for all $J \in \mathcal{R}_P^* \setminus \mathcal{R}_P$ do
    $m_{P,J} := v_{P,J_1} \| r_{P,J_1} \| v_{P,J_2} \| r_{P,J_2}$
    $\sigma_{P,J} \leftarrow \mathsf{Sign}(sk, 1 \| m_{P,J})$
    insert $\sigma_{P,J}$ into $S_s$ preserving the post-order of the latter
  // sign the edge $(P, L)$
  $m_e = P \| r_P \| L \| r_L$
  $\sigma_e \leftarrow \mathsf{Sign}(sk, 0 \| m_e)$
  insert $\sigma_e$ into $S_e$ preserving the post-order of the latter
  insert $r_L$ into sequence of random numbers preserving their post-order
  denote the re-ordered random values by $r'_{v_i}$ for $i \in \{1, \ldots |V| + 1\}$
  return $(T, \sigma_\rho \| S_s \| S_e \| r'_{v_1} \| \ldots \| r'_{v_{|V|+1}})$
```

We note that the above algorithm preserves the hiding properties (privacy and transparency) of the input tree-signature pairs (T, σ_T). Indeed, each edge and each linear order relation have their own signatures. Similarly to the sCut algorithm, which removes irrelevant signatures from the original set, the sSignUpd updates the set with new signatures. An adversary is thus unable to distinguish whether some σ_T is output by a single execution of sSign or of several consecutive executions of sSignUpd.

Such dynamic updates of signed trees are a very valuable feature of our scheme since in the envisioned applications, such as XML records with medical data, the documents are frequently updated with new diagnoses or treatment procedures.

Acknowledgments

We thank Moti Yung and the anonymous reviewers for valuable comments. Marc Fischlin and Dominique Schröder are supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG). This work was supported by CASED (`http://www.cased.de`).

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of San-itizable Signatures Revisited. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography – PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
3. Bertino, E., Kundu, A.: A New Model for Secure Dissemination of XML Content. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38, 292–301 (2008)
4. Chang, E.-C., Lim, C.L., Xu, J.: Short Redactable Signatures Using Random Trees. Cryptology ePrint Archive, Report 2009/025 (2009), `http://eprint.iacr.org/`; A preliminary version has appeared at Fischlin, M. (ed.): CT-RSA 2009. LNCS, vol. 5473. Springer, Heidelberg (2009)
5. 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)
6. Kundu, A., Bertino, E.: Structural signatures for tree data structures. Proceedings of the VLDB Endowment 1(1), 138–150 (2008)
7. Kundu, A., Bertino, E.: Leakage-Free Integrity Assurance for Tree Data Structures. Technical Report 2009-1, CERIAS (2009)
8. Miyazaki, K., Hanaoka, G., Imai, H.: Invisibly Sanitizable Digital Signature Scheme. IEICE Transactions 91-A(1), 392–402 (2008)
9. Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H.: Digital documents sanitizing problem. Technical Report ISEC2003-20. IEICE (2003)
10. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)

A Randomized Traversal Numbers

Kundu and Bertino [6] use traversal numbers to assign unique numbers to nodes; subsequently, all edges (including the content of adjacent nodes and their unique numbers) are signed. Consider for example a binary tree with a root v_0 and left child v_l and right child v_r; we will denote the unique numbers assigned to the vertices as r_0, r_l, r_r. Traversing the tree in pre-order we obtain the sequence of associated numbers $(r_0, r_l, r_r) = (1, 2, 3)$. Similarly, if we perform a post-order traversal, we obtain $(r_0, r_l, r_r) = (3, 1, 2)$. Given both pre- and post-order traversal numbers one can reconstruct the tree.

As noted in [6], since the post-order traversal number 2 of the right child v_r in the example above reveals that this node had a sibling, even after v_l has been pruned, such numbers inhibit transparency. Therefore, [6] introduce random traversal numbers which are basically order-preserving random numbers. These numbers remain unchanged during the document's life time. In the full version [7] of the paper the authors outline three implementations of such random traversal numbers:

– Sorted random numbers: Generate sufficiently random numbers, sort them, and assign them to nodes.
– Order-preserving encryption: Assign ordered random numbers to the nodes and apply order-preserving encryption.[2]
– Addition of random numbers: iteratively assign numbers to nodes by taking the previous traversal number and adding a random offset.

We discuss next that none of the methods above yields a transparent solution.[3] Consider again our simple example of a tree with root, left child and right child. Suppose for the moment that the scheme uses post-order traversal. From an abstract point of view, this traversals assign random numbers $(r_l, r_r, r_0) \leftarrow R$ to the nodes v_l, v_r, v_0 (visited in this order) according to some distribution R. This distribution is balanced in the first argument for the examples above, i.e., letting $\mu = \mathbf{E}[r_l]$ be the expected random number assigned to v_l (and assuming for sufficiently large random numbers simply that $\text{Prob}[r_l = \mu] = 0$), we have $\text{Prob}[r_l \leq \mu] = \text{Prob}[r_l \geq \mu] = \frac{1}{2}$. Furthermore, we have $\text{Prob}[r_r \geq \mu] \geq \frac{3}{4}$ in the examples above, since r_r is the largest among two random traversal numbers and is thus only smaller than μ if both r_l and r_r are below μ.

We can now break transparency for the simple three-node tree as follows. In the experiment, we either get a structural signature for the tree containing only the nodes v_0 and v_r or a signature for the whole tree, but where v_l has been cut. To solve the challenge, we check the randomized post-order traversal number of the only child v_r and output 0 if $r_r \geq \mu$, and 1 otherwise. We remark that, following Kerckhoff's principle, the expectation μ should be assumed to be known by the adversary (since μ can be derived from the signing algorithm).

As for the analysis note that, if the signer creates a signature by cutting a previously generated signature for the full tree, then r_r is larger than μ with probability $\frac{3}{4}$. If, on the other hand, the pruned tree is signed from scratch, then r_r is distributed as r_v in the original tree and thus $\text{Prob}[r_r \geq \mu] = \frac{1}{2}$. It follows that we predict the secret choice b with probability $\frac{5}{8}$, which contradicts Definition 5. Note that the attack even works on a very simple tree and requires only one tree signature.

Note further that [6] also discusses other uniquely determining traversal combinations, like in-order traversal for binary trees together with post-order traversal; the aforementioned attack applies in a similar fashion.

[2] The description of this step is rather sketchy but all possible interpretations seem to suffer from the same problems discussed below.

[3] Note that transparency, although not defined rigorously, *is* mentioned as a desired security property in [6].

B Proof of Lemma 1

Essentially, the proof is an induction over the depth of the tree, i.e. we show at each level d of the tree, that either we already found a forgery against *EU-CMA* security of $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$, or there are still nodes at level $d+1$ in T, i.e. there is at least one node at depth d having at least one child node. The finiteness of the tree will assure that at some point, we find a forgery.

Within the proof, we use the following statement in an essential way: Let T' be a connected subgraph of T containing the distinguished vertex ρ and respecting the sibling order. By a simple induction proof over the number of nodes $|V \setminus V'|$, one can show that starting with T, via successive leaf cutting operations, one can obtain T'. This entails that any such connected subgraph of any of the T_i would not be a valid forgery, as $T \preceq T_i$. Therefore, in the following, we may use $\preceq$ and "subtree of" interchangeably.

We assume that (T, σ) is a valid forgery with respect to queries $(T_i, ..., T_n)$ to sSign, and we show that $\mathsf{Extract}$ extracts a valid forgery against the underlying signature scheme. The proof follows the structure of the $\mathsf{Extract}$ algorithm:

First of all, (T, σ) is a valid forgery which entails that it is validly signed. Therefore, $0 \leftarrow \mathsf{sVf}(pk, (T, \sigma))$ is impossible, and $\mathsf{Extract}$ does not return failure in the first execution step.

If for all $i = 1, ..., n$, $\rho_T \neq \rho_{T_i}$, then $(2 \| \rho \| r_\rho, \sigma_\rho)$ is a valid forgery against the underlying signature scheme, because $\mathsf{Sign}(sk, \cdot)$ queries beginning with 2 are only asked for root nodes, as queries for edges and siblings order relation do not start with 2.

If for some $i = 1, ..., n$, $\rho_T = \rho_{T_i}$ and $r_{\rho_T} = r_{\rho_{T_i}}$, we fix $I := i$ as to adapt the $\mathsf{Extract}$ algorithm notation. Before getting to the induction proof, we introduce some helpful notation first: For $d \in \mathbb{N}$, let T^d denote T cut at depth d, i.e. all nodes having distance strictly greater than d from the root node ρ_T are cut. We show by induction on the depth of the tree T that the following statement is true up to negligible probability: At each level $d \geq 0$, either

(i) we already found a forgery against *EU-CMA* security of $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$, or
(ii) T^d is a subtree of T_I.

Note that the latter always entails that the depth of T is strictly greater than d.

Base case. As $\rho_T = \rho_{T_I}$, T^0 is a subtree of T_I. Therefore, the statement is clearly true for $d = 0$.

Induction hypothesis. We assume that at level $d-1$, $1 \leq d$, we already found a forgery against *EU-CMA* security of $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ or T^{d-1} is a subtree of T_I.

Induction step. If we already found a forgery at stage $d-1$ or lower, the statement is trivially true for d. It is thus sufficient to treat the case, where there is no forgery until level $d-1$ and T^{d-1} is a subtree of T_I. As this entails that there is at least one node in T at level $d-1$ having at least one child, there is at least one node in T at level d. We now consider all of these nodes:

1. **New edge.** Assume there is an edge $e \in E_{d,T} \setminus E_{d,T_I}$, which we denote by $(v, w) = e$. We claim that $(0 \| v \| r_v \| w \| r_w, \sigma)$ was no output of $\mathsf{Sign}(sk, \cdot)$: First

of all, only queries for edges need to be considered, as only those start by 0. Furthermore, only queries for edges in T_I are relevant, as by construction, r_w and r_v are unique and do not appear in other trees except for negligible probability. Therefore $0\|v\|r_v\|w\|r_w$ can only be an edge query for an edge in T_I. Furthermore, r_v and r_w are unique within T_I. If $0\|v\|r_v\|w\|r_w$ had been queried to $\mathsf{Sign}(sk, \cdot)$, this would mean that e is contained in E_{T_I} and because of the randomness' uniqueness within T_I, it follows $e \in E_{d,T_I}$, a contradiction. Thus, $(0\|v\|r_v\|w\|r_w, \sigma)$ is a valid forgery.

2. **Wrong order of siblings.** Now, assume that there is no forgery until level $d-1$ and T^{d-1} is a subtree of T_I and furthermore, $E_{d,T} \subseteq E_{d,T_I}$. For the sake of contradiction, assume that there is a $P \in V_{d-1,T}$ and a $J \in \mathcal{R}_P^T \setminus \mathcal{R}_P^{T_I}$. We claim that $1\|v_{P,J_1}\|r_{P,J_1}\|v_{P,J_2}\|r_{P,J_2}$ had not been asked to $\mathsf{Sign}(sk, \cdot)$: first of all, as in the previous case, only signatures related to T_I need to be considered up to negligible probability. And within T_I, only order relation signatures may have caused such a query to $\mathsf{Sign}(sk, \cdot)$. Furthermore, by uniqueness of the random values r_{P,J_1} and r_{P,J_2} within T_I, such strings may only be signed as order relation strings within $P \in V_{d-1,T}$, whereas $J \in \mathcal{R}_P^T \setminus \mathcal{R}_P^{T_I}$ was assumed. Thus, $(1\|v_{P,J_1}\|r_{P,J_1}\|v_{P,J_2}\|r_{P,J_2}, \sigma_{P,J})$ is a valid forgery against the underlying signature scheme.
3. If T^{d-1} is a subtree of T_I and if at stage d, we neither found a forgery in the two previous cases, then T^d is a subtree of T_I, as all nodes added while getting from T^{d-1} to T^d do exist in the same position and same order in T_I. For a more formal argument, T^{d-1} being a subtree of T_I entails that there is an embedding from T^{d-1} into T_I, and the subset relations $E_{d,T} \subseteq E_{d,T_I}$ and $\mathcal{R}_P^T \subseteq \mathcal{R}_P^{T_I}$ define a unique way to extend this embedding to an embedding from T^d into T_I.

Finally, we argue that at least at one stage d, we find a valid forgery against the underlying signature scheme, if (T, σ) is a valid forgery. T^d cannot be a subtree of T_I at all levels d, as the trees are finite and thus, at some point, $T = T_d$. Thus, $T^d \preceq T_I$ would contradict the assumption that T is a valid forgery. Thus, we are sure to find a forgery at some level, which concludes the proof of the lemma.

Impossible Differential Cryptanalysis on Feistel Ciphers with *SP* and *SPS* Round Functions

Yuechuan Wei[1], Ping Li[2], Bing Sun[2], and Chao Li[1,2,3]

[1] School of Computer Science, National University of Defense Technology, Changsha, China, 410073
[2] Science College of National University of Defense Technology, Changsha, China, 410073
[3] State Key Laboratory of Information Security, Chinese Academy of Sciences, Beijing, China, 100049

wych004@163.com, leave17@gmail.com, happy_come@163.com, lichao_nudt@sina.com

Abstract. Impossible differential cryptanalysis is well known to be effective in analyzing the security of block ciphers. Known result shows that there always exists 5-round impossible differentials of a Feistel cipher with bijective round function. However, if more details of the round function are known, the result could be improved. This paper mainly studies the impossible differentials of Feistel ciphers with both SP and SPS round functions where the linear transformation P is defined over $\mathbb{F}_2^{n\times n}$. For Feistel ciphers with SP round functions, any column of $P \oplus P^{-1}$ whose Hamming weight is greater than 1 corresponds to some 6-round impossible differentials. The existence of some 7-round impossible differentials can be determined by counting the times that 1 appears at some special positions of P and P^{-1}. Some 8-round impossible differentials can be found by computing the rank of some sub-matrix of P. Impossible differentials of Camellia found by these techniques are well consistent with previously known results. For Feistel ciphers with SPS round functions, by determining the rank of some sub-matrix of P, 6-round impossible differentials can be found, which improves the results on E2 by one round. These results tell that when designing a Feistel cipher with SP or SPS round function where the diffusion layer is selected from $\mathbb{F}_2^{n\times n}$, the linear transformation should be chosen carefully to make the cipher secure against impossible differential cryptanalysis.

Keywords: Block cipher, Feistel cipher, Impossible differential.

1 Introduction

Impossible differential cryptanalysis, proposed by Biham and Knudsen, was first applied to the cipher DEAL [7] and later to Skipjack [8]. The main idea is to specify a differential with probability zero over some rounds of the cipher. Then one can derive the right keys by discarding the wrong keys which lead to the impossible differential. Impossible differential cryptanalysis has been applied to AES, Camellia, MISTY1 and so on with very good results [10–16].

The key step of impossible differential cryptanalysis is to retrieve the longest impossible differential. The main technique is miss-in-the-middle [8, 9], namely to find two differential characteristics with probability 1 from encryption and decryption directions, and connect them together. When there are some inconsistencies, their combination is the impossible differential that we are looking for. Once the impossible differential is found, it can be used to distinguish the cipher from a random permutation. In [17], Kim *et al.* introduced the $\mathcal{U}$-method to find impossible differentials of various block ciphers. However, $\mathcal{U}$-method is so general that some information is often lost during calculating the impossible differentials. Some longer impossible differentials cannot be found by using the $\mathcal{U}$-method.

The class of block ciphers considered in this paper is Feistel cipher with SP and SPS round functions whose diffusion layers can be represented by matrices over $\mathbb{F}_2$. These structures are worth being looked at since they are so popular that they have been employed by many famous ciphers, including Camellia, E2 and so on. For Feistel structure, 5-round impossible differential always exists if the round function is bijective [7]. However, if more details of the round function are taken into consideration, we can prove the existence of impossible differentials over more than 5 rounds. By carefully analyzing the properties of the linear transformations, we found that the existence of impossible differentials in a cipher is strongly related to the properties of the diffusion layer P. We should emphasize that the idea of exploiting incomplete diffusion of the round function is not new. Impossible differential for 7 rounds of DES is shown in [9], 8-round impossible differential for Camellia has been used in the previous attacks [11].

The contribution of this paper is an improvement of the original judgement about the impossible differential for Feistel cipher. Instead of searching by experience and intuition, some sufficient conditions are given to characterize the existence of 6/7/8-round impossible differentials of Feistel cipher with SP round functions and 6-round impossible differential of Feistel cipher with SPS round functions. One can discover these impossible differentials just by observing the linear transformation. All of these kinds of impossible differentials cannot be found by $\mathcal{U}$-method. As examples, 6-round impossible differential of E2 is found while previously known round of impossible differentials of E2 is 5 [3]. 8-round impossible differentials of Camellia found by this technique are well consistent with [11].

The paper is organized as follows: Feistel structure and χ-function are described in Section 2. In Section 3 and Section 4, we discuss the existence of impossible differentials of Feistel ciphers with SP and SPS round functions, respectively. Section 5 concludes this paper.

2 Preliminaries

In this section, we describe Feistel structure firstly, and then give the definition and properties of χ-function.

2.1 Feistel Structure

A Feistel network consists of r rounds, each of which is defined as follows. Denote by (L, R) the $2n$-bit input, set $\alpha_0 = L$ and $\beta_0 = R$, let $(\alpha_{i-1}, \beta_{i-1})$ be the input to the i-th round, (α_i, β_i) and k_i be the output and the round key of the i-th round, respectively. Then $(\alpha_i, \beta_i) = Round(\alpha_{i-1}, \beta_{i-1})$ is defined as:

$$\begin{cases} \alpha_i = \beta_{i-1}, \\ \beta_i = f(\beta_{i-1}, k_i) \oplus \alpha_{i-1}, \end{cases}$$

where f is the round function and in this paper, we always assume that $f(\beta_{i-1}, k_i) = f(\beta_{i-1} \oplus k_i)$. After iterating $Round$ r times, the ciphertext (C_L, C_R) is defined as (β_r, α_r). According to the definition of round function f, Feistel cipher can be fractionized to many branch structures. Major round functions under study are based on SP structure and SPS structure (See Fig. 1).

The former structure has one nonlinear transformation layer, and one linear transformation layer. Examples of these ciphers are DES [4], Camellia [5]. The later structure consist of 1st nonlinear transformation layer, linear transformation layer, and 2nd nonlinear transformation layer. Example of this kind of cipher is E2 [1].

This paper focuses on the above two kinds of Feistel ciphers with following nonlinear transformation S and linear transformation P. $S : \mathbb{F}_{2^t}^n \rightarrow \mathbb{F}_{2^t}^n$ is defined as $S(x_1, x_2, \ldots, x_n) = (S_1(x_1), S_2(x_2), \ldots, S_n(x_n))$, where $S_i(1 \leq i \leq n)$ are nonlinear bijective mappings on $\mathbb{F}_{2^t}$. P is an invertible linear transformation defined over $\mathbb{F}_2^{n \times n}$.

To be convenient, we simply denote $P = (p_{i,j})_{1 \leq i,j \leq n} = (p_1, \ldots, p_n)$, $P^{-1} = (q_{i,j})_{1 \leq i,j \leq n} = (q_1, \ldots, q_n)$, where p_i and q_i are the i-th columns of P and P^{-1}, respectively. $\mathcal{E}$ denotes a Feistel cipher with SP round function. $\mathcal{D}$ denotes a Feistel cipher with SPS round function. Brief descriptions of Camellia, SNAKE(2) and E2 are presented in Appendix A.

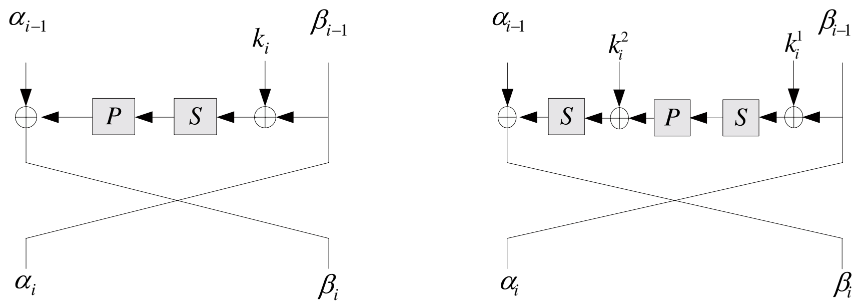


Fig. 1. Feistel Ciphers with SP and SPS Round Function

Proposition 1. *If the round function of a Feistel cipher is bijective, then* $(x, 0) \nrightarrow (0, x)$ *is a* 5*-round impossible differential of the cipher, where* $x \neq 0$.

The above proposition is pointed out by Knudsen. As described in Fig. 2, the output difference of the 3rd round function should be $x \oplus x = 0$, while the input difference is non-zero, which indicates a contradiction since f is bijective.

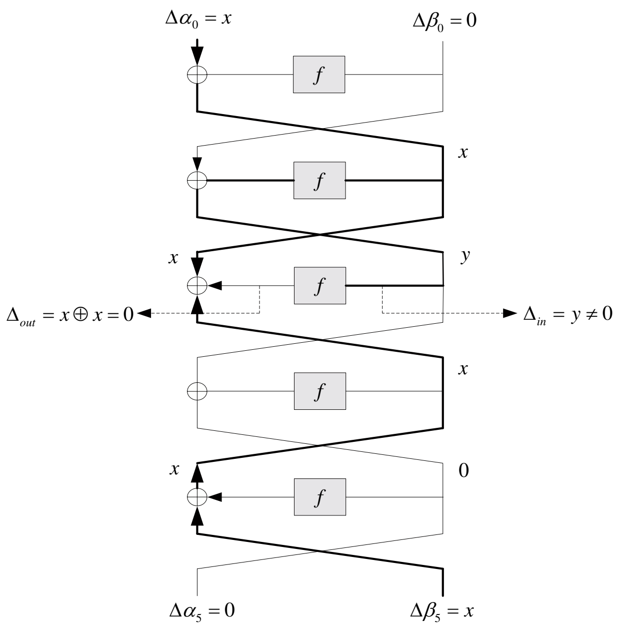


Fig. 2. 5-round Impossible Differential of Feistel Structure

2.2 χ-Function

In this section, we first give the definition of χ-function that maps any element of $\mathbb{F}_{2^t}^n$ to $\mathbb{F}_2^n$, and then discuss basic properties of χ-function.

Definition 1. ***(χ-Function)*** *Let $\theta : \mathbb{F}_{2^t} \to \mathbb{F}_2$ be defined as*

$$\theta(x) = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{if } x \neq 0. \end{cases}$$

Then $\chi : \mathbb{F}_{2^t}^n \to \mathbb{F}_2^n$ is defined as

$$\chi(x_1, x_2, \ldots, x_n) = (\theta(x_1), \theta(x_2), \ldots, \theta(x_n)),$$

while $\chi_s : \mathbb{F}_{2^t}^n \to \mathbb{F}_2$ is defined as

$$\chi_s(x_1, x_2, \ldots, x_n) = \theta(x_s).$$

The χ-function is well used in truncated differential cryptanalysis, when we only consider whether there is a difference or not while the concrete value of the difference is out of consideration. If $\chi_s(\Delta X) = 1$ ($\Delta X \in \mathbb{F}_{2^t}^n$), it means that there is some non-zero difference at the s position.

For convenience, let $E_i \in \mathbb{F}_2^n$ be a vector whose i-th component is 1 while other components are 0, and e_i is any one of the vectors such that $\chi(e_i) = E_i$. For nonlinear transform layer S, we denote $S(X) \oplus S(X \oplus \Delta X)$ by $S(\Delta X)$.

Property 1. (1) For any difference $\Delta X \in \mathbb{F}_{2^t}^n$,

$$\chi(S(\Delta X)) = \chi(\Delta X);$$

(2) Let $P = (p_1, \ldots, p_n)$ where p_i is the i-th column of P, if $\Delta X = e_i$, then

$$\chi(P \circ S(\Delta X)) = \chi(P(\Delta X)) = p_i;$$

(3) Let $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, respectively, if $x_s = 0$, then

$$\chi_s(X \oplus Y) = \chi_s(Y).$$

Definition 2. ***(Hamming Weight)*** *Let $\mathbb{F}_q$ be a finite field with q elements, $X = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$. Then the Hamming Weight of X is defined as the number of non-zero components of X:*

$$w(X) = |\{i | x_i \neq 0, 1 \leq i \leq n\}|.$$

3 Analysis of Round-Reduced Feistel Cipher with *SP* Structure

By carefully analyzing the properties of the linear transformations and taking the χ-function into consideration, some sufficient conditions will be given which characterize the existence of 6/7/8-round impossible differentials of Feistel cipher with SP round functions (Notice that Feistel cipher with this structure is denoted by $\mathcal{E}$).

To apply the miss-in-the-middle technique effectively, we concentrate on differentials with the form $(e_i, 0) \to (0, e_j)$, i.e. both the Hamming weight of input difference and out difference are 1.

3.1 Analysis of 6-Round Feistel Cipher with *SP* Structure

Let (α_r, β_r) be the output of the r-th round, and Y_r and Z_r be the outputs of S-Box layer and P layer of the r-th round, respectively. In the following, impossibility of some differential $(e_i, 0) \to (0, e_j)$ will be proved given that some special properties of the linear transformation P are satisfied.

Proposition 2. *For linear transformation P, let $P \oplus P^{-1} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$, where γ_i is the i-th column of $P \oplus P^{-1}$. If there exists an i, $1 \leq i \leq n$, such that $w(\gamma_i) \geq 2$, then for any j, $1 \leq j \leq n$, $(e_i, 0) \to (0, e_j)$ is a 6-round impossible differential of $\mathcal{E}$.*

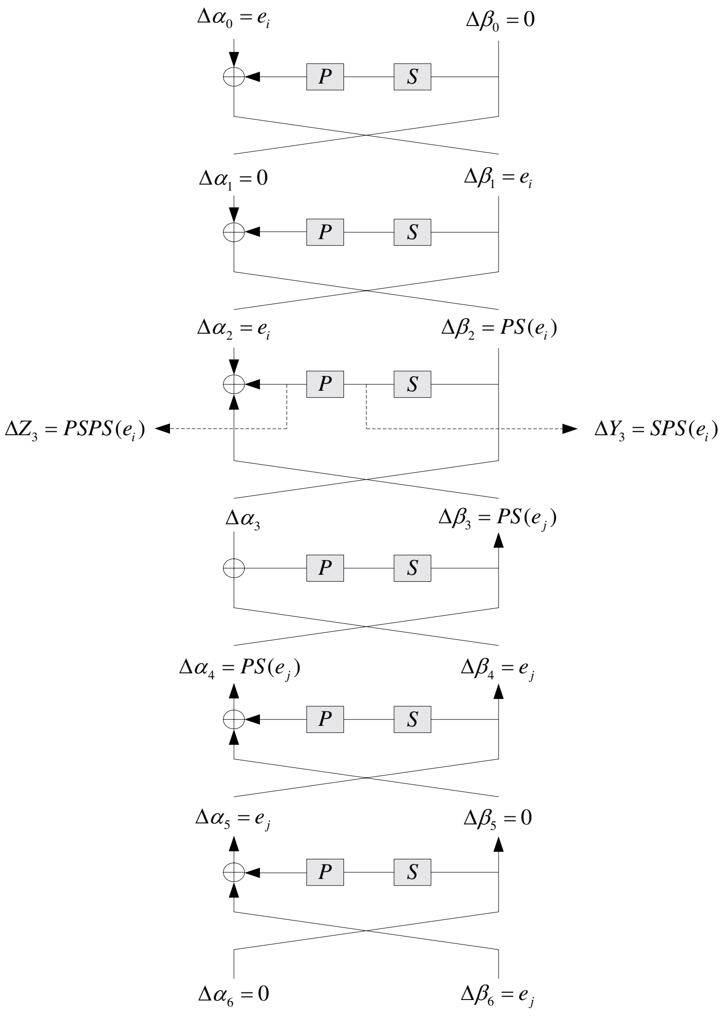


Fig. 3. 6-round Impossible Differential of Feistel-SP

Proof. Fig. 3 describes the 6-round impossible differential. From the encryption direction, if the input difference is

$$\Delta(\alpha_0, \beta_0) = (e_i, 0),$$

the differences of the output of the 1st and 2nd rounds can be calculated as follows:

$$\Delta(\alpha_1, \beta_1) = (0, e_i),$$
$$\Delta(\alpha_2, \beta_2) = (e_i, P \circ S(e_i)).$$

Accordingly, in the third round,

$$\begin{aligned}\Delta Y_3 &= S \circ P \circ S(e_i),\\ \Delta Z_3 &= P \circ S \circ P \circ S(e_i).\end{aligned}$$

From the decryption direction, if the output difference (the 6-th round) is

$$\Delta(\alpha_6, \beta_6) = (0, e_j),$$

the differences of the output of the 5-th and 4-th rounds are

$$\begin{aligned}\Delta(\alpha_5, \beta_5) &= (e_j, 0),\\ \Delta(\alpha_4, \beta_4) &= (P \circ S(e_j), e_j).\end{aligned}$$

According to the Feistel structure,

$$\Delta\alpha_4 = \Delta\beta_3 = \Delta Z_3 \oplus \Delta\alpha_2 = \Delta Z_3 \oplus \Delta\beta_1,$$

the following equation must hold:

$$P \circ S(e_j) = \Delta\alpha_4 = \Delta Z_3 \oplus \Delta\beta_1 = P \circ S \circ P \circ S(e_i) \oplus e_i,$$

which implies that

$$S(e_j) = S \circ P \circ S(e_i) \oplus P^{-1}(e_i),$$

and

$$\chi(S(e_j)) = \chi\left(S \circ P \circ S(e_i) \oplus P^{-1}(e_i)\right).$$

From Property 1,

$$\chi(S(e_j)) = \chi(e_j) = E_j.$$

If $w(p_i \oplus q_i) \geq 2$, which implies that p_i and q_i differ at least 2 positions, say $p_{t_1,i} = 0$, $q_{t_1,i} = 1$ and $p_{t_2,i} = 1$, $q_{t_2,i} = 0$. Thus

$$\begin{aligned}\chi_{t_1}(S \circ P \circ S(e_i) \oplus P^{-1}(e_i)) &= \chi_{t_1}(P^{-1}(e_i)) = 1,\\ \chi_{t_2}(S \circ P \circ S(e_i) \oplus P^{-1}(e_i)) &= \chi_{t_2}(S \circ P \circ S(e_i)) = 1,\end{aligned}$$

which implies that $w(\chi(S \circ P \circ S(e_i) \oplus P^{-1}(e_i))) \geq 2$, and this is contradicted with $\chi(S(e_j)) = E_j$ whose Hamming weight is 1. Thus $(e_i, 0) \to (0, e_j)$ is a 6-round impossible differential. □

Example 1. (6-Round Impossible Differential of Camellia) By careful computation, we have:

$$P \oplus P^{-1} = \begin{pmatrix} 1&1&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 1&0&0&1&0&0&0&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&0&0&1&1\\ 0&0&0&0&1&0&0&1 \end{pmatrix} = (\gamma_1, \gamma_2, \ldots, \gamma_8).$$

Since for any $1 \le i \le 8$, $w(\gamma_i) = 2$, according to Proposition 2, for any $1 \le i, j \le 8$, $(e_i, 0) \to (0, e_j)$ is a 6-round differential of Camellia.

3.2 Analysis of 7-Round Feistel Cipher with *SP* Structure

The 7-round Feistel ciphers with SP round functions can be analyzed similarly.

Proposition 3. *For linear transformation P, if there exists a triplet (i, j, k) such that the multiset $\{p_{k,i}, p_{k,j}, q_{k,i}, q_{k,j}\}$ is equal to $\{1, 0, 0, 0\}$, then $(e_i, 0) \to (0, e_j)$ is a 7-round impossible differential of $\mathcal{E}$.*

Proof. Let $\Delta(\alpha_0, \beta_0) = (e_i, 0)$ and $\Delta(\alpha_7, \beta_7) = (0, e_j)$, respectively. Then by analyzing the propagation of $\Delta(\alpha_0, \beta_0)$ and $\Delta(\alpha_7, \beta_7)$ from the encryption and decryption directions, respectively, we have (see Fig. 4)

$$\begin{aligned}
\Delta(\alpha_1, \beta_1) &= (0, e_i),\\
\Delta(\alpha_2, \beta_2) &= (e_i, P \circ S(e_i)),\\
\Delta Z_3 &= P \circ S \circ P \circ S(e_i),\\
\Delta(\alpha_6, \beta_6) &= (e_j, 0),\\
\Delta(\alpha_5, \beta_5) &= (P \circ S(e_j), e_j),\\
\Delta Z_5 &= P \circ S \circ P \circ S(e_j).
\end{aligned}$$

Since

$$\Delta\alpha_2 \oplus \Delta Z_3 = \Delta\beta_3 = \Delta\alpha_4 = \Delta\beta_5 \oplus \Delta Z_5,$$

thus

$$e_i \oplus P \circ S \circ P \circ S(e_i) = e_j \oplus P \circ S \circ P \circ S(e_j),$$

from which we have

$$P^{-1}(e_i) \oplus P^{-1}(e_j) = (S \circ P \circ S(e_i)) \oplus (S \circ P \circ S(e_j)).$$

Let $\rho_1 = \chi(P^{-1}(e_i))$, $\rho_2 = \chi(P^{-1}(e_j))$, $\rho_3 = \chi(S \circ P \circ S(e_i))$, $\rho_4 = \chi(S \circ P \circ S(e_j))$. According to Property 1, the following equations hold:

$$\begin{aligned}
\rho_1 &= q_i,\\
\rho_2 &= q_j,\\
\rho_3 &= p_i,\\
\rho_4 &= p_j.
\end{aligned}$$

Assume that there exists some t, such that $\{\rho_{1,t}, \rho_{2,t}, \rho_{3,t}, \rho_{4,t}\} = \{1, 0, 0, 0\}$, say $\rho_{1,t} = 1$, and $\rho_{2,t} = \rho_{3,t} = \rho_{4,t} = 0$, then

$$\chi_t(P^{-1}(e_i) \oplus P^{-1}(e_j)) = 1,$$

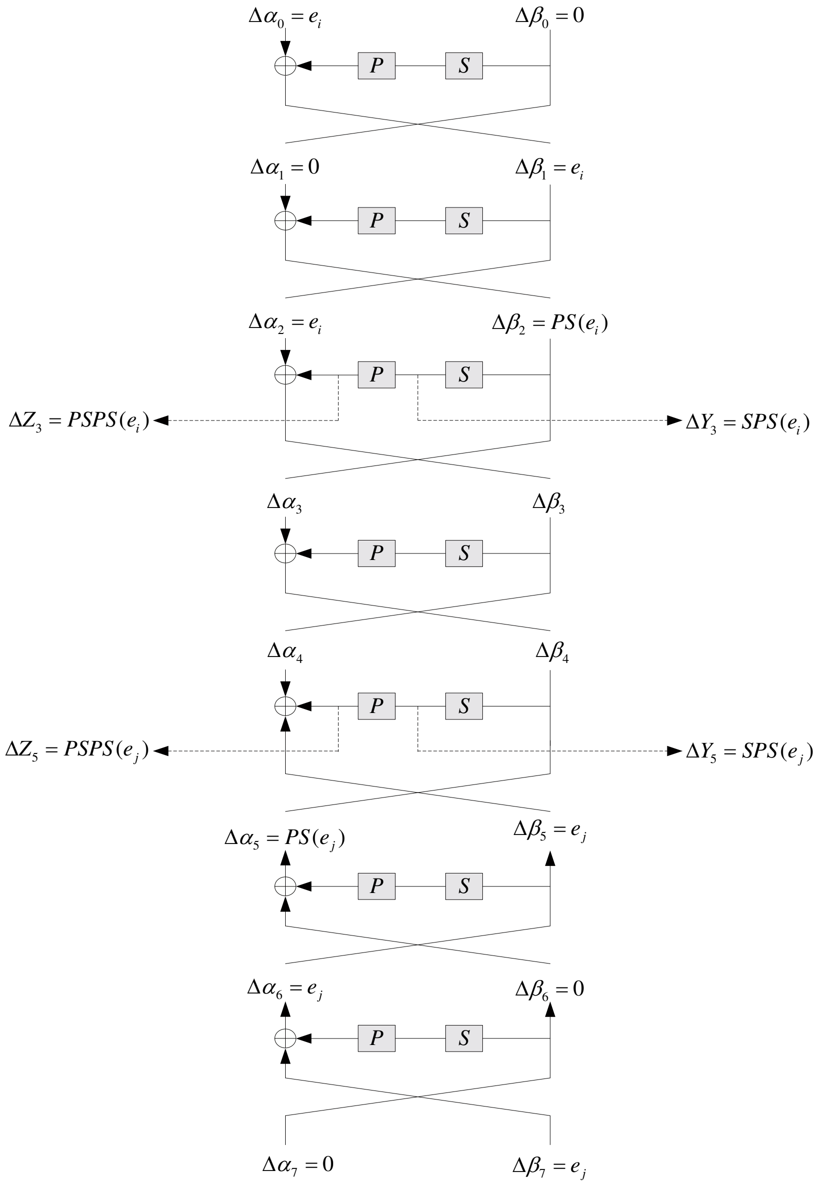


Fig. 4. 7-round Impossible Differential of Feistel-SP

and

$$\chi_t\left((S \circ P \circ S(e_i)) \oplus (S \circ P \circ S(e_j))\right) = 0,$$

which is a contradiction. Thus the above Proposition holds. □

Example 2. (7-Round Impossible Differentials of Camellia) By the definition of Camellia, we can determine $P = (p_{i,j})_{1\le i,j\le 8}$ and $P^{-1} = (q_{i,j})_{1\le i,j\le 8}$ as follows:

$$P = \begin{pmatrix} \underline{\mathbf{1}} & 0 & 1 & 1 & \underline{\mathbf{0}} & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & \underline{\mathbf{0}} & 1 & 1 & 1 & 1 & \underline{\mathbf{0}} & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \qquad P^{-1} = \begin{pmatrix} \underline{\mathbf{0}} & 1 & 1 & 1 & \underline{\mathbf{0}} & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & \underline{\mathbf{0}} & 1 & 1 & 1 & 1 & \underline{\mathbf{1}} & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Since $p_{1,1} = 1$, $p_{1,5} = q_{1,1} = q_{1,5} = 0$, $(e_1, 0) \to (0, e_5)$ is a 7-round impossible differential of Camellia; Similarly, $(e_2, 0) \to (0, e_7)$ is another 7-round impossible differential of Camellia since $q_{7,7} = 1$, $p_{7,2} = p_{7,7} = q_{7,2} = 0$.

3.3 Analysis of 8-Round Feistel Cipher with *SP* Structure

Let $\Delta(\alpha_0, \beta_0) = (e_i, 0)$, $\Delta(\alpha_8, \beta_8) = (0, e_j)$. Then from the encryption direction, we have (see Fig. 5):

$$\begin{aligned} &\Delta(\alpha_1, \beta_1) = (0, e_i), \\ &\Delta(\alpha_2, \beta_2) = (e_i, P \circ S(e_i)), \\ &\Delta(\alpha_3, \beta_3) = (P \circ S(e_i), e_i \oplus P \circ S \circ P \circ S(e_i)), \\ &\Delta Z_4 = P \circ S(e_i \oplus P \circ S \circ P \circ S(e_i)), \end{aligned}$$

and while analyzing from the decryption direction, we have

$$\begin{aligned} &\Delta(\alpha_7, \beta_7) = (e_j, 0), \\ &\Delta(\alpha_6, \beta_6) = (P \circ S(e_j), e_j), \\ &\Delta(\alpha_5, \beta_5) = (e_j \oplus P \circ S \circ P \circ S(e_j), P \circ S(e_j)). \end{aligned}$$

Since

$$\Delta\beta_2 \oplus \Delta Z_4 = \Delta\alpha_3 \oplus \Delta Z_4 = \Delta\beta_4 = \Delta\alpha_5,$$

the following equation holds

$$P \circ S(e_i) \oplus P \circ S(e_i \oplus P \circ S \circ P \circ S(e_i)) = e_j \oplus P \circ S \circ P \circ S(e_j),$$

which implies that

$$S(e_i \oplus P \circ S \circ P \circ S(e_i)) = P^{-1}(e_j) \oplus S \circ P \circ S(e_j) \oplus S(e_i).$$

Let $U_{i,j} = \{t | p_{t,j} = q_{t,j} = 0, t \neq i\} = \{t_1, \ldots, t_u\}$, thus for any $t \in U_{i,j}$,

$$\chi_t\left(P^{-1}(e_j) \oplus S \circ P \circ S(e_j) \oplus S(e_i)\right) = 0,$$

which tells that

$$\chi_t(e_i \oplus P \circ S \circ P \circ S(e_i)) = 0.$$

and now, we have the following Proposition:

Proposition 4. *For any i and j, let*

$$\begin{aligned} U_{i,j} &= \{t | p_{t,j} = q_{t,j} = 0, t \neq i\} = \{t_1, \dots, t_u\}, \\ V_i &= \{r | p_{r,i} = 1\} = \{r_1, \dots, r_v\}, \end{aligned}$$

and

$$M_{i,j} = (p_{t_a,r_b})_{u \times v} = (m_1, \dots, m_v).$$

If $U_{i,j} \neq \emptyset$, $V_i \neq \emptyset$, and there exists an s, $1 \leq s \leq v$, such that

$$\text{rank}\{m_1, \dots, m_v\} = \text{rank}\{\{m_1, \dots, m_v\} \setminus \{m_s\}\} + 1,$$

then $(e_i, 0) \to (0, e_j)$ is an 8-round impossible differential of $\mathcal{E}$.

Proof. Let $\eta = e_i \oplus P \circ S \circ P \circ S(e_i)$, $\lambda = S \circ P \circ S(e_i)$. Then

$$\chi_t(\lambda) = \begin{cases} 1 & \text{if } t \in V_i, \\ 0 & \text{if } t \notin V_i. \end{cases}$$

Accordingly, $\chi_t(\lambda) \neq 0$ holds if and only if when $\lambda_t \neq 0$. Thus

$$\eta = e_i \oplus (p_{r_1}, \dots, p_{r_v})(\lambda_{r_1}, \dots, \lambda_{r_v})^{\mathrm{T}},$$

where $r_1 < \cdots < r_v$, $r_k \in V_i (1 \leq k \leq v)$ and p_{r_k} is the r_k-th column of P.

By the definition of $U_{i,j}$, we have

$$\eta_t = 0 \qquad \text{if} \qquad t \in U_{i,j},$$

thus

$$\begin{pmatrix} p_{t_1,r_1} & \cdots & p_{t_1,r_v} \\ \vdots & & \vdots \\ p_{t_u,r_1} & \cdots & p_{t_u,r_v} \end{pmatrix} \begin{pmatrix} \lambda_{r_1} \\ \vdots \\ \lambda_{r_v} \end{pmatrix} \triangleq (M_{i,j})_{u,v} \tilde{\lambda} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

where $t_k \in U_{i,j}$, $r_k \in V_i$ and $\lambda_{r_k} \neq 0$.

The above equation can be described as

$$(m_1, \dots, m_{s-1}, m_{s+1}, \dots, m_v)(\lambda_1, \dots, \dots, \lambda_v)^{\mathrm{T}} = \lambda_s m_s,$$

from linear algebra, the equation has solutions if and only if

$$\text{rank}\{\{m_1, \dots, m_v\} \setminus \{m_s\}\} = \text{rank}\{m_1, \dots, \lambda_s m_s, \dots, m_v\}.$$

Since $\text{rank}\{m_1, \ldots, m_v\} = \text{rank}\{\{m_1, \ldots, m_v\} \setminus \{m_s\}\} + 1$, if $(M_{i,j})_{u,v}\tilde{\lambda} = 0$ has a solution $\tilde{\lambda} = (\lambda_1, \ldots, \lambda_v)$, λ_s must be 0 which is a contradiction. □

For most cases, especially when $n = 4$ or $n = 8$, $|U_{i,j}| = u \leq 2$. According to Proposition 4, the case that $u = 1$ and $u = 2$ can be characterized as follows:

Proposition 5. *Let $U_{i,j}$ and V_i defined as in Proposition 4, and*

$$M_{i,j} = (p_{t_a,r_b})_{u \times v} = \begin{pmatrix} l_1 \\ \vdots \\ l_u \end{pmatrix}.$$

(1) *If $u = 1$ and $w(l_1) = 1$, then $(e_i, 0) \to (0, e_j)$ is an 8-round impossible differential of $\mathcal{E}$.*
(2) *If $u = 2$ and $w(l_1 \oplus l_2) = 1$, then $(e_i, 0) \to (0, e_j)$ is an 8-round impossible differential of $\mathcal{E}$.*

Example 3. (8-Round Impossible Differentials of Camellia) We verify $(e_2, 0) \to (0, e_2)$ is an 8-round impossible differential. From the linear transformation of Camellia we have,

$$P = \begin{pmatrix} 1&0&1&1&0&1&1&1 \\ 1&1&0&1&1&0&1&1 \\ 1&1&1&0&1&1&0&1 \\ 0&1&1&1&1&1&1&0 \\ 1&1&0&0&0&1&1&1 \\ 0&1&1&0&1&0&1&1 \\ 0&\underline{\mathbf{0}}&\underline{\mathbf{1}}&\underline{\mathbf{1}}&\underline{\mathbf{1}}&\underline{\mathbf{1}}&0&1 \\ 1&\underline{\mathbf{0}}&\underline{\mathbf{0}}&\underline{\mathbf{1}}&\underline{\mathbf{1}}&\underline{\mathbf{1}}&1&0 \end{pmatrix} \qquad P^{-1} = \begin{pmatrix} 0&1&1&1&0&1&1&1 \\ 1&0&1&1&1&0&1&1 \\ 1&1&0&1&1&1&0&1 \\ 1&1&1&0&1&1&1&0 \\ 1&1&0&0&1&0&1&1 \\ 0&1&1&0&1&1&0&1 \\ 0&\underline{\mathbf{0}}&1&1&1&1&1&0 \\ 1&\underline{\mathbf{0}}&0&1&0&1&1&1 \end{pmatrix}.$$

Since $p_{7,2} = q_{7,2} = 0$, $p_{8,2} = q_{8,2} = 0$,

$$U_{2,2} = \{7, 8\}.$$

Since $p_{2,2} = p_{3,2} = p_{4,2} = p_{5,2} = p_{6,2} = 1$, we have

$$V_2 = \{2, 3, 4, 5, 6\},$$

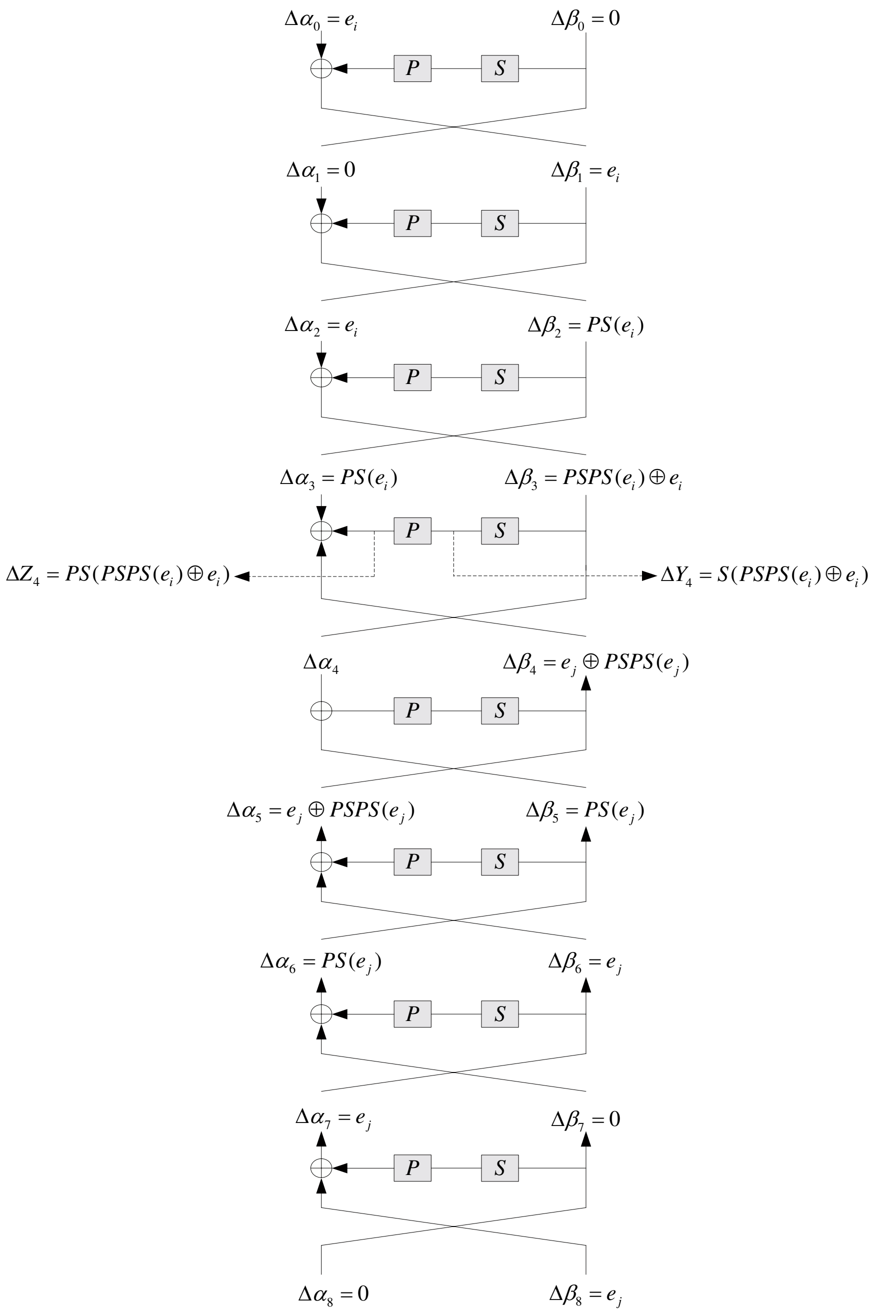
Thus $M_{2,2}$ is a sub-matrix of P

$$M_{2,2} = \begin{pmatrix} 0&1&1&1&1 \\ 0&0&1&1&1 \end{pmatrix}.$$

Since,

$$2 = \text{rank} M_{2,2} = \text{rank}\{M_{2,2} \setminus \{l_2\}\} + 1,$$

we know that $(e_2, 0) \to (0, e_2)$ is an 8-round impossible differential of Camellia which is consistent with [11].



$\Delta\alpha_0 = e_i$
$\Delta\beta_0 = 0$
P
S
$\Delta\alpha_1 = 0$
$\Delta\beta_1 = e_i$
$\Delta\alpha_2 = e_i$
$\Delta\beta_2 = PS(e_i)$
$\Delta\alpha_3 = PS(e_i)$
$\Delta\beta_3 = PSPS(e_i) \oplus e_i$
$\Delta Z_4 = PS(PSPS(e_i) \oplus e_i)$
$\Delta Y_4 = S(PSPS(e_i) \oplus e_i)$
$\Delta\alpha_4$
$\Delta\beta_4 = e_j \oplus PSPS(e_j)$
$\Delta\alpha_5 = e_j \oplus PSPS(e_j)$
$\Delta\beta_5 = PS(e_j)$
$\Delta\alpha_6 = PS(e_j)$
$\Delta\beta_6 = e_j$
$\Delta\alpha_7 = e_j$
$\Delta\beta_7 = 0$
$\Delta\alpha_8 = 0$
$\Delta\beta_8 = e_j$

Fig. 5. 8-round Impossible Differential of Feistel-SP

Example 4. (8-Round Impossible Differentials of SNAKE(2)) SNAKE(2) is equivalent to Feistel structure with SP round function by adding a P^{-1} in the beginning and a P in the end. By the definition of SNAKE(2), we describe P and P^{-1} as follows:

$$P = \begin{pmatrix} 1\,1\,1\,1 \\ 1\,0\,0\,\underline{\mathbf{0}} \\ 1\,1\,0\,0 \\ 1\,1\,1\,0 \end{pmatrix} \qquad P^{-1} = \begin{pmatrix} 0\,1\,0\,0 \\ 0\,1\,1\,\underline{\mathbf{0}} \\ 0\,0\,1\,1 \\ 1\,0\,0\,1 \end{pmatrix}.$$

Hence,

$$U_{4,4} = \{2\},\ V_4 = \{1\},\ \text{and}\ M_{4,4} = \{1\}.$$

Since $w(l_1) = 1$, $(e_4, 0) \to (0, e_4)$ is an 8-round impossible differential of SP part of SNAKE(2). Therefore, $(P(e_4), 0) \to (0, P(e_4))$ is an 8-round impossible differential of SNAKE(2).

4 Analysis of 6-Round Feistel Cipher with *SPS* Structure

By using the same techniques that are used in analyzing 8-round Feistel ciphers with SP round functions, a characterization for the existence of 6-round of Feistel cipher with SPS round functions (Notice that Feistel cipher with this structure is denoted by $\mathcal{D}$) can be given as follows, and the details of the proof are omitted.

Proposition 6. *For any i and j, let*

$$\begin{aligned} U_{i,j} &= \{t | p_{t,j} = 0, t \neq i\} = \{t_1, \ldots, t_u\}, \\ V_i\ &= \{r | p_{r,i} = 1\} = \{r_1, \ldots, r_v\}, \end{aligned}$$

and

$$M_{i,j} = (p_{t_a, r_b})_{u \times v} = (m_1, \ldots, m_v).$$

If $U_{i,j} \neq \emptyset$, $V_i \neq \emptyset$, and there exists an s, $1 \leq s \leq v$, such that

$$\text{rank}\{m_1, \ldots, m_v\} = \text{rank}\{\{m_1, \ldots, m_v\} \setminus \{m_s\}\} + 1,$$

then $(e_i, 0) \to (0, e_j)$ is a 6-round impossible differential of $\mathcal{D}$.

Proposition 7. *Let $U_{i,j}$ and V_i be defined as in Proposition 6, and*

$$M_{i,j} = (p_{t_a, r_b})_{u \times v} = \begin{pmatrix} l_1 \\ \vdots \\ l_u \end{pmatrix}.$$

(1) *If $u = 1$ and $w(l_1) = 1$, then $(e_i, 0) \to (0, e_j)$ is a 6-round impossible differential of $\mathcal{D}$.*

(2) *If $u = 2$ and $w(l_1 \oplus l_2) = 1$, then $(e_i, 0) \to (0, e_j)$ is a 6-round impossible differential of $\mathcal{D}$.*

Example 5. (6-Round Impossible Differentials of E2) Since the permutation BRL after the 2nd nonlinear transformation layer of E2 is a byte-transposition, the structure is equivalent to an SPS structure, where the linear transformation $P^{'}$ is defined as:

$$P^{'} = BRL \circ P = \begin{pmatrix} 1\,0\,1\,1\,0\,1\,1\,1 \\ 1\,1\,0\,1\,1\,0\,1\,1 \\ 1\,1\,1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,1\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,1\,1\,0\,0\,1 \\ 0\,1\,1\,1\,1\,1\,1\,0 \end{pmatrix}.$$

We have

$$U_{1,3} = \{2, 4\}, \quad V_1 = \{1, 2, 3, 4, 5, 7\},$$

hence,

$$M_{1,3} = \begin{pmatrix} 1\,1\,0\,1\,1\,1 \\ 1\,1\,0\,1\,1\,0 \end{pmatrix}.$$

Since,

$$w(l_1 \oplus l_2) = 1,$$

$(e_1, 0) \to (0, e_3)$ is a 6-round impossible differential of E2, whereas only 5-round impossible differentials were previously known[3].

5 Conclusion

In this paper, we propose impossible differential cryptanalysis on Feistel ciphers with SP and SPS round functions. Both 6/7/8-round impossible differentials of Feistel cipher with SP round functions and 6-round impossible differential of Feistel cipher with SPS round functions can be judged by verifying some properties of linear transformations. The former result that 5-round impossible differential exists when round function is bijective is improved. Since we know a lot about Feistel cipher against impossible differential cryptanalysis, the properties presented in this paper should be considered when designing a block cipher.

Acknowledgements

The authors would like to thank Vincent Rijmen and Ruilin Li for their helpful comments and corrections that improved the quality of the paper. We are also grateful to the anonymous referees for their valuable remarks.

This work is supported by the Natural Science Foundation of China (No: 60803156) and the open research fund of State Key Laboratory of Information Security (No: 01– 07).

References

1. NTT-Nippon Telegraph and Telephone Corporation: E2: Efficient Encryption Algorithm, http://info.isl.ntt.co.jp/e2
2. Lee, C., Cha, Y.: The Block Cipher: SNAKE with Provable Resistance against DC and LC attacks. In: JW-ISC 1997, pp. 3–17 (1997)
3. Aoki, K., Kanda, M.: Search for Impossible Differential of E2, http://csrc.nist.gov/encryption/aes/round1/comment
4. Feistel, H.: Cryptography and Data Security. Scientific American 228(5), 15–23 (1973)
5. Aoki, K., Ichikawa, T., Kanda, M., et al.: Specification of Camellia — a 128–bit Block Cipher. In: Stinson, D.B., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 183–191. Springer, Heidelberg (2001)
6. Duo, L., Li, C., Feng, K.: New Observation on Camellia. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 51–64. Springer, Heidelberg (2006)
7. Knudsen, L.: DEAL — A 128-bit Block Cipher. Technical Report 151, Department of Informatics, University of Bergen, Bergen, Norway (1998)
8. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
9. Biham, E., Biryukov, A., Shamir, A.: Miss in the Middle Attacks on IDEA and Khufu. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 124–138. Springer, Heidelberg (1999)
10. Sugita, M., Kobara, K., Imai, H.: Security of Reduced Version of the Block Cipher Camellia against Truncated and Impossible Differential Cryptanalysis. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 193–207. Springer, Heidelberg (2001)
11. Wu, W., Zhang, W., Feng, D.: Impossible Differential Cryptanalysis of Reduced-Round ARIA and Camellia. Journal of Computer Science and Technology 22(3), 449–456 (2007)
12. Wu, W., Zhang, L., Zhang, W.: Improved Impossible Differential Cryptanalysis of Reduced–Round Camellia. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 442–456. Springer, Heidelberg (2009)
13. Lu, J., Kim, J., Keller, N., et al.: Improving the Efficiency of Impossible Differential Cryptanalysis of Reduced Camellia and MISTY1. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 370–386. Springer, Heidelberg (2008)
14. Lu, J., Dunkelman, O., Keller, N., et al.: New Impossible Differential Attacks on AES. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 279–293. Springer, Heidelberg (2008)
15. Dunkelman, O., Keller, N.: An Improved Impossible Differential Attack on MISTY1. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 441–454. Springer, Heidelberg (2008)
16. Mala, H., Shakiba, M., Dakhilalian, M., Bagherikaram, G.: New Results on Impossible Differential Cryptanalysis of Reduced-Round Camellia–128. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 281–294. Springer, Heidelberg (2009)
17. Kim, J., Hong, S., Sung, J., Lee, S., Lim, J.: Impossible Differential Cryptanalysis for Block Cipher Structures. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 82–96. Springer, Heidelberg (2003)

A Appendix

A.1 Brief Description of Camellia

Camellia is a Feistel cipher with SP round function and has 18 rounds (for 128-bit keys) or 24 rounds (for 192/256-bit keys). The FL/FL^{-1} function layer is inserted at every 6 rounds. In this paper, we consider Camellia without FL/FL^{-1} function layer. The nonlinear layer S and linear transformation P in the round function of Camellia are represented as follows. For more details, we refer to [5].

$$S : \mathbb{F}_{2^8}^8 \rightarrow \mathbb{F}_{2^8}^8 : (x_1, x_2, \ldots, x_8) \rightarrow (y_1, y_2, \ldots, y_8)$$
$$y_1 = s_1(x_1),\ y_2 = s_2(x_2),\ y_3 = s_3(x_3),\ y_4 = s_4(x_4),$$
$$y_5 = s_2(x_5),\ y_6 = s_3(x_6),\ y_7 = s_4(x_7),\ y_8 = s_1(x_8),$$

where s_1, s_2, s_3 and s_4 are 8×8 nonlinear transformations (s-boxes).

$$P : \mathbb{F}_{2^8}^8 \rightarrow \mathbb{F}_{2^8}^8\text{:}\ (y_1, y_2, \ldots, y_8) \rightarrow P(y_1, y_2, \ldots, y_8)$$

$$P = \begin{pmatrix} 1&0&1&1&0&1&1&1 \\ 1&1&0&1&1&0&1&1 \\ 1&1&1&0&1&1&0&1 \\ 0&1&1&1&1&1&1&0 \\ 1&1&0&0&0&1&1&1 \\ 0&1&1&0&1&0&1&1 \\ 0&0&1&1&1&1&0&1 \\ 1&0&0&1&1&1&1&0 \end{pmatrix}$$

A.2 Brief Description of SNAKE(2)

SNAKE(1) and SNAKE(2) are Feistel ciphers proposed by Lee and Cha at JW-ISC'97 [2] and this paper concentrates on SNAKE(2) only. Although it employs a PS round function, according to [6], SNAKE(2) is equivalent to a Feistel cipher with SP round function by adding a P^{-1} transformation before the first round and a P transformation after the last round. The nonlinear layer S and linear transformation P in the round function of SNAKE(2) are represented as follows.

$$S : \mathbb{F}_{2^8}^4 \rightarrow \mathbb{F}_{2^8}^4 : (x_1, x_2, x_3, x_4) \rightarrow (y_1, y_2, y_3, y_4)$$
$$y_1 = s(x_1),\ y_2 = s(x_2),\ y_3 = s(x_3),\ y_4 = s(x_4),$$

where s is an 8×8 nonlinear transformation.

$$P : \mathbb{F}_{2^8}^4 \rightarrow \mathbb{F}_{2^8}^4\text{:}\ (y_1, y_2, y_3, y_4) \rightarrow P(y_1, y_2, y_3, y_4)$$

$$P = \begin{pmatrix} 1&1&1&1 \\ 1&0&0&0 \\ 1&1&0&0 \\ 1&1&1&0 \end{pmatrix}$$

A.3 Brief Description of E2

E2, designed by NTT, is a candidate of AES [1]. It employs Feistel structure with an SPS round function. There is also an initial transformation in the beginning and a final transformation in the end. Another permutation BRL is placed after the 2nd non linear transformation layer. The non-linear layer S and linear transformation P and BRL in the round function of E2 are represented as follows.

$$\begin{aligned} S : \mathbb{F}_{2^8}^8 \rightarrow \mathbb{F}_{2^8}^8 : (x_1, x_2, \ldots, x_8) \rightarrow (y_1, y_2, \ldots, y_8) \\ y_1 = s(x_1),\, y_2 = s(x_2),\, y_3 = s(x_3),\, y_4 = s(x_4), \\ y_5 = s(x_5),\, y_6 = s(x_6),\, y_7 = s(x_7),\, y_8 = s(x_8), \end{aligned}$$

where s is an 8×8 nonlinear transformation.

$$P : \mathbb{F}_{2^8}^8 \rightarrow \mathbb{F}_{2^8}^8 \colon (y_1, y_2, \ldots, y_8) \rightarrow P(y_1, y_2, \ldots, y_8)$$

$$P = \begin{pmatrix} 0\,1\,1\,1\,1\,1\,1\,0 \\ 1\,0\,1\,1\,0\,1\,1\,1 \\ 1\,1\,0\,1\,1\,0\,1\,1 \\ 1\,1\,1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,1\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,1\,0 \\ 0\,1\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,1\,1\,0\,0\,1 \end{pmatrix}$$

$$BRL : \mathbb{F}_{2^8}^8 \rightarrow \mathbb{F}_{2^8}^8 \colon (y_1, y_2, \ldots, y_8) \rightarrow (y_2, y_3, \ldots, y_8, y_1).$$

Multi-trail Statistical Saturation Attacks

Baudoin Collard[*] and Francois-Xavier Standaert[**]

UCL Crypto Group, Microelectronics Laboratory, Université catholique de Louvain
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium
baudoin.collard, fstandae@uclouvain.be

Abstract. Statistical Saturation Attacks have been introduced and applied to the block cipher PRESENT at CT-RSA 2009. In this paper, we consider their natural extensions. First, we investigate the existence of better trails than the one used in the former attack. For this purpose, we provide a theoretical evaluation of the trail distributions using probability transition matrices. Since the exhaustive evaluation of all possible distributions turned out to be computationally hard, we additionally provide a heuristic branch-and-bound algorithm that allows us to generate a large number of good trails. These tools confirm that the trail of CT-RSA 2009 was among the best possible ones, but also suggest that numerous other trails have similar properties. As a consequence, we investigate the use of multiple trails and show that it allows significant improvements of the previous cryptanalysis attempts against PRESENT. Estimated complexities indicate that PRESENT-80 is safe against key recovery, by a small security margin. We also discuss the impact of multiple trails for the security of the full PRESENT-128. We finally put forward a "statistical hull" effect that makes the precise theoretical analysis of our results difficult, when the number of block cipher rounds increases.

Introduction

PRESENT is a block cipher presented at CHES 2007, that was designed for small embedded applications [3]. It has a Substitution Permutation Network architecture, with a 64-bit block size and 31 rounds. The same 4-bit S-box is applied 16 times in parallel in each round. The designers have proposed two key sizes: 80 and 128 bits. Due to its simple and elegant structure, it has been the focus of different cryptanalysis attempts. In [23], the author presented a first attack against PRESENT, using differential cryptanalysis. It applies to 16 block cipher rounds and requires the whole codebook and a time complexity of 2^{65}. In 2009, Ozen et al. proposed a related key rectangle attack against up to 17 rounds of PRESENT, with a time complexity of 2^{104} and 2^{63} chosen plaintexts [20]. Two papers exploit the linear cryptanalysis and target 26 rounds, respectively by taking advantage of the linear hull effect [17] and multiple approximations [10]. These attacks require the whole codebook. Finally, [19] combines linear cryptanalysis and weak keys and targets up to 28 rounds in this context.

[*] Work supported by the project Nanotic-Cosmos of the Walloon Region.

[**] Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

In this paper, we pay a particular attention to a Statistical Saturation Attack that was specifically devised for the cryptanalysis of PRESENT (although it could apply to other ciphers). It exploits the weak diffusion of certain bits (called the trail) during the encryption process, when some of the plaintext bits are fixed. This property did lead to an estimated attack against up to 24 rounds, using approximately 2^{60} chosen plaintexts. As detailed in [6], the main limitation when trying to extend this technique towards more rounds is the data complexity that exceeds the complete codebook. We consequently investigated the tracks that could be used to get rid of this limitation. Our contributions are threefold.

First, we provide tools allowing one to approximate the diffusion in a trail, using Markov chains. We exhibit that, besides the iterative trail proposed in [6], there exists many other trails with a similarly weak diffusion. We also propose a heuristic branch-and-bound algorithm in order to generate them efficiently. We finally confirm our theoretical analysis with experiments that can break up to 16 rounds PRESENT. Then, in a second part of the paper, we investigate the exploitation of these multiple trails. We show that they can be used to trade data complexity for time complexity. As a result, we discuss the possibility to mount a key-recovery attack against the full 31-round PRESENT-128, using the complete codebook, and a time complexity below 2^{128} memory accesses. We put forward that such an attack could be possible under certain (optimistic) conditions of independence for the trails - the exact evaluation of these conditions being an important scope for further research. We note that such an attack would anyway be of theoretical interest only. In particular, the authors in [3] clearly suggest PRESENT-80 for their target applications (rather than PRESENT-128). However, these results question the number of rounds for PRESENT-128 and highlights that they could have been increased over those for PRESENT-80. Finally, in a third part of the paper, we put forward a "statistical hull effect", *i.e.* a counterpart of the linear hull effect in linear cryptanalysis. We discuss its impact for the theoretical analysis of our estimated attack complexities.

1 The Statistical Saturation Attack

1.1 Principle of the Attack

The Statistical Saturation Attack, originally described in [6], takes advantage of a weakness in the diffusion layer of PRESENT. For the S-boxes 5, 6, 9 and 10 (called the *active* S-boxes), only 8 out of 16 input bits are directed to other S-boxes. Figure 1 illustrates this observation (note that there exists many other examples of weak diffusion in the permutation). Consequently, if we fix the 16 bits at the input of the active S-boxes, then 8 bits will be known at the very same input for the next round. We can iteratively repeat this process round by round and observe a non-uniform behavior at the output of the active S-boxes.

Thanks to this non-uniform behavior, 16 bits of the last subkey can be recovered as follows. We first generate a large number of plaintexts with 8 fixed bits. The plaintexts are encrypted using r-rounds PRESENT and the distribution of the ciphertexts are recorded for the 16 bits at the output of the 4 active S-boxes

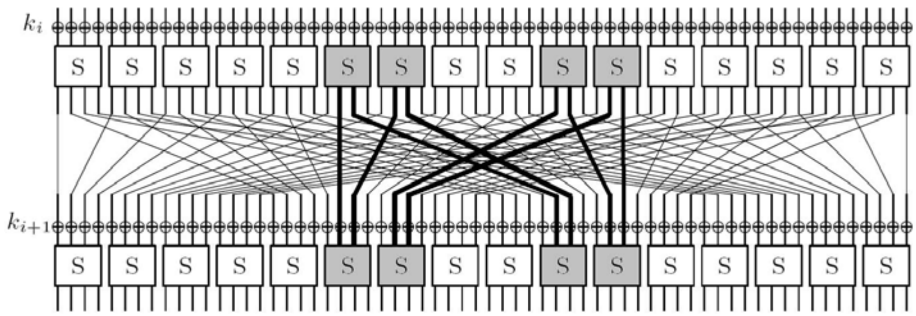


Fig. 1. Permutation layer of PRESENT: bold lines show the weak diffusion

in the last round. Given this experimental distribution, it is possible to compute the output distribution of the target 8-bit trail one round before, using a partial decryption process. For one key guess, the evaluation of such an $r-1$-round distribution requires 2^{16} computations. Hence the total time complexity for all the key guesses equals $2^{16} * 2^{16} = 2^{32}$. Additionally using an FFT-based trick similar to the technique presented in [4], this complexity can be decreased to $16 \cdot 2^{16} \cdot 2^{8}$. For the correct key guess, the experimental 8-bit distribution in the penultimate round is expected to be more non-uniform than for any other guess. This is because decrypting with a wrong guess is expected to have the same effect as encrypting one more round. We can thus hope to distinguish the correct key from the wrong ones by computing the distance between a partially decrypted distribution and the uniform distribution. If the attack works properly, the distribution with the highest distance should correspond to the correct key.

1.2 Extensions of the Attack

In [6], the authors propose 3 extensions to improve the cryptanalysis:

(ext. 1) Increase the fixed part in the plaintexts. One can easily gain one round in the attack by simply fixing the 16 bits of plaintext corresponding to the 4 active input S-boxes of the trail. This way, the 8-bit trail in the second round is also fixed and the diffusion is postponed by one round. By fixing 32 bits out of 64 (corresponding to S-boxes 4-5-6-7-8-9-10-11), one can similarly extend the attack by 2 rounds. However, we are then limited in the generation of at most 2^{32} texts. This limitation may be mitigated with the following extension.

(ext. 2) Use multiple fixed plaintext values. The same analysis can be performed multiple times, using different values for the 8-bit (or 16- or 32-bit) fixed part of the plaintexts and then combining the results (*e.g.* taking the sum of the uniform *vs.* measured distances corresponding to the different fixed plaintexts). This allows exploiting more texts and moving to a known-plaintext context. The resulting attack is similar to multiple linear cryptanalysis: each fixed part of the plaintext can be seen as analogous to an additional approximation in [2,11].

(ext. 3) Partial decryption of two rounds instead of one. In this case, 8 S-boxes are active in the last round and 4 S-boxes are active in the penultimate round. As detailed in [7] and illustrated in Figure 4, one can perform two independent partial decryptions in parallel, in order to decrease the time complexity of the attack down to $2 \cdot (16 \cdot 2^{16} \cdot 2^8) \cdot (8 \cdot 2^8 \cdot 2^4) = 2^{44}$ elementary operations.

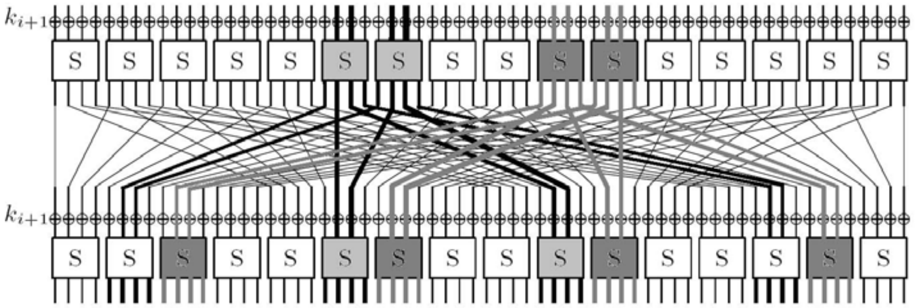


Fig. 2. Practical trails for 2-round partial decryption in PRESENT with reduced time complexity. The two independent trails are shown in different shades of gray.

2 Evaluating the Trail Distributions with Markov Chains

As a matter of fact, the previous attack essentially exploits the property that it is possible to evaluate the distribution of a subset of output bits given the distribution of a subset of input bits for one round of PRESENT. Minier and Gilbert use a similar technique in their attack against Crypton [16]. In [6], the authors exploited an iterative trail with 8 active bits in 4 active S-boxes in each round. However, as already mentioned, this is not the only possible trail and an interesting problem is to determine if there are other trails leading to better attacks. In this section, we show how to evaluate the distribution of a trail going through several rounds of PRESENT. For this purpose, we characterize such a trail by a matrix containing the transition probabilities between the inputs and outputs. Additionally, we rely on the assumption that the bits that are not part of the trail are uniformly distributed. This assumption as well as the Markov chains that we exploit in the following were also used by Vaudenay in his paper on χ^2 cryptanalysis [21]. As will be discussed in Section 5, this is becoming incorrect as the number of rounds in the trail increases. But as the next section will show, this assumption is required in order to limit the computational cost of our estimations to tractable values, when comparing different trails.

2.1 Transition Matrix for an S-Box

Let us consider an active S-box with size $n*n$, and suppose that the trail includes i active bits among n in input and j active bits in output. Consequently, there are 2^i possible inputs and 2^j possible outputs, and the size of the transition matrix is $2^i * 2^j$. This matrix is constructed in the following way:

- Initialize a matrix of size $2^i * 2^j$ and fill it with zeros.
- For every possible S-box input value, extract a masked i-bit input and the j-bit output, and increment the matrix in the corresponding position.
- Multiply the matrix by $2^i/2^j$ for normalization.

By construction, any transition matrix has the properties that the sum over any row is equal to one. For example, the iterative trail represented in Figure 1 uses the same transition matrix for each active S-box:

$$A = \begin{pmatrix} 0 & 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0 & 0.5 \\ 0.5 & 0 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 & 0 \end{pmatrix}$$

In this case, the matrix is square, but it is not mandatory as it depends on the number of active input/output bits. The interpretation of the transition matrix is easy: each row represents a possible value for the input and the column represents the probability of transition to a particular output value given the input.

2.2 Transition Matrix for the Permutation Layer

For a permutation layer like the one used in PRESENT, the number of active bits in output is equal to the number k of active bits in input. Consequently, the matrix is square with size $2^k * 2^k$. Moreover, at each input value in the trail corresponds one and only one output value and thus the transition matrix contains only zeros and ones (i.e. it is a particular instance of permutation matrix).

2.3 Transition Matrix for the Subkey Addition

The effect of a XOR between the input bits in the trail and unknown key bits is similar to a permutation. To each value in the trail before the key addition corresponds only one output value after the key addition. Hence, the transition matrix for the subkey addition is also a permutation matrix. However, unlike the transition matrix for the permutation layer, this transition matrix does not increase the diffusion in the trail. Intuitively, this is because the key addition does not mix the active bits coming from different active S-boxes as with the permutation layer. Mathematically, this corresponds to the property that the transition matrix can be decomposed into a Kronecker product of small submatrices. Consequently, given the assumption of uniform distribution for the bits that are not part of the trail, different subkeys have different output distributions in the trail, but they present an identical non-uniform behavior. Hence, it is sound to compute the distribution of a trail independently of the keys.

2.4 Composition of Transition Matrices

If several S-boxes are active in parallel in a trail, the overall transition matrix is given by the Kronecker product of the transition matrix related to each S-box. The transition matrix of a round can then be computed as the matrix product of the transition matrices for the S-box and permutation layers. Thereafter, given the transition matrix for a complete trail, the output distribution can be directly evaluated as the the vector-matrix product of the input distribution and the transition matrix. The main drawback of this method is that it requires to compute a matrix product with matrices of size $2^n * 2^n$ where n is the number of active bits involved at any point in the trail during the encryption process.

2.5 Practical Example

We illustrate this technique using the iterative trail of Figure 1. This trail is composed of 4 active S-boxes at each round, with 2 bits out of 4 active in each S-box. As there are 8 active bits at each round, the full transition matrix has a size of $2^8 * 2^8$. It is computed in the following way: The matrix transition for the 4 parallel S-boxes is computed as: $A^4 = A \otimes A \otimes A \otimes A$, where $\otimes$ is the symbol for the Kronecker product. The matrix for one full round is then given by $R = A^4 \cdot P$ (where P is the transition matrix for the permution on 8 bits). The transition matrix after n rounds is then $R^n = \underbrace{R \cdot R .... \cdot R}_{n\ times}$. Given a vector d_{in} of size $(1 * 2^8)$ describing the distribution of the 8-bits active bits in input, the distribution of the output active bits is finally given by $d_{out} = d_{in} \cdot R^n$.

3 Heuristic Branch-and-Bound for Trail Search

As detailed in the previous section, the evaluation of the distribution for a single trail can be computationally intensive if this trail involves a lot of active bits. This was not an issue for the attack in [6], but may become the limiting factor for other trails (or other ciphers). In order to mitigate this limitation, we now present a heuristic algorithm based on the branch-and-bound proposed by Matsui for linear approximation search in [15]. The goal of this heuristic is to perform a pre-selection of “interesting trails” that minimizes the number of active S-boxes (so that the treatment of the previous section can still be applied) while trying to limit the diffusion based on a simpler criteria than a probabilistic distance.

3.1 Description of the Algorithm

The basic principle of the heuristic is to maximize the ratio between the number of active bits and the number of active S-boxes in a trail. While this criteria does not ensure finding the best trail distributions, it is extremely fast to evaluate and to integrate into the execution of a branch-and-bound algorithm. As will be shown later in the section, it also provides reasonably good results. The justification is the following: even though all the trails with low diffusion are not necessarily good trails (because of the influence of the transition matrices for the S-boxes), all the good trails must have low diffusion in their permutations layers. Consequently, a good strategy is to first generate a large number of trail candidates with a branch-and-bound heuristic, then to compute the full transition matrices for each of these candidates and to select the best ones only.

In order to speed-up the execution of the algorithm, we used a similar implementation technique as presented in [4]. That is, we start by exhaustively counting the couples (input, output) of the permutation for which the number of active S-boxes is low. Such couples are called *permutation candidates* and are entirely defined by the number of active input and output S-boxes, the position of these S-boxes and their corresponding mask value. The permutation candidates are then stored in a database (a hash table) instead of being generated on-the-fly during each branching phase. All the candidates having the same active output S-boxes are stored in the same list. Once the database is created, we launch the actual trail search: a trail on r rounds can be obtained by the concatenation of r permutation candidates, if the positions of the active S-boxes at the exit of a permutation candidate correspond to those of the active S-boxes at the input of the next candidate. These constraints are easily checked, as the candidates are picked up in the database according to the position of their active output S-boxes. The objective function that we need to maximize is the average ratio between the number of active bits and the number of active S-boxes in the trail. Note finally that we pile up the candidates starting with the last round, then going down gradually until the first round, in order to benefit from the knowledge of the best ratio in each phase of the branch-and-bound.

3.2 Results

As an illustration, we generated 1000 trails with maximum 5 active S-boxes in each round and computed the theoretical data complexity as in [6], for the distinguishers based on each of these trails. That is, following the analysis of Baignère et al. [1], we estimate the data complexity as proportional to the inverse of the Euclidean distance between the distributions evaluated in Section 2 and a uniform distribution. The results in the figure show that after 15 rounds, the data complexity varies between 2^{50} and 2^{66}, according to the trail. The original trail

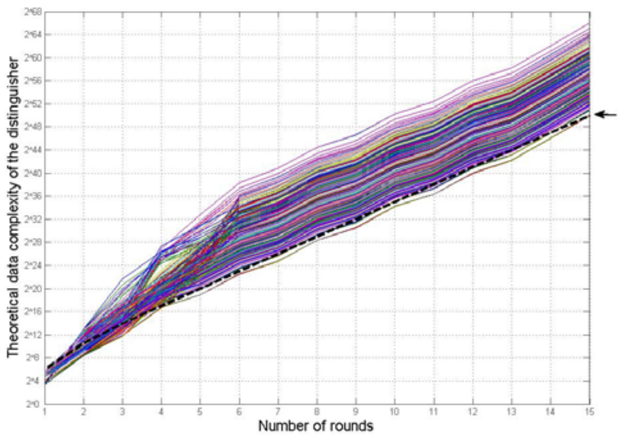


Fig. 3. Theoretical data complexity for distinguishers based on 1000 different trails

of Figure 1 is marked with an arrow and is among the best ones (see Figure 3). Note that by increasing the number of trails generated by the branch-and-bound (beyond 1000), we can easily produce very large amounts of trails with good theoretical data complexities. As will be detailed in Section 5, the amount of such trails increases exponentially with the number of rounds.

3.3 Experimental Validation of the Estimated Data Complexity

The estimations in the previous section indicate that the data complexity increases by approximately 2^3 for every additional round. As these estimations rely on the assumptions needed to evaluate the distributions in Section 2, we confirmed these predictions experimentally, in order to verify that our assumptions hold to a sufficient extent. For this purpose, we complemented the experiments in [6] and attacked up to 15 rounds PRESENT with 2^{32} texts, using a 2-round partial decryption process. Figure 4 illustrates that we gain one round compared to the original attack of CT-RSA, and confirms the theoretical expectations. Note that the two-round decryption also allows a significantly increased gain (because there are more key bits guessed in the experiment).

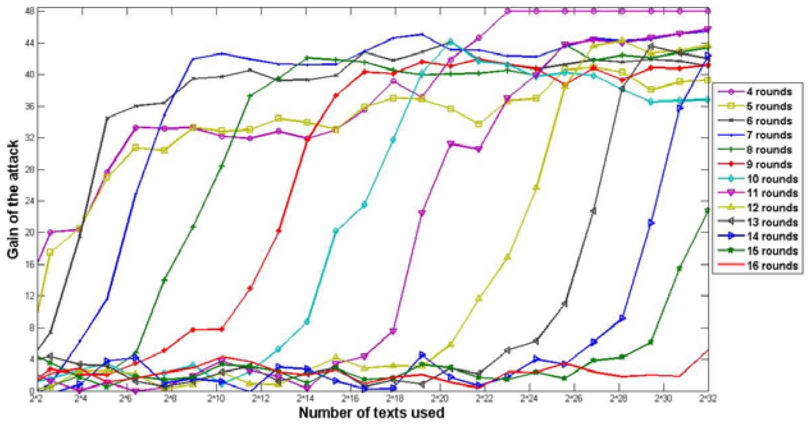


Fig. 4. Average gain of 6 attacks against 4 to 16 rounds PRESENT, using up to 2^{32} texts (these attacks exploit **ext. 1**, with 32 fixed bits and **ext. 3**)

4 Multiple Trails

The previous section shows that the simple trail of Figure 1 is among the best ones to perform a Statistical Saturation Attack against PRESENT. On the other hand, we also observe that a large number of trails perform similarly good in theory. Hence, a natural idea is to investigate the use of multiple trails, as can be done in linear cryptanalysis with multiple approximations [2]. In the following, we consequently consider two questions. First, we study the possibility to exploit several trails with different input masks and the same output mask, in order to increase the gain of the attack. Our experiments suggest that this technique yields good results and allows improving the best-reported cryptanalysis against PRESENT. Then, in Section 5, we show that there exists many different trails with the same input and output masks, the combination of which affects the distribution of the output in a hardly predictable way. We discuss the impact of such a "statistical hull" effect on the assumptions of Section 2.

4.1 (ext. 4) Multiple Trails Cryptanalysis

In the Statistical Saturation Attack of [6], the main limitation of the attack was the number of texts required to find the correct subkey. Above 24 rounds (and assuming that **ext. 2** yields the expected improvements), the data complexity of the attack reaches the codebook size of PRESENT. In this section, we consequently investigate the possibility to use several distinct trails in order to partially remove this limitation. Thanks to our branch-and-bound, we were able to generate many trails with different input masks and the same output mask. It allowed us to run several independent attacks in parallel, each one using a different trail and thus different partitioning of the plaintexts. As the output mask is the same, we can combine the results of the attacks together because the partial decryption involves the same subkey bits. In practice, each single-trail attack produces a vector containing the distances between the uniform and output distributions after partial decryption with the keyguess. A straightforward combination that was used in the context of linear cryptanalysis using multiple approximations (and for **ext. 2**) simply consists of taking the mean of these vectors. Such heuristic may not be optimal (*e.g.* compared to a maximum likelihood approach), but as detailed in [5], it is convenient when we lack the exact information about the expected distribution of the trail output after partial decryption. In particular, it is useful when linear hull (or related) effects imply errors when determining the approximated probability density functions of multidimensional approximations in linear cryptanalysis (see [9]). As will be exhibited in Section 5, this is exactly the type of situation that we face in this paper.

If we use n such trails, the time complexity is multiplied by the same factor because we have to repeat the partial decryption for each trail. The overhead required to combine the results is negligible. The effect on the data complexity is more intricate because each input mask defines a different partition of the plaintexts, according to the bits that are fixed and those that can change. For example, if the whole codebook is used, each plaintext will be used exactly once for each trail. The plaintexts can either be stored, requiring 2^{67} bytes of memory, or they can be generated on-the-fly, which would require $n * 2^{64}$ encryptions.

In order to evaluate the feasibility of this technique, we applied the statistical saturation attack on 9-round PRESENT for 128 different input masks with 2^{28} texts each. We selected the trails according to two different rules:

1. Best trails: we generated masks using our branch-and-bound and we selected the 8-round trails leading to the lowest theoretical data complexity.
2. Random trails: we generated trails from random (compatible) masks.

The results of our experiments are in Figure 5, which compares the mean gain (as defined in [2]) of attacks against 9-round PRESENT, exploiting different amounts of trails (up to 128), in function of the number of plaintexts used in the attacks. They illustrate that the combination of the information coming from different trails can be done constructively (*i.e.* lead to increased gains). For example, reaching a gain of 10 bits with a single trail requires approximately 2^{24} texts in the left part of the figure. But a combination of 2^7 trails leads to a nearly equivalent gain after 2^{18} texts in this case. Interestingly, the positive impact of combining several trails appears to be reduced for the random trails case (in the left part of the figure). In addition, there are two phenomenons that are worth being mentioned. First, the practical gains of trails having similar theoretical data complexity turned out to be quite different. For example, in the context of the best selected trails, we observed that 12.5% of them led to relatively low bias (less than 4 bits) even after 2^{28} texts. Second, the difference between the best and random trails was not as strong in practice as expected from the theoretical data complexities computed in the previous section. In both cases, this observation relates to the "statistical hull" effect discussed in Section 5.

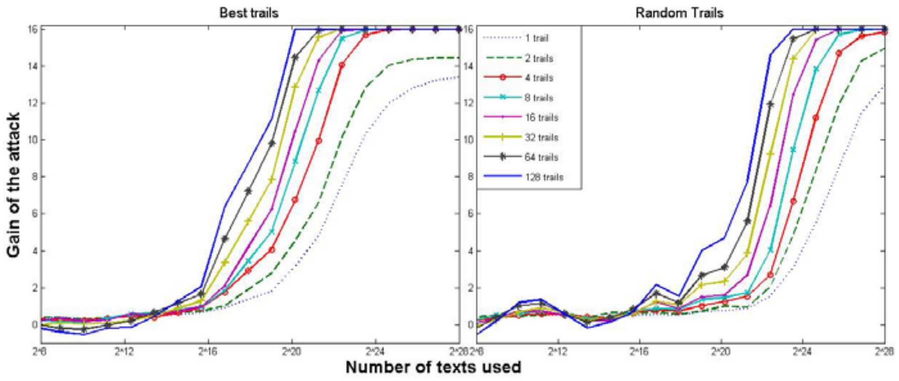


Fig. 5. Gains of attacks using multiple trails (left: best trails, right: random trails)

Note that, since the experiments in Figure 5 are far from using the full codebook of PRESENT, the attacks using different trails also use different plaintexts. By contrast, when estimating the effectiveness of attacks against more than 26-round PRESENT, the data complexity gets close to the full codebook. It means that exploiting multiple trails will require to rearrange (and hence, reuse) the codebook several times. Such a context raises the question to evaluate whether these multiple partitions of the codebook also improve the gain of the attack. Quite naturally, generating the full codebook is unfeasible for a 64-bit cipher. As a first step, we consequently considered a reduced-size version of PRESENT, with 16-bit blocks [12]. This allowed us to compute the average gain of one versus a combination of 128 trails, for different amounts of plaintexts. The results of these experiments are in Figure 6, for attacks against different number of rounds.

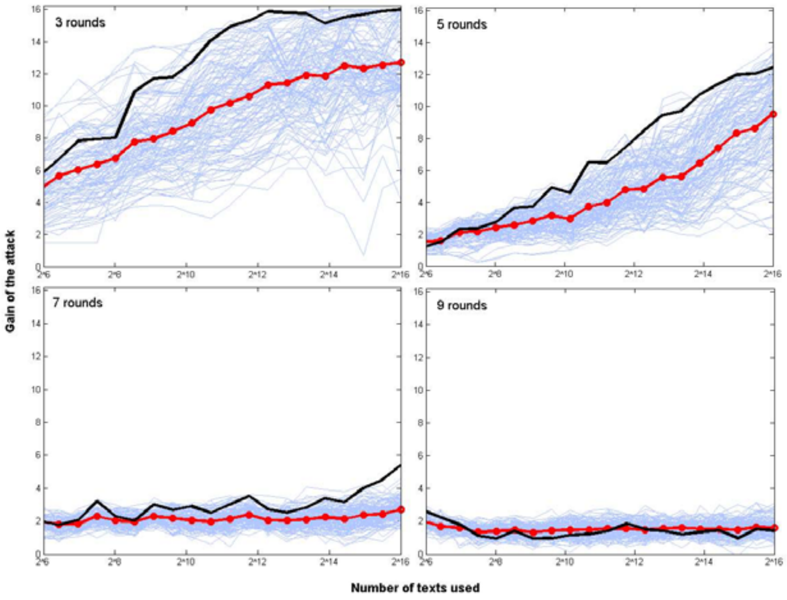


Fig. 6. Comparison between the gain of a single trail and the combined gain of 128 trails, using the full codebook against a simplified PRESENT with 16-bit block size

A first observation is that, even in this extreme context, the combination of the trails improves the overall gain of the attack significantly. That is, the bold plain curves (representing the combined gains) exceed the bold dotted curves (representing the average gains of single trails - the other curves representing all the 128 single-trail experiments). On the other hand, the improvements are not as large as in Figure 5, arguably because in such a small scale example, the input masks of the different trails are correlated. Also, it is noticeable that for the 9-round case, the gain of the multi-trail attack is similar to the one of a single-trail attack. This illustrates a context where the key-dependent signal provided by a single trail is so small that combining 128 multiple trails is not sufficient to reach a significant gain (since multiple trails can only be used to amplify an existing signal, here too small for the considered data complexities).

Summarizing, in the best case, different trails bring independent information, meaning that using two trails is equivalent to doubling the amount of texts with a single trail (this is the expectation in multiple linear cryptanalysis [2,9])[1]. In practice, these (best) conditions of independence (*e.g.* for the masks) are not perfectly respected in our context. But as the previous experiments illustrate for reduced-round PRESENT, different trails yield useful information, even when recombining the same set of plaintext with correlated masks. We leave the exact evaluation of these dependencies as an important scope for further research.

[1] Just as it is expected when using multiple fixed values in **ext. 2**.

4.2 Consequence for the Security of PRESENT-128

The previous section showed experimentally that combining multiple trails can lead to an improvement of the attack's gain with constant data complexity. In this section, we consider the impact of this observation for the security of PRESENT and quantify the overheads that it causes in terms of time and memory complexity. In particular, we analyze the possibility to perform a key-recovery attack exploiting the complete codebook of PRESENT-128.

According to [6], an attack against 24 rounds requires between 2^{57} and 2^{60} texts and the data complexity increases by a factor of 2^3 for every additional round. This was experimentally confirmed in Figure 4 for up to 16 rounds. If we extrapolate these estimations for 7 more rounds, it amounts to a total of approximately $2^{60+3\cdot 7} = 2^{81}$ texts for 31 rounds, which is more than the whole codebook. However, using multiple trails, we can decrease this complexity by extracting more information from a reduced number of texts. For example, using $2^{81}/2^{64} = 2^{17}$ trails with similar distributions as the one in [6] with the whole codebook - and assuming that they give rise to independent information ! - should be enough to recover 48 bits of the key with a significant gain.

The time complexity of such an hypothetical attack would be 2^{64} memory accesses for each trail, meaning 2^{81} memory accesses for all the 2^{17} trails. It would additionally require 2^{67} bytes to store the codebook. This complexity is slightly higher than an exhaustive search for 80-bit keys, but is a significant improvement for a 128-bit key. Also, there is a possible time-memory tradeoff since one can avoid storing the codebook by re-generating it for each trail.

Again, it is important to emphasize that these complexities are optimistic compared to what would be observed if experiments could be launched with the full codebook. This is because they assume that multiple trails bring independent information. As experimented in the previous section, this is only correct up to a certain (for now, hard to quantify) extent. Hence, it is necessary to multiply our estimated time complexities by a constant factor (as it was done with the data complexities in [6]). These corrective terms should mainly incorporate two effects: first, the possible correlation between different mask and trails as mentioned in this section; second, the possible deterioration and key dependencies of the statistical biases of single trails when the number of rounds increases, due to the statistical hull effect that we detail in the next section. Since we do not have a sound theory to analyze this statistical hull effect, and its experimental evaluation beyond 16 rounds is computationally intensive, we can only conjecture that the combination of multiple trails can be used to trade data complexity for time complexity up to a certain level. Yet, it remains that multiple trails improve the previous results from CT-RSA 2009. And the approximated time complexity of the hypothetical attack (*i.e.* 2^{81}) is small enough compared to 2^{128}, so that there is a reasonable chance that it will remain faster than exhaustive key search, even after the introduction of these corrections. At least, these estimations raise interesting questions about the number of rounds in PRESENT-128.

5 Statistical Hull Effect

As detailed in Section 4.1, the theoretical data complexities computed following the transition matrices in Section 2 do not always correspond to our practical experiments. In this section, we underline one possible reason explaining this divergence, in relation with the assumption of uniform distributions for the bits that are not part of the trail. That is, while this assumption is nicely respected for the input plaintexts, it becomes incorrect as the number of rounds increases. In fact, this behavior can be related to the statistical hull effect that has been put forward in the context of linear cryptanalysis. A linear hull describes a set of linear approximations that share the same input and output masks, but have different trails and different biases. Consequently, each of these approximations contributes to the bias of the overall approximation [18]. This phenomenon explains why linear cryptanalysis can perform better than expected by the theoretical bias evaluated for a particular approximation. Differential cryptanalysis has a similar concept of differential that is made of several differential characteristics with the same input and output differences, *e.g.* described in [14].

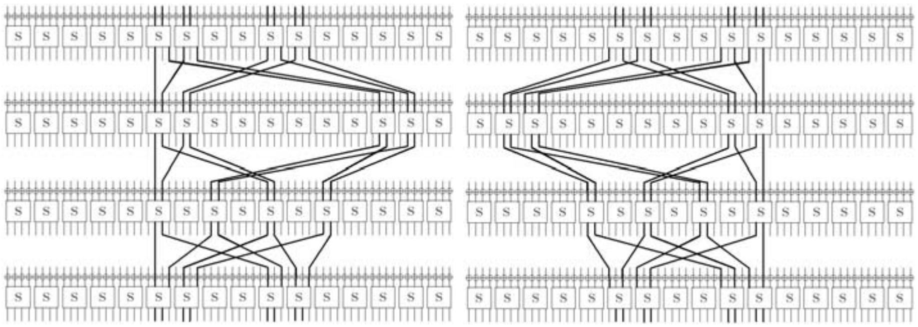


Fig. 7. 4-round trails with the same input and output masks

In Statistical Saturation Attacks, an analogous phenomenon can also be observed. Namely, several trails with the same active input and output bits can be found, each of them having its own specific transition matrix. For example, two trails with the same input and output masks as the one of Figure 1 are given in Figure 7. By running our branch-and-bound algorithm, we could find numerous other trails corresponding to this input and output masks, as detailed in Table 1.

Table 1. Number of trails with the same input and output masks as in Figure 7

#rounds	*#trails*
2	1
3	5
4	54
5	1044

The table directly suggests that the number of such trails increases exponentially with the number of rounds. For a large enough number of trails, it consequently becomes difficult to estimate their global effect on the distribution of the output bits. That is, as the trails may be correlated, the combined output distribution is not a simple combination of the theoretical output distributions of each trail. This observation can in fact be related to the work of Keliher *et al.* [13], in which the estimation of an upper bound for the linear hull effect was shown to be computationally hard in the number of rounds.

In the context of linear cryptanalysis, such experiments explain why, as the number of rounds increases, random masks can be almost as effective in recovering a key than a carefully selected trail (as witnessed, *e.g.* by Vaudenay's χ^2 cryptanalysis [21]). Strong hull effects may also imply key dependencies in the sense that the behavior of a trail for different keys may not be identical anymore (hence illustrating that the key equivalence hypothesis first discussed in [8] would not hold for PRESENT). In our (mainly experimental) setting, we conjecture that similar effects explain the deviations between the practical gain of trails having similar deviations from uniform under the assumptions of Section 2. However, we note that for a number of trails (*e.g.* the iterative one in Figure 1), these assumptions holds nicely. Analyzing the possible differences between these experimental observations and the ones made in the context of a linear cryptanalysis in another interesting scope for further research.

6 Conclusion and Further Works

A summary of the published cryptanalysis results against PRESENT is given in Table 2. It shows that Statistical Saturation Attacks outperform other types of cryptanalyses (in particular linear and differential) against this cipher. This is due to the design of its permutation layer. The main outcome of this paper is to show that the use of multiple trails allows improving the previous result of CT-RSA 2009. Also, if the assumptions in this paper are verified for larger number of rounds, the use of multiple-trails could lead to attacks with smaller time complexity than exhaustive key search against the full PRESENT-128.

As discussed in the previous sections, these estimations have to be considered with care, which is made explicit with the constant multiplicative factor c that we give for the time complexities in the table. This situation is similar to the one in linear cryptanalysis, where the precise estimation of the complexities is made difficult by the large cardinality of the trails to investigate.

In fact, the situation in the present paper is even more difficult, since we have to deal with complete distributions rather than scalar bias values. Positively, the experimental attacks that we performed against reduced number of rounds confirm our theoretical estimations to a reasonable extent. They at least show a significant improvement of the attacks when using multiple trails.

While these results do not threaten the practical applications of PRESENT (especially since it is mainly its 80-bit version that was advertised in [3]), they

Table 2. Summary of attacks (italic are not experimented and use **ext. 2, 4**.)

#rounds	Attack	Data compl.	Time compl.	Memory compl.	Ref.
16	SSA	$c * 2^{36}$CP	2^{28} MA	2^{16} counters	[6]
16	DC	2^{64}CP	2^{65} MA	$6 * 2^{32}$ bits	[23]
17	RKR	2^{63}CP	2^{104} MA	2^{53} counters	[20]
24	*SSA*	$c * 2^{60}CP$	$2^{28}MA$	2^{16} *counters*	[6]
26	LH	2^{64}KP	$2^{98.7}$ MA	2^{40} counters	[17]
26	MLC	2^{64}KP	2^{72} MA	2^{34} bytes	[10]
27	*MT-SSA*	2^{64} *CP*	$c \cdot 2^{69}MA$	2^{67} *bytes*	This paper
29	*MT-SSA*	2^{64} *CP*	$c \cdot 2^{75}MA$	2^{67} *bytes*	This paper
31	*MT-SSA*	2^{64} *CP*	$c \cdot 2^{81}MA$	2^{67} *bytes*	This paper

CP-Chosen Plaintext, KP-Known Plaintext, MA-Memory Access
DC-Differential Cryptanalysis, SSA-Statistical Saturation Attack, RKR-Related Key Rectangle, MLC-Multidimensional Linear Cryptanalysis, LH-Linear Hull

raise interesting open questions. For example, they make a case for designing efficient ciphers in which all the statistical effects that can be exploited in cryptanalysis are taken into account. The decorrelation theory appears as an interesting alternative in this respect [22]. But most importantly, the present experimental work implies the need of a better understanding of the statistical saturation attack and its extensions (in particular, **ext. 2**, *i.e.* using multiple fixed values in the trails, and **ext. 4**, *i.e.* using multiple trails). This implies providing sound explanations for the statistical hull and correlation effects between masks and trails, informally described in this paper. The similarities of our results with recent works in multidimensional cryptanalysis [9] also need to be investigated.

References

1. Baignères, T., Junod, P., Vaudenay, S.: How Far Can We Go Beyond Linear Cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 432–450. Springer, Heidelberg (2004)
2. Biryukov, A., De Cannière, C., Quisquater, M.: On Multiple Linear Approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004)
3. Bogdanov, A., Knudsen, L., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
4. Collard, B., Standaert, F.-X., Quisquater, J.-J.: Improving the Time Complexity of Matsui's Linear Cryptanalysis. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 77–88. Springer, Heidelberg (2007)
5. Collard, B., Standaert, F.-X., Quisquater, J.-J.: Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 382–397. Springer, Heidelberg (2008)

6. Collard, B., Standaert, F.-X.: A Statistical Saturation Attack on the Block Cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
7. Collard, B., Standaert, F.-X.: A Statistical Saturation Attack on the Block Cipher PRESENT, Errata and Improvement, http://www.dice.ucl.ac.be/~fstandae/PUBLIS/62b.pdf
8. Harpes, C., Kramer, G., Massey, J.: A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-Up Lemma. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 24–38. Springer, Heidelberg (1995)
9. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional Extension of Matsui's Algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 209–227. Springer, Heidelberg (2009)
10. Cho, J.Y.: Linear Cryptanalysis of Reduced-Round PRESENT. Cryptology ePrint Archive: Report 2009/397, http://eprint.iacr.org/2009/397
11. Kaliski, B.S., Robshaw, M.J.B.: Linear Cryptanalysis Using Multiple Approximations. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 26–39. Springer, Heidelberg (1994)
12. Leander, G.: Small Scale Variants of the Block Cipher PRESENT. IACR ePrint Archive, http://eprint.iacr.org/2010/143
13. Keliher, L., Meijer, H., Tavares, S.E.: New Method for Upper Bounding the Maximum Average Linear Hull Probability for SPNs. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 420–436. Springer, Heidelberg (2001)
14. Lai, X., Massey, J.L., Murphy, S.: Markov Ciphers and Differential Cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991)
15. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
16. Minier, M., Gilbert, H.: Stochastic Cryptanalysis of Crypton. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 121–133. Springer, Heidelberg (2001)
17. Nakahara Jr., J., Seperhdad, P., Zhang, B., Wang, M.: Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT. To appear in the proceedings of CANS 2009, Kanazawa, Japan (December 2009)
18. Nyberg, K.: Linear Approximation of Block Ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
19. Ohkuma, K.: Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 249–265. Springer, Heidelberg (2009)
20. Özen, O., Varici, K., Tezcan, C.: Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 90–107. Springer, Heidelberg (2009)
21. Vaudenay, S.: An Experiment on DES Statistical Cryptanalysis. In: The proceedings of the third ACM Conference on Computer and Communications Security (CCS 1996), New Delhi, India, pp. 139–147. ACM, New York (1996)
22. Vaudenay, S.: Decorrelation: A Theory for Block Cipher Security. Journal of Cryptology 16(4), 249–286 (2003)
23. Wang, M.: Differential Cryptanalysis of Reduced-Round PRESENT. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 40–49. Springer, Heidelberg (2008)

Multiset Collision Attacks on Reduced-Round SNOW 3G and SNOW 3G$^{\oplus}$

Alex Biryukov, Deike Priemuth-Schmid, and Bin Zhang

University of Luxembourg
{alex.biryukov,deike.priemuth-schmid,bin.zhang}@uni.lu

Abstract. The stream cipher SNOW 3G designed in 2006 by ETSI/SAGE is a base algorithm for the second set of 3GPP confidentiality and integrity algorithms. In this paper we study the resynchronization mechanism of SNOW 3G and of a similar cipher SNOW 3G$^{\oplus}$ using multiset collision attacks. For SNOW 3G we show a simple 13-round multiset distinguisher with complexity of 2^8 steps. We show full key recovery chosen IV resynchronization attacks for up to 18 out of 33 initialization rounds of SNOW3G$^{\oplus}$ with a complexity of 2^{57} to generate the data and 2^{53} steps of analysis.

Keywords: Stream ciphers, SNOW 3G, Resynchronization attack.

1 Introduction

The SNOW 3G stream cipher is the core of the 3GPP confidentiality and integrity algorithms UEA2 and UIA2, published in 2006 by the 3GPP Task Force [5]. Compared to its predecessor, SNOW 2.0 [4], SNOW 3G adopts a finite state machine (FSM) of three 32-bit words and 2 S-Boxes to increase the resistance against algebraic attacks by Billet and Gilbert [2]. Full evaluation of the design by the consortium is not public, but a survey of this evaluation is given in [6]. SNOW 3G$^{\oplus}$ (in which the two modular additions are replaced by xors) is also defined and evaluated in this document. The designers and external reviewers show that SNOW 3G has remarkable resistance against linear distinguishing attacks [7,8], while SNOW 3G$^{\oplus}$ offers much better resistance against algebraic attacks.

In this paper we analyze the resynchronization mechanizm of SNOW 3G and SNOW 3G$^{\oplus}$ using multiset collision attacks. This technique has proved itself useful against AES [3] but to the best of our knowledge has not been used yet for the analysis of the key-IV setup of stream ciphers. It seems natural to apply this technique to SNOW 3G since its finite state machine (FSM) is essentially a 96-bit AES like cipher in which the LFSR plays a role of a key-schedule. This picture is complicated by the fact that there is a feedback from the FSM to the LFSR during the setup phase (a feature never present in block ciphers) and that the attacker sees only 32-bits of output at a time, while the internal state keeps changing constantly.

We start by showing a very efficient multiset distinguisher for 13-round SNOW 3G with complexity of 2^8 steps. We then switch to the analysis of SNOW 3G$^{\oplus}$ which is a very good model for the main features of SNOW 3G, since its analysis is not blurred by the presence of carries. We have found an attack on up to 18-rounds (out of 33) with a complexity of 2^{57} to generate the data and 2^{53} steps of analysis. In this attack for the first 10 rounds the multiset propagates for free since we put it in the most significant byte of the IV word IV_0. It enters the FSM at the 11th round. Strong cancellations due to balanced properties of multisets stop to be useful after 15 rounds and we have to resort to *multiset collision* techniques which can allow us to go three more rounds deeper. Multisets still help us to cancel out the keystream words out of the keystream equation, which are an obstacle for a simple differential analysis at this depth. This attack is very technical and is more involved than attacks of similar type on block ciphers. We have experimentally verified the crucial parts of our attacks.

This paper is organized as follows. We give a description of SNOW 3G and SNOW 3G$^{\oplus}$ in Section 2. The multiset collision chosen IV attacks on round-reduced SNOW3G and SNOW 3G$^{\oplus}$ are presented in Section 3. Finally, some conclusions are given in Section 4.

2 Description of SNOW 3G and SNOW 3G$^{\oplus}$

The SNOW 3G stream cipher uses a 128-bit key and a 128-bit IV, considered as four 32-bit words vectors. It consists of a linear feedback shift register (LFSR) of 16 32-bit words and a finite state machine (FSM) with three 32-bit words, shown in Figure 1. Here '$\oplus$' denotes the bit-wise xor and '$\boxplus$' denotes the addition modulo 2^{32}. The feedback word of the LFSR is recursively computed as

$$s_{15}^{t+1} = \alpha^{-1} \cdot s_{11}^{t} \oplus s_2^t \oplus \alpha \cdot s_0^t,$$

where α is the root of the $GF(2^8)[x]$ polynomial $x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239}$ with β being the root of the $GF(2)[x]$ polynomial $x^8 + x^7 + x^5 + x^3 + 1$. The FSM has two input word s_5^t and s_{15}^t from the LFSR and is updated as follows.

$$R_3^t = S_2(R_2^{t-1}), \quad R_2^t = S_1(R_1^{t-1}), \quad R_1^t = R_2^{t-1} \boxplus (R_3^{t-1} \oplus s_5^{t-1}),$$

and output $F^t = (s_{15}^t \boxplus R_1^t) \oplus R_2^t$, where S_1 and S_2 are 32-bit to 32-bit S-boxes defined as compositions of 4 parallel applications of two 8-bit to 8-bit small S-boxes, S_R and S_Q, with a linear diffusion layer respectively. Here S_R is the well known AES S-box and S_Q is defined as $S_Q(x) = x \oplus x^9 \oplus x^{13} \oplus x^{15} \oplus x^{33} \oplus x^{41} \oplus x^{45} \oplus x^{47} \oplus x^{49} \oplus 0x25$ for $x \in GF(2^8)$ defined by $x^8 + x^6 + x^5 + x^3 + 1$. If we decompose a 32-bit word B into four bytes $B = B^0 \| B^1 \| B^2 \| B^3$ with B^0 being the most and B^3 the least significant bytes, then

$$S_i(B) = MC_i \cdot \begin{pmatrix} S_R(B^0) \\ S_R(B^1) \\ S_R(B^2) \\ S_R(B^3) \end{pmatrix} = \begin{pmatrix} 2\,1\,1\,3 \\ 3\,2\,1\,1 \\ 1\,3\,2\,1 \\ 1\,1\,3\,2 \end{pmatrix}_i \cdot \begin{pmatrix} S_R(B^0) \\ S_R(B^1) \\ S_R(B^2) \\ S_R(B^3) \end{pmatrix}, \ (i = 1, 2)$$

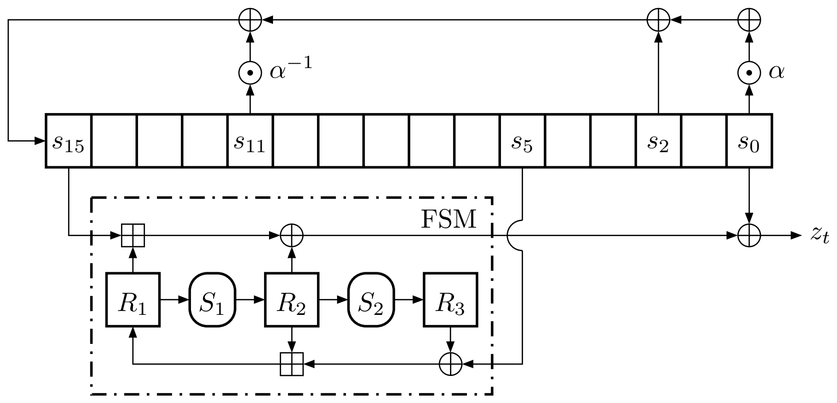


Fig. 1. Keystream generation of SNOW 3G

where MC_1 is the AES mix-column for S_1 over $GF(2^8)$ defined by $x^8 + x^4 + x^3 + x + 1$ and MC_2 is the similar operation for S_2 over $GF(2^8)$ defined by $x^8 + x^6 + x^5 + x^3 + 1$.

SNOW 3G is initialized with the key $K = (k_0, k_1, k_2, k_3)$ and the $IV = (IV_0, IV_1, IV_2, IV_3)$ as follows. Let $\mathbf{1}$ be the all-one word, first load the LFSR as follows.

$$\begin{array}{llll} s_{15} = k_3 \oplus IV_0 & s_{14} = k_2 & s_{13} = k_1 & s_{12} = k_0 \oplus IV_1 \\ s_{11} = k_3 \oplus \mathbf{1} & s_{10} = k_2 \oplus \mathbf{1} \oplus IV_2 & s_9 = k_1 \oplus \mathbf{1} \oplus IV_3 & s_8 = k_0 \oplus \mathbf{1} \\ s_7 = k_3 & s_6 = k_2 & s_5 = k_1 & s_4 = k_0 \\ s_3 = k_3 \oplus \mathbf{1} & s_2 = k_2 \oplus \mathbf{1} & s_1 = k_1 \oplus \mathbf{1} & s_0 = k_0 \oplus \mathbf{1} \end{array} .$$

The FSM is initialized with $R_1 = R_2 = R_3 = 0$. Then run the cipher 32 times with the FSM output F xored to the feedback of the LFSR and no keystream generated. After this, the cipher is switched into the keystream generation mode, but the first keystream word is discarded. Hence, there are 33 initialization rounds. The keystream word generated at clock t is

$$\text{SNOW 3G:} \qquad z^t = s_0^t \oplus F^t = (s_{15}^t \boxplus R_1^t) \oplus R_2^t \oplus s_0^t \qquad (1)$$

$$\text{SNOW 3G}^{\oplus}\text{:} \qquad z^t = s_0^t \oplus F^t = s_{15}^t \oplus R_1^t \oplus R_2^t \oplus s_0^t \qquad (2)$$

If we replace the two modulo additions in SNOW 3G by xors, we get SNOW 3G$^{\oplus}$.

3 Chosen IV Attacks on Reduced Round SNOW 3G and SNOW 3G$^{\oplus}$

In this section, we evaluate the security margin of SNOW 3G and SNOW 3G$^{\oplus}$ against chosen IV attacks. Our results are listed in Table 1.

Table 1. Our results on SNOW 3G and SNOW 3G$^{\oplus}$

Cipher	Round	Data	Time	Type
SNOW 3G	13	2^8	2^8	distinguisher
SNOW 3G$^{\oplus}$	14	2^8	2^8	distinguisher
SNOW 3G$^{\oplus}$	14	$2^{12.1}$	2^{27}	full key recovery
SNOW 3G$^{\oplus}$	15	$2^{32.1}$	$2^{32.4}$	partial state recovery
SNOW 3G$^{\oplus}$	18	2^{57}	2^{53}	full key recovery

3.1 Distinguishing Attack on 13-Round SNOW 3G

We first look at SNOW 3G with 13-round initializations. For each secret key K, we randomly choose an IV and make a multiset at the most significant byte IV_0^0 of the most significant word IV_0 such that it takes all the byte values in $[0, 255]$ exactly once. From the key/IV loading of SNOW 3G, we known that the multiset difference is introduced in the most significant byte of s_{15}. Now we trace the multiset difference propagation in the 19 registers during the 13 rounds of initialization, which is shown in Table 2. The differences at round i are the differences at the end of the corresponding round.

Here we actually have 256 IVs associated with the same key. Denote the first keystream word generated by (K, IV) when $IV_0^0 = i$ by $z_{i,0}$ and denote the corresponding content in the j-th LFSR cell by $s_{i,j}$, then we have

$$\bigoplus_{i=0}^{255} z_{i,0} = \bigoplus_{i=0}^{255}(s_{i,0} \oplus R_{i,2}) \oplus \bigoplus_{i=0}^{255}(s_{i,15} \boxplus R_{i,1}) = \bigoplus_{i=0}^{255}(s_{i,15} \boxplus R_{i,1}).$$

From Table 2, the least significant bit is always 0. To show that this property holds for the other 7 bits in the least significant byte, it suffices to note that the least significant bytes of R_1 forms an permutation set, while the least significant bytes of $s_{i,15}$ are the same, so by lemma 2 in [1], the least significant byte of $\bigoplus_{i=0}^{255} z_{i,0}$ is always 0. In experiments, we randomly choose 2^6 IVs to check this property. For each chosen IV, we make a multiset attack as above and calculate $\bigoplus_{i=0}^{255} z_{i,0}$. We found that the least significant byte of this sum is always 0. This gives a very simple distinguishing attack of complexity 2^8 IV's and key-stream words for 13-round SNOW 3G. We expect that this attack can be extended into a key recovery attack on 14-round SNOW 3G, but we preferred to concentrate on breaking more rounds of SNOW 3G$^{\oplus}$ instead.

3.2 Distinguishing Attack on 14-Round SNOW 3G$^{\oplus}$

The above distinguisher can be extended by several rounds in SNOW 3G$^{\oplus}$. For each secret key K, we also randomly choose an IV and make a multiset at the most significant byte IV_0^0 of the most significant word IV_0 such that it takes all the byte values in $[0, 255]$ exactly once. The multiset difference propagation is formally derived in Table 6 in Appendix A, where $\Delta_i = i$ denotes the difference in IV_0^0 for $i = 0, \cdots, 255$. From that table, we can see that until round 11, the

Table 2. Multiset sum propagation in 13-round initialization of SNOW 3G. (? indicates that the sum in this byte takes some random value and 0 means that the corresponding sum is 0).

	s_{15}	s_{14}	s_{13}	s_{12}	s_{11}	s_{10}	s_9	s_8	$s_{j:\ (0\leq j\leq 7)}$	R_1	$R_{i:\ (i=2,3)}$
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
5	0	0	0	0	0	0	0	0	0	0	0
6	?000	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
7	?000	?000	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
8	?000	?000	?000	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
9	?000	?000	?000	?000	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
10	?000	?000	?000	?000	?000	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
11	??00	?000	?000	?000	?000	?000	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$
12	??00	??00	?000	?000	?000	?000	?000	0	$\vdots$	0	$\vdots$
13	0?00	??00	??00	?000	?000	?000	?000	?000	0	???0	0

difference $\varDelta_i$ will not affect the memory registers R_i $(i = 1, 2, 3)$. Hence, at the end of round 10, the contents in R_i $(i = 1, 2, 3)$ are three unknown constants not depending on IV_0 and the difference $\varDelta_i$. Let the unknown constant in R_i be c_i for $i = 1, 2, 3$. Table 3 shows the contents evolution process in the three memory registers. We have the following theorem:

Theorem 1. *If there are* 14 *initialization rounds in SNOW 3G*$^{\oplus}$ *and the multiset is taken at* IV_0^0*, then* $\bigoplus_{i=0}^{255} z_{i,0} = (2a, 3a, a, a)$ *with* $a \in [0, 255]$*.*

Proof. From the line 14 of Table 3 and the keystream equation (2), we have

$$\begin{aligned}\bigoplus_{i=0}^{255} z_{i,0} &= \bigoplus_{i=0}^{255}(s_{i,0} \oplus R_{i,2} \oplus s_{i,15} \oplus R_{i,1}) = \bigoplus_{i=0}^{255}(R_{i,1} \oplus R_{i,2})\\ &= \bigoplus_{i=0}^{255}(B_{i,13} \oplus C_{i,13} \oplus s_{18} \oplus \varDelta_i) \oplus \bigoplus_{i=0}^{255} MC_1[S_1(A_{13} \oplus \varDelta_i)].\end{aligned}$$

Note that there is only one active byte in IV_0, so we have $\bigoplus_{i=0}^{255} B_{i,13} = 0$, $\bigoplus_{i=0}^{255} C_{i,13} = 0$ by Lemma 2 in [1]. Since $\forall i$, $s_{i,18} = s_{18}$, we have $\bigoplus_{i=0}^{255} s_{18} = 0$. Thus, $\bigoplus_{i=0}^{255} z_{i,0} = \bigoplus_{i=0}^{255} MC_1[S_1(A_{13} \oplus \varDelta_i)] = MC_1 \cdot \bigoplus_{i=0}^{255}[S_1(A_{13} \oplus \varDelta_i)]$. Expanding $\bigoplus_{i=0}^{255}[S_1(A_{i,13} \oplus \varDelta_i)]$, we have

Table 3. The contents evolution process of the memory registers during the $10-17$ initializations round of SNOW 3G$^{\oplus}$. (A_i, B_i and C_i are the contents in R_1, R_2 and R_3 for $i \geq 11$ with A_i being the content when $\Delta_i = 0$. * denotes permutation property of the values, c denotes constant property, ? denotes property to be determined, b denotes balanced property).

	R_1	R_2	R_3
10	c_1	c_2	c_3
11	$c_2 \oplus c_3 \oplus k_3 \oplus IV_0 \oplus \Delta_i$	$MC_1[S_1(c_1)]$	$MC_2[S_2(c_2)]$
	$(*, c, c, c)$	(c, c, c, c)	(c, c, c, c)
12	$B_{11} \oplus C_{11} \oplus s_{16} \oplus \Delta_i$	$MC_1[S_1(A_{11} \oplus \Delta_i)]$	$MC_2[S_2(B_{11})]$
	$(*, c, c, c)$	$(*, *, *, *)$	(c, c, c, c)
13	$B_{12} \oplus C_{12} \oplus s_{17} \oplus \Delta_i$	$MC_1[S_1(A_{12} \oplus \Delta_i)]$	$MC_2[S_2(B_{12})]$
	$(b, *, *, *)$	$(*, *, *, *)$	(b, b, b, b)
14	$B_{13} \oplus C_{13} \oplus s_{18} \oplus \Delta_i$	$MC_1[S_1(A_{13} \oplus \Delta_i)]$	$MC_2[S_2(B_{13})]$
	(b, b, b, b)	$(?, ?, ?, ?)$	(b, b, b, b)
15	$B_{14} \oplus C_{14} \oplus s_{19} \oplus \Delta_i$	$MC_1[S_1(A_{14} \oplus \Delta_i)]$	$MC_2[S_2(B_{14})]$
16	$B_{15} \oplus C_{15} \oplus s_{20} \oplus \Delta_i \oplus \alpha^{-1}\Delta_i$	$MC_1[S_1(A_{15} \oplus \Delta_i)]$	$MC_2[S_2(B_{15})]$
17	$B_{16} \oplus C_{16} \oplus s_{21} \oplus \Delta_i$	$MC_1[S_1(A_{16} \oplus \Delta_i \oplus \alpha^{-1}\Delta_i)]$	$MC_2[S_2(B_{16})]$

$$\bigoplus_{i=0}^{255} S_1(A_{13} \oplus \Delta_i) = \bigoplus_{i=0}^{255} S_1\big(s_{17} \oplus C_{12} \oplus \Delta_i \oplus MC_1[S_1(A_{11} \oplus \Delta_i)]\big), \qquad (3)$$

where $A_{11} = c_2 \oplus c_3 \oplus k_3 \oplus IV_0$. From (3) and Table 4, we can see that s_{17}, C_{12} and A_{11} do not dependent on Δ_i. Let $A_{11} = A_{11}^0 \| A_{11}^1 \| A_{11}^2 \| A_{11}^3$ and $s_{17} \oplus C_{12} = m^0 \| m^1 \| m^2 \| m^3$, then we have the byte equations:

$$\bigoplus_{i=0}^{255} S_R[2S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1) \oplus S_R(A_{11}^2) \oplus 3S_R(A_{11}^3) \oplus m^0 \oplus \Delta_i] = a \qquad (4)$$

$$\bigoplus_{i=0}^{255} S_R[3S_R(A_{11}^0 \oplus \Delta_i) \oplus 2S_R(A_{11}^1) \oplus S_R(A_{11}^2) \oplus S_R(A_{11}^3) \oplus m^1] = 0 \qquad (5)$$

$$\bigoplus_{i=0}^{255} S_R[S_R(A_{11}^0 \oplus \Delta_i) \oplus 3S_R(A_{11}^1) \oplus 2S_R(A_{11}^2) \oplus S_R(A_{11}^3) \oplus m^2] = 0 \qquad (6)$$

$$\bigoplus_{i=0}^{255} S_R[S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1) \oplus 3S_R(A_{11}^2) \oplus 2S_R(A_{11}^3) \oplus m^3] = 0 \qquad (7)$$

It is easy to see that (5), (6) and (7) equal to 0 for any value of its inputs, while the value of (4) is dependent on the input value. Let (3) $= a$, by passing the vector $(a, 0, 0, 0)$ through the MC_1 we finish the proof. $\square$

Theorem 1 shows that the four sub-bytes of $\bigoplus_{i=0}^{255} z_{i,0}$ are correlated. In experiments, we randomly choose 2^6 IVs to verify it. We found that Theorem 1 holds all the time. This property is a distinguisher with a complexity of 2^8 steps, given 1 keystream word for each IV.

3.3 Key Recovery Attack on 14-Round SNOW 3G$^{\oplus}$

The above distinguisher can be converted into a key recovery attack on 14-round SNOW 3G$^{\oplus}$. It works as follows. From Theorem 1, we have

$$\bigoplus_{i=0}^{255} S_R[2S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1) \oplus y(0) \oplus \Delta_i] = \bigoplus_{i=0}^{255} z_{i,0}^3, \tag{8}$$

where $y(0) = S_R(A_{11}^2) \oplus 3S_R(A_{11}^3) \oplus m^0$. To solve it, we randomly choose two other IVs, IV' and IV'', such that

1. $IV_r' = IV_r'' = IV_r$ for $r = 1, 2, 3$.
2. $IV_0^{'r} = IV_0^r$ for $r = 0, 2, 3$.
3. $IV_0^{'1} = IV_0^1 \oplus \beta_1$.
4. $IV_0^{''r} = IV_0^r$ for $r = 0, 2, 3$.
5. $IV_0^{''1} = IV_0^1 \oplus \beta_2$.

with $\beta_i \in GF(2^8)$ for $i = 1, 2$. For IV' and IV'', we also make a multiset at the corresponding most significant byte. Our observation is that for such chosen IVs, we can derive similar equations to (8) due to the linearity of $A_{11} = c_2 \oplus c_3 \oplus k_3 \oplus IV_0$, $A_{11}' = c_2 \oplus c_3 \oplus k_3 \oplus IV_0'$ and $A_{11}'' = c_2 \oplus c_3 \oplus k_3 \oplus IV_0''$:

$$\bigoplus_{i=0}^{255} S_R[2S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1 \oplus \beta_1) \oplus y(\beta_1) \oplus \Delta_i] = \bigoplus_{i=0}^{255} z_{i,0}^{'3} \tag{9}$$

$$\bigoplus_{i=0}^{255} S_R[2S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1 \oplus \beta_2) \oplus y(\beta_2) \oplus \Delta_i] = \bigoplus_{i=0}^{255} z_{i,0}^{''3}, \tag{10}$$

where $y(\beta_1) = y(\beta_2) = y(0)$ according to the conditions 2 and 4. From $(8) - (10)$, we can derive A_{11}^1 with 2^{24} steps. It is interesting to note that we cannot restore A_{11}^0 and $y(0)$ together with A_{11}^1 from $(8) - (10)$. The reason is that $(8) - (10)$ cannot be regraded as random equations, which is supported by extensive experiments. Note that the information we recovered is the byte where we introduce the difference β_i. In order to determine other bytes of A_{11}, we just shift the byte position where the difference β_i is introduced to $A_{11}^r (r = 2, 3)$. Thus, we will get equations looking like

$$\bigoplus_{i=0}^{255} S_R[2S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1) \oplus S_R(A_{11}^2 \oplus \gamma_j) \oplus 3S_R(A_{11}^3) \oplus m^0 \oplus \Delta_i] = \bigoplus_{i=0}^{255} z_{i,0}^3$$

$$\bigoplus_{i=0}^{255} S_R[2S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1) \oplus S_R(A_{11}^2) \oplus 3S_R(A_{11}^3 \oplus \delta_j) \oplus m^0 \oplus \Delta_i] = \bigoplus_{i=0}^{255} z_{i,0}^3$$

for randomly chosen $\gamma_j, \delta_j \in GF(2^8)$ $(j = 1, 2, 3)$, from which we can recover A_{11}^2 and A_{11}^3. We can determine A_{11}^0 by shifting the multiset position to another byte and introduce the byte differences at A_{11}^0. Thus, we will get

$$\bigoplus_{i=0}^{255} S_R[3S_R(A_{11}^0 \oplus \xi_j) \oplus 2S_R(A_{11}^1 \oplus \Delta_i) \oplus S_R(A_{11}^2) \oplus S_R(A_{11}^3) \oplus m^1 \oplus \Delta_i] = \bigoplus_{i=0}^{255} z_{i,0}^0$$

for randomly chosen $\xi_j \in GF(2^8)(j=1,2,3)$. In this case, $\bigoplus_{i=0}^{255} z_{i,0}=(a, 2a, 3a, a)$ with $a \in GF(2^8)$.

Next, we can restore $s_{17} \oplus C_{12}$ by substituting A_{11}^i $(i = 0, 1, 2, 3)$ into the solution set of $(8) - (10)$, identifying the corresponding variable and determining m^i. To make a full key recovery, we need to look at the second keystream word and derive the following byte equations:

$$\bigoplus_{i=0}^{255} S_R[s_{18}^0 \oplus \Delta_i \oplus \underbrace{2f_0 \oplus f_1 \oplus f_2 \oplus 3f_3}_{MC_2} \oplus 2S_R(A_{12}^0 \oplus \Delta_i) \oplus S_R(A_{12}^1 \oplus \alpha_j) \oplus S_R(A_{12}^2) \oplus 3S_R(A_{12}^3)] = \bigoplus_{i=0}^{255} z_{i,1}^0 \oplus \bigoplus_{i=0}^{255} z_{i,0}^3 \tag{11}$$

$$\bigoplus_{i=0}^{255} S_R[s_{18}^1 \oplus \alpha_j \oplus \underbrace{3f_0 \oplus 2f_1 \oplus f_2 \oplus f_3}_{MC_2} \oplus 3S_R(A_{12}^0 \oplus \Delta_i) \oplus 2S_R(A_{12}^1 \oplus \alpha_j) \oplus S_R(A_{12}^2) \oplus S_R(A_{12}^3)] = \bigoplus_{i=0}^{255} z_{i,1}^1 \tag{12}$$

$$\bigoplus_{i=0}^{255} S_R[s_{18}^2 \oplus \underbrace{f_0 \oplus 3f_1 \oplus 2f_2 \oplus f_3}_{MC_2} \oplus S_R(A_{12}^0 \oplus \Delta_i) \oplus 3S_R(A_{12}^1 \oplus \alpha_j) \oplus 2S_R(A_{12}^2) \oplus S_R(A_{12}^3)] = \bigoplus_{i=0}^{255} z_{i,1}^2 \tag{13}$$

$$\bigoplus_{i=0}^{255} S_R[s_{18}^3 \oplus \underbrace{f_0 \oplus f_1 \oplus 3f_2 \oplus 2f_3}_{MC_2} \oplus S_R(A_{12}^0 \oplus \Delta_i) \oplus S_R(A_{12}^1 \oplus \alpha_j) \oplus 3S_R(A_{12}^2) \oplus 2S_R(A_{12}^3)] = \bigoplus_{i=0}^{255} z_{i,1}^3, \tag{14}$$

where

$$f_0 = S_Q(2S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1 \oplus \alpha_j) \oplus S_R(A_{11}^2) \oplus 3S_R(A_{11}^3)) \tag{15}$$

$$f_1 = S_Q(3S_R(A_{11}^0 \oplus \Delta_i) \oplus 2S_R(A_{11}^1 \oplus \alpha_j) \oplus S_R(A_{11}^2) \oplus S_R(A_{11}^3)) \tag{16}$$

$$f_2 = S_Q(S_R(A_{11}^0 \oplus \Delta_i) \oplus 3S_R(A_{11}^1 \oplus \alpha_j) \oplus 2S_R(A_{11}^2) \oplus S_R(A_{11}^3)) \tag{17}$$

$$f_3 = S_Q(S_R(A_{11}^0 \oplus \Delta_i) \oplus S_R(A_{11}^1 \oplus \alpha_j) \oplus 3S_R(A_{11}^2) \oplus 2S_R(A_{11}^3)) \tag{18}$$

for randomly chosen α_j $(j = 1, 2, 3)$. Since A_{11} is known, we can recover A_{12}^0, A_{12}^1 and $S_R(A_{12}^2) \oplus 3S_R(A_{12}^3) \oplus s_{18}^0$ from (11) by the three equations corresponding to α_j $(j = 1, 2, 3)$. Shifting the byte position of α_j and the multiset position

Δ_i to the other positions, we can derive A_{12}^2, A_{12}^3 in a similar manner. After obtaining A_{12}, we can restore s_{18} by (11) – (14), which is a linear combination of the internal states after initialization. Then we proceed in the same way as above to look at the next 15 keystream words and derive 16 linear equations on the internal states of LFSR after the key/IV setup. Solving this linear system will yield the initial internal state of the LFSR. The values of the three memory registers can be recovered from A_{11}, A_{12} and $s_{17} \oplus C_{12}$ according to Table 4. Then we can run the cipher backwards to recover the secret key since all the steps here are invertible.

The total complexity of the above attack is $4 \cdot 2^{24} + 4 \cdot 2^8 + 4 \cdot 2^{24} + 4 \cdot 2^8 \approx 2^{27}$ steps and 17 keystream words for each IV. We also made experiments to verify the attack. The experiments show that there are exactly 256 solutions to (7) – (9) with a common A_{11}^1 and (10) – (13) behaves like random equations. From (10) – (13), we always recover A_{12} and s_{18} correctly.

3.4 Key Recovery Attack on 15-Round SNOW 3G$^\oplus$

In this and the following subsections, we extend previous ideas and combine them with the Gilbert-Minier [3] like ideas of functional collisions in order to cover more rounds of SNOW 3G$^\oplus$.

For 15-round SNOW 3G$^\oplus$, from the first keystream word we have:

$$\bigoplus_{i=0}^{255} MC_1[S_1(s_{17} \oplus C_{12} \oplus \Delta_i \oplus MC_1[S_1(A_{11} \oplus \Delta_i)])] \oplus \bigoplus_{i=0}^{255} MC_1[S_1(s_{18}$$
$$\oplus \Delta_i \oplus MC_2[S_2(MC_1[S_1(A_{11} \oplus \Delta_i)])] \oplus MC_1[S_1(A_{12} \oplus \Delta_i)])] = \bigoplus_{i=0}^{255} z_{i,0}.$$

Note that the first term $B_{14} = \bigoplus_{i=0}^{255} MC_1[S_1(s_{17} \oplus C_{12} \oplus \Delta_i \oplus MC_1[S_1(A_{11} \oplus \Delta_i)])]$ has a special pattern $(2a, 3a, a, a)$ with unknown a. Denote the inverse of MC_1 by MC_1^{-1}, we have

$$\bigoplus_{i=0}^{255} S_1(s_{18} \oplus \Delta_i \oplus MC_2[S_2(B_{12})] \oplus MC_1[S_1(A_{12} \oplus \Delta_i)]) = MC_1^{-1}(\bigoplus_{i=0}^{255} z_{i,0} \oplus B_{14}). \tag{19}$$

Expanding (19) to byte equations, we have

$$\bigoplus_{i=0}^{255} S_R[s_{18}^0 \oplus \Delta_i \oplus \underbrace{2f_0 \oplus f_1 \oplus f_2 \oplus 3f_3}_{MC_2} \oplus 2S_R(A_{12}^0 \oplus \Delta_i) \tag{20}$$
$$\oplus S_R(A_{12}^1 \oplus \eta_j) \oplus S_R(A_{12}^2) \oplus 3S_R(A_{12}^3)] = kc_j^0 \oplus a$$

$$\bigoplus_{i=0}^{255} S_R[s_{18}^1 \oplus \eta_j \oplus \underbrace{3f_0 \oplus 2f_1 \oplus f_2 \oplus f_3}_{MC_2} \oplus 3S_R(A_{12}^0 \oplus \Delta_i) \tag{21}$$
$$\oplus 2S_R(A_{12}^1 \oplus \eta_j) \oplus S_R(A_{12}^2) \oplus S_R(A_{12}^3)] = kc_j^1$$

$$\bigoplus_{i=0}^{255} S_R[s^2_{18} \oplus \underbrace{f_0 \oplus 3f_1 \oplus 2f_2 \oplus f_3}_{MC_2} \oplus S_R(A^0_{12} \oplus \Delta_i) \qquad (22)$$

$$\oplus\, 3S_R(A^1_{12} \oplus \eta_j) \oplus 2S_R(A^2_{12}) \oplus S_R(A^3_{12})] = kc^2_j$$

$$\bigoplus_{i=0}^{255} S_R[s^3_{18} \oplus \underbrace{f_0 \oplus f_1 \oplus 3f_2 \oplus 2f_3}_{MC_2} \oplus S_R(A^0_{12} \oplus \Delta_i) \qquad (23)$$

$$\oplus\, S_R(A^1_{12} \oplus \eta_j) \oplus 3S_R(A^2_{12}) \oplus 2S_R(A^3_{12})] = kc^3_j,$$

where $kc_j = MC_1^{-1} \cdot \bigoplus_{i=0}^{255} z_{i,0}$ corresponding to η_j, f_i $(0 \le i \le 3)$ defined in (15) – (18) and η_j $(1 \le j \le t)$ are randomly chosen byte differences with t to be determined. Our first observation is that there is no unknown variables on the right hand of (21) – (23), so these equations can be used directly to restore the involving variables. However, if we try to solve (21) by exhaustively searching all the possible values of A_{11}, A^0_{12}, A^1_{12} and $s^1_{18} \oplus S_1(A^2_{12}) \oplus S_1(A^3_{12})$, we need to choose $t = 7$ and the time complexity is 2^{56} steps. In order to get an efficient attack, we proceed as follows.

We regard the left part of (21) as a function of the following variables: A_{11}, A^0_{12}, A^1_{12} and $s^1_{18} \oplus S_R(A^2_{12}) \oplus S_R(A^3_{12})$. Note that there are 7 bytes involved here and if these bytes take the same value for two independent IVs, the outputs of (21) should be equal. In order to detect such an internal collision, we randomly choose a series of byte differences η_j $(1 \le j \le t)$ and compare the corresponding output kc^1_j. If a pair of IV, IV and IV', passes all the t tests, i.e., the outputs of (21) remain the same for η_j $(1 \le j \le t)$, we can conclude with high probability that the 7 bytes involved in the two equations have the same value for IV and IV'.

More precisely, given 2^{28} $(K, \overline{IV_i})$s such that

$$\forall i \ne j, (\overline{IV_i})_r = (\overline{IV_j})_r \text{ for } r = 2, 3. \qquad (24)$$

$$\forall i \ne j, (\overline{IV_i})_r \ne (\overline{IV_j})_r \text{ for } r = 0, 1. \qquad (25)$$

will guarantee that there exists such a pair. To filter out the wrong candidates, we choose $t = 8$. A wrong candidate will pass 8 consecutive tests with probability $2^{56} \cdot 2^{-64} = 2^{-8}$ which is less than 1, while the correct candidate will always pass the tests. We use the standard birthday paradox argument to detect such a pair, the time complexity is about $2^{28} \cdot 8 = 2^{31}$ steps. Now we have two IVs, IV and IV', that generate the same input values for (21), i.e.,

- $A_{11} = A'_{11}$, $A^0_{12} = A'^0_{12}$, $A^1_{12} = A'^1_{12}$.
- $s^1_{18} \oplus S_R(A^2_{12}) \oplus S_R(A^3_{12}) = s'^1_{18} \oplus S_R(A'^2_{12}) \oplus S_R(A'^3_{12})$.

We need to investigate the value evolution process of the memory registers in the first 10 rounds of initialization to derive the state information, which is shown in Table 4. In Table 4, c_i $(i = 1, 2, 3)$ are the same variables as those in Table 3.

We have the following facts on Table 4 when the $(K, \overline{IV_i})$ pair are chosen according to the conditions (24) and (25):

Table 4. The value evolution process of the memory registers in the first 10-round initialization of SNOW 3G$^{\oplus}$ (h_i ($i = 1, 2, 3$) are known constants)

	A_i	B_i	C_i
0	0	0	0
1	k_1	h_1	h_2
2	$k_2 \oplus h_1 \oplus h_2$	$MC_1[S_1(k_1)]$	h_3
3	$h_3 \oplus k_3 \oplus MC_1[S_1(k_1)]$	$MC_1[S_1(A_2)]$	$MC_2[S_2(B_2)]$
4	$B_3 \oplus C_3$ $\oplus k_0 \oplus \mathbf{1}$	$MC_1[S_1(A_3)]$	$MC_2[S_2(B_3)]$
5	$B_4 \oplus C_4$ $\oplus k_1 \oplus \mathbf{1} \oplus IV_3$	$MC_1[S_1(A_4)]$	$MC_2[S_2(B_4)]$
6	$B_5 \oplus C_5$ $\oplus k_2 \oplus \mathbf{1} \oplus IV_2$	$MC_1[S_1(A_5)]$	$MC_2[S_2(B_5)]$
7	$B_6 \oplus C_6$ $\oplus k_3 \oplus \mathbf{1}$	$MC_1[S_1(A_6)]$	$MC_2[S_2(B_6)]$
8	$B_7 \oplus C_7$ $\oplus k_0 \oplus IV_1$	$MC_1[S_1(A_7)]$	$MC_2[S_2(B_7)]$
9	$B_8 \oplus C_8 \oplus k_1$	$MC_1[S_1(A_8)]$	$MC_2[S_2(B_8)]$
10	$\underbrace{B_9 \oplus C_9 \oplus k_2}_{c_1}$	$\underbrace{MC_1[S_1(A_9)]}_{c_2}$	$\underbrace{MC_2[S_2(B_9)]}_{c_3}$
11	$c_2 \oplus c_3 \oplus k_3$ $\oplus IV_0 \oplus \Delta_i$	$MC_1[S_1(c_1)]$	$MC_2[S_2(c_2)]$
12	$B_{11} \oplus C_{11}$ $\oplus s_{16} \oplus \Delta_i$	$MC_1[S_1(A_{11} \oplus \Delta_i)]$	$MC_2[S_2(B_{11})]$
	R_1	R_2	R_3

1. $A_4 = A'_4$, $B_4 = B'_4$ and $C_4 = C'_4$, which are only determined by K.
2. $A_i = A'_i$, $B_i = B'_i$ and $C_i = C'_i$ for $i = 5, 6, 7$.
3. $A_8 \oplus A'_8 = IV_1 \oplus IV'_1$, $B_8 = B'_8$ and $C_8 = C'_8$.
4. $A_9 = A'_9$, $C_9 = C'_9$.
5. $A_{10} \oplus A'_{10} = c_1 \oplus c'_1 = MC_1[S_1(A_8)] \oplus MC_1[S_1(A'_8)]$.
6. $B_{10} = B'_{10}$, i.e., $c_2 = c'_2$.

From Table 4, $A_{11} = c_2 \oplus c_3 \oplus k_3 \oplus IV_0$ and $A_{11} = A'_{11}$, we have $c_3 \oplus c'_3 = IV_0 \oplus IV'_0$, i.e.,

$$S_2(MC_1[S_1(A_8)]) \oplus S_2(MC_1[S_1(A'_8)]) = MC_2^{-1} \cdot (IV_0 \oplus IV'_0). \qquad (26)$$

We can determine A_8 and A'_8 from (26) and $A_8 \oplus A'_8 = IV_1 \oplus IV'_1$ with 2^{32} steps. Knowing A_8 and A'_8, we can derive c_3 and c'_3. So far, we have partially recovered the internal states corresponding to IV and IV'.

3.5 Key Recovery Attack on 18-Round SNOW 3G$^{\oplus}$

Now we skip the 16 and 17 rounds case, since they are similar and go directly to the 18-round. Let us denote $F = MC_1[S_1(\cdot)]$, $G = MC_2[S_2(\cdot)]$ and $H = G[F(\cdot)]$, then from the first keystream word, we have:

$$\begin{aligned}
&\bigoplus_{i=0}^{255} F[s_{20} \oplus \alpha^{-1}\Delta_i \oplus \Delta_i \oplus H(\underline{s_{17} \oplus C_{12} \oplus \Delta_i \oplus F(A_{11} \oplus \Delta_i)}) \oplus F(\underline{\underline{s_{18} \oplus}} \\
&\underline{\underline{H(A_{11} \oplus \Delta_i) \oplus F(A_{12} \oplus \Delta_i)}}] \oplus \bigoplus_{i=0}^{255} H(\underline{\underline{s_{19} \oplus \Delta_i \oplus H(A_{12} \oplus \Delta_i)}} \oplus F(s_{17} \\
&\oplus C_{12} \oplus \Delta_i \oplus F(A_{11} \oplus \Delta_i)) \oplus \bigoplus_{i=0}^{255} F(s_{21} \oplus \Delta_i \oplus H(\underline{\underline{s_{18} \oplus \Delta_i \oplus H(A_{11}}} \\
&\underline{\underline{\oplus \Delta_i) \oplus F(A_{12} \oplus \Delta_i)}}) \oplus F(\underline{\underline{s_{19} \oplus \Delta_i \oplus H(A_{12} \oplus \Delta_i)}}) \oplus F(\underline{s_{17} \oplus C_{12} \oplus} \\
&\underline{\Delta_i \oplus F(A_{11} \oplus \Delta_i)}) = \bigoplus_{i=0}^{255} z_{i,0}.
\end{aligned} \tag{27}$$

Here by using multisets, we get rid of the LFSR words, s_{33} and s_{18}, involved in the keystream equations. Of these, s_{33} is the main obstacle to a differential analysis of the keystream equation. To have an intuitive view, we color the repeating patterns of variables in the left side of (23) in the same color. Note that we can control the values of s_{17}, s_{18}, s_{19}, s_{20} and s_{21} by properly choosing the IVs. From the following equations (28) – (33),

$$s_{16} = \alpha^{-1}(k_3 \oplus \mathbf{1}) \oplus (k_2 \oplus \mathbf{1}) \oplus \alpha(k_0 \oplus \mathbf{1}) \oplus k_3 \oplus IV_0 \tag{28}$$

$$s_{17} = \alpha^{-1}(k_0 \oplus IV_1) \oplus (k_3 \oplus \mathbf{1}) \oplus \alpha(k_1 \oplus \mathbf{1}) \oplus k_1 \oplus s_{16} \oplus h_1 \tag{29}$$

$$s_{18} = \alpha^{-1}k_1 \oplus k_0 \oplus \alpha(k_2 \oplus \mathbf{1}) \oplus k_2 \oplus F(k_1) \oplus h_1 \oplus h_2 \oplus s_{17} \tag{30}$$

$$\begin{aligned} s_{19} = {} & \alpha^{-1}k_2 \oplus k_1 \oplus \alpha(k_3 \oplus \mathbf{1}) \oplus k_3 \oplus F(k_1) \oplus F(k_2 \oplus h_1 \oplus h_2) \\ & \oplus h_3 \oplus s_{18} \end{aligned} \tag{31}$$

$$\begin{aligned} s_{20} = {} & \alpha^{-1}(k_3 \oplus IV_0) \oplus k_2 \oplus \alpha k_0 \oplus F(k_2 \oplus h_1 \oplus h_2) \oplus H(k_2) \oplus (k_0 \oplus \mathbf{1}) \\ & \oplus F(k_3 \oplus h_3 \oplus F(k_1)) \oplus s_{19} \end{aligned} \tag{32}$$

$$\begin{aligned} s_{21} = {} & \alpha^{-1}s_{16} \oplus k_3 \oplus \alpha k_1 \oplus k_1 \oplus \mathbf{1} \oplus IV_3 \oplus F(k_3 \oplus F(k_1)) \oplus H(k_2) \\ & \oplus F(A_4) \oplus s_{20}. \end{aligned} \tag{33}$$

we know that if we choose IV and IV' such that: $IV_3 = IV_3'$ and

$$\alpha^{-1}(IV_1 \oplus IV_1') \oplus (IV_0 \oplus IV_0') = 0,$$

then, we have $s_{17} = s_{17}'$, $s_{18} = s_{18}'$, $s_{19} = s_{19}'$, $s_{21} = s_{21}'$ and $s_{20} \oplus s_{20}' = \alpha^{-1}(IV_0 \oplus IV_0')$. If we undo the MC_1 on both sides of (27), we can see that in order to have a collision on the involved variables of the left side of (27), we need the following 32-bit conditions:

$$C_{12} = C_{12}'. \tag{34}$$

$$A_{11} = A_{11}'. \tag{35}$$

$$A_{12} = A_{12}'. \tag{36}$$

There are 96 bits involved here, so 2^{48} random (K, IV) pairs satisfying the above specified conditions will ensure that there exists such a collision with high probability. Then injecting the byte differences λ_j $(1 \leq j \leq 8)$ into the second most significant byte of the involved variables will preserve the collision. Such a collision can be detected by changing these byte differences and comparing the most and second most significant bytes of the corresponding keystream words xors. The time complexity of this detection is $2^{48} \cdot 8 \cdot 2 = 2^{52}$ steps. Then from Table 4 and (34) – (36), we have $c_1 = c_1'$, i.e.,

$$k_2 \oplus F(A_8) \oplus G(B_8) = k_2 \oplus F(A_8') \oplus G(B_8'). \tag{37}$$

From Table 4, B_8 is determined by B_6 and C_6 which are the same when the key and IV_3 are fixed, and thus $B_8 = B_8'$. Further, from (37), we get $A_8 = A_8'$, i.e.,

$$\begin{aligned} F(A_6) \oplus F(A_6') = IV_1 \oplus IV_1' \Rightarrow F(k_2 \oplus \mathbf{1} \oplus IV_2 \oplus F(A_4) \oplus G(B_4)) \oplus \\ F(k_2 \oplus \mathbf{1} \oplus IV_2' \oplus F(A_4) \oplus G(B_4)) = IV_1 \oplus IV_1'. \end{aligned} \tag{38}$$

Let $k_2 \oplus F(A_4) \oplus G(B_4) \oplus \mathbf{1} = V$ which is an unknown constant, then from (38) we get V in 2^{32} steps. Then we know $A_6 = V \oplus IV_2$ and $A_6' = V \oplus IV_2'$.

From $A_{12} = A_{12}'$, $c_1 = c_1'$ and $B_{11} = B_{11}'$, we have $G(c_2) \oplus G(c_2') = IV_0 \oplus IV_0'$, see Table 4. Again from Table 4, we have

$$H(k_1 \oplus B_8 \oplus H(A_6)) \oplus H(k_1 \oplus B_8 \oplus H(A_6')) = IV_0 \oplus IV_0'. \tag{39}$$

So we can derive $k_1 \oplus B_8$ from (39) in 2^{32} steps. Combining $k_1 \oplus B_8$ with A_6 and A_6', we get c_2 and c_2'. Then from $A_{11} = A_{11}'$, we get $c_3 \oplus c_3' = IV_0 \oplus IV_0' \oplus c_2 \oplus c_2'$, i.e.,

$$\begin{aligned} H(IV_1 \oplus F(A_6) \oplus k_0 \oplus G(B_6)) \oplus H(IV_1' \oplus F(A_6') \oplus k_0 \oplus G(B_6)) \\ = IV_0 \oplus IV_0' \oplus c_2 \oplus c_2'. \end{aligned} \tag{40}$$

From (40), we can get $k_0 \oplus G(B_6)$ in 2^{32} steps. Thus, we know A_8 and A_8', c_3 and c_3'. So far, the information restored is shown in Table 5, where ♣ means recovered and ♢ means partially recovered.

Table 5. The register values restored

	5	6	7	8	9	10	11
R_1		♣		♣	♣		
R_2		♢	♣	♢	♣	♣	
R_3				♣		♣	♣

In order to recover the key, we recall the above attack once with a different value of $IV_3 = IV_3'$, i.e., we choose another set of 2^{48} random $(K, \overline{IV})$ pairs such that the key K is the same as before and the $\overline{IV}$s satisfy the same conditions, but with a different set of values. Then the above analysis process also applies to the second set. Our observation is that the values of the registers in the FSM in the second set case are highly correlated to those in the first set case. For example, $k_2 \oplus F(A_4) \oplus G(B_4) \oplus \mathbf{1} = V$ is the same, since it only depends on the key K. From the values of $k_0 \oplus G(B_6)$, we have

$$k_0 \oplus G(B_6) = const1. \tag{41}$$

$$k_0 \oplus G(\overline{B_6}) = const2. \tag{42}$$

From (41) and (42), we have

$$H(IV_3 \oplus \mathbf{1} \oplus k_1 \oplus B_4 \oplus C_4) \oplus H(\overline{IV_3} \oplus \mathbf{1} \oplus k_1 \oplus B_4 \oplus C_4) = const3. \tag{43}$$

From (43), we can restore $k_1 \oplus B_4 \oplus C_4$ in 2^{32} steps, which in turn gives us A_5 and $\overline{A_5}$, B_6 and $\overline{B_6}$. Combining these values with A_8 and $\overline{A_8}$ which are known from the corresponding individual analysis, we can recover k_0 successfully. We can use similar procedures to recover the other key words. The total time complexity is $2^{52} \cdot 2 = 2^{53}$ steps and the data complexity is $2^8 \cdot 2^{48} \cdot 2 = 2^{57}$ keystream words.

4 Conclusions

In this paper, we have shown chosen IV resynchronization attacks on SNOW 3G and SNOW $3G^{\oplus}$. We show full key-recovery attacks on up to 18 out of 33 initialization rounds of SNOW $3G^{\oplus}$ using a *multiset collision* idea. We also show 13-round distinguisher of 2^8 complexity for the actual SNOW 3G. Practical parts of all these attacks have been verified experimentally on a PC. Our results show that about half of the initialization rounds of SNOW 3G might succumb to chosen IV resynchronization attacks. The remaining security margin however is quite significant and thus these attacks pose no threat to the security of SNOW 3G.

Acknowledgements. We would like to thank the anonymous reviewers for very helpful comments. Bin Zhang was with State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China and supported by the key programm of the National Natural Science Foundation of China (Grant No. 60833008) and the general programm of the National Natural Science Foundation of China (Grant No. 60603018).

References

1. Biryukov, A., Shamir, A.: Structural Cryptanalysis of SASAS. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 394–405. Springer, Heidelberg (2001)
2. Billet, O., Gilbert, H.: Resistance of SNOW 2.0 Against Algebraic Attacks. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 19–28. Springer, Heidelberg (2005)
3. Gilbert, H., Minier, M.: A Collision Attack on 7 Rounds of Rijndael. In: AES Candidate Conference 2000, pp. 230–241 (2000)

4. Ekdahl, P., Johansson, T.: A New Version of the Stream Cipher SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 37–46. Springer, Heidelberg (2003)
5. ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification, version 1.1 (September 2006), `http://www.3gpp.org/ftp/`
6. ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 5: Design and Evaluation Report, version 1.1 (September 2006), `http://www.3gpp.org/ftp/`
7. Nyberg, K., Wallén, J.: Improved Linear Distinguishers for SNOW 2.0. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 144–162. Springer, Heidelberg (2006)
8. Watanabe, D., Biryukov, A., De Canniére, C.: A Distinguishing Attack of SNOW 2.0 with Linear Masking Method. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 222–233. Springer, Heidelberg (2004)

A Multiset Difference Propagation Table

Table 6. Multiset difference propagation in 10-round initialization of SNOW 3G$^{\oplus}$

	s_{15}	s_{14}	s_{13}	s_{12}	s_{11}	s_{10}	s_9	s_8	s_7	s_6	s_5	s_j $(0\leq j\leq 4)$	R_i $(i=1,2,3)$
0	Δ_i	0	0	0	0	0	0	0	0	0	0	0	0
1	Δ_i	Δ_i	0	0	0	0	0	0	0	0	0	$\vdots$	$\vdots$
2	Δ_i	Δ_i	Δ_i	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
3	Δ_i	Δ_i	Δ_i	Δ_i	0	0	0	0	0	0	0	$\vdots$	$\vdots$
4	Δ_i	Δ_i	Δ_i	Δ_i	Δ_i	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
5	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	Δ_i	Δ_i	Δ_i	Δ_i	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
6	Δ_i	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	Δ_i	Δ_i	Δ_i	Δ_i	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
7	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	Δ_i	Δ_i	Δ_i	Δ_i	0	$\vdots$	$\vdots$	$\vdots$	$\vdots$
8	Δ_i	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	Δ_i	Δ_i	Δ_i	Δ_i	0	$\vdots$	$\vdots$	$\vdots$
9	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	Δ_i	Δ_i	Δ_i	Δ_i	0	$\vdots$	$\vdots$
10	$\Delta_i\oplus\alpha^{-2}\Delta_i$	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	$\Delta_i\oplus\alpha^{-1}\Delta_i$	Δ_i	Δ_i	Δ_i	Δ_i	Δ_i	0	0

High Performance GHASH Function for Long Messages

Nicolas Méloni[1], Christophe Négre[2], and M. Anwar Hasan[1]

[1] Department of Electrical and Computer Engineering
University of Waterloo, Canada
[2] Team DALI/ELIAUS
University of Perpignan, France

Abstract. This work presents a new method to compute the GHASH function involved in the Galois/Counter Mode of operation for block ciphers. If $X = X_1 \dots X_n$ is a bit string made of n blocks of 128 bits each, then the GHASH function effectively computes $X_1 H^n + X_2 H^{n-1} + \dots X_n H$, where H is an element of the binary field $\mathbb{F}_{2^{128}}$. This operation is usually computed by using n successive multiply-add operations over $\mathbb{F}_{2^{128}}$. In this work, we propose a method to replace all but a fixed number of those multiplications by additions on the field. This is achieved by using the characteristic polynomial of H. We present both how to use this polynomial to speed up the GHASH function and how to efficiently compute it for each session that uses a new H.

Keywords: Galois/Counter mode, GHASH function, characteristic polynomial.

1 Introduction

The Galois/Counter mode (GCM) is one of the modes of operation recommended by the National Institute of Standards and Technology (NIST) [10]. As an authenticated encryption mode, it generates both a cipher text and an authentication tag for each session. The cipher text is obtained by using the counter mode of encryption (CRT) with a 128 bit block cipher algorithm (usually AES). The authentication part is performed by using a universal hashing (GHASH) based on multiplication in the binary field $\mathbb{F}_{2^{128}}$ [9]. More precisely, if $X = X_1 \dots X_n$ is a bit stream divided into 128 bit blocks, the authentication process computes $X_1 H^n + X_2 H^{n-1} + \dots + X_n H$, where H depends on the encryption key.

High speed GCM designs are usually based on both fast implementations of the block cipher [2,4,6,16] and bit-parallel multiplier over $\mathbb{F}_{2^{128}}$ [1,3,8,11]. One can refer to [7,13,14,15] for various implementations of the AES-GCM mode of encryption.

In practice, the $\mathbb{F}_{2^{128}}$ multiplier operates faster than the block cipher. However, it is easy to speed up the block cipher computation by taking advantage of the natural parallelism of the CRT mode. In that case, the $\mathbb{F}_{2^{128}}$ multiplier's critical path of the GHASH function becomes the bottleneck. To overcome this problem, it is always possible to parallelize the GHASH computation process as suggested in [9]. Such solutions imply a proportional increase in the number of multipliers.

In this paper, we propose a new way to speed up the GHASH function for long messages, by moving the bottleneck from the multiplier to that of one XOR and one AND gate. This is achieved by first computing the characteristic polynomial of H and then performing the computation of the GHASH function modulo this polynomial. Effectively, we replace the finite field multiplication by a faster operation. In addition, our method can be parallelized to further reduce the GHASH computation time.

The remainder of this paper is organized as follow: Section 2 briefly describes the functioning of GHASH and the implementation of the binary field $\mathbb{F}_{2^{128}}$. In section 3, we describe our approach to use characteristic polynomials to compute GHASH and a parallelization of this operation using multiple polynomial reduction units. In section 4, we present how to efficiently compute a characteristic polynomial over $\mathbb{F}_{2^{128}}$. Finally, a few concluding remarks are made in section 5.

2 GHASH Authentication Function

GCM uses two main operations: encryption and authentication. Encryption is performed using a block cipher encryption function whereas the authentication mainly requires the use of multiplication over $\mathbb{F}_{2^{128}}$. One can find a full description of the encryption process in [9]. Below we only describe the authentication part.

2.1 Authentication Function

Let $X = X_1X_2 \dots X_n$ be a bit string divided into 128 bit blocks (X_n might be padded with 0's) and H the 128 bit hash sub-key. The GHASH function performs as follow:

Algorithm 1. GHASH function

```
Require: $X$, $H$
Ensure: $GHASH_H(X)$
  $Y \leftarrow 0$
  for $i = 1$ to $n$ do
    $Y \leftarrow (Y + X_i) \times H$
  end for
  return $Y$
```

One can verify that Algorithm 1 computes $X_1H^n + X_2H^{n-1} + \cdots + X_nH$. In practice, X is obtained as the concatenation of the cipher text, the authentication data and the length of those two data, and H is generated by applying the block cipher to the zero block. Note that Algorithm 1 requires n multiplications and n additions on the field $\mathbb{F}_{2^{128}}$. Assuming that we have only one multiplier and one adder, these operations would be performed in sequence and the total

Table 1. Complexities of various multipliers over $\mathbb{F}_{2^m}$ where D_X (resp. D_A) is the delay of a two-input XOR (resp. AND) gate

Multiplier	#AND	#XOR	Gate delay (D_M)
CRT [1]	$O(m^{1.6})$	$O(m^{1.6})$	$O(m)D_X + 4m^{0.4}D_A$
Karatsuba [11]	$O(m^{1.58})$	$O(m^{1.58})$	$(3\log_2(m)+1)D_X + D_A$
Fan-Hasan [3]	$O(m^{1.58})$	$O(m^{1.58})$	$(2\log_2(m)+1)D_X + D_A$
Mastrovito [8]	$O(m^2)$	$O(m^2)$	$\log_2(m)D_X + D_A$

computation time for GHASH is approximately n times the combined delay of a multiplication and an addition over the field. On binary fields, the delay of an addition is exactly that of a bitwise XOR operation. In Table 1, we summarize various bit-parallel multiplication methods and their space and time complexities in terms of gate counts and gate delays.

2.2 Underlying Computations

In GCM specifications, the binary field $\mathbb{F}_{2^{128}}$ is defined as $\mathbb{F}_2[x]/(x^{128} + x^7 + x^2 + x + 1)$. That is to say, the set of residues of polynomials with coefficients in $\mathbb{F}_2$ modulo $x^{128} + x^7 + x^2 + x + 1$. An element A of this field can be easily represented as a vector of length 128 with coefficient in $\mathbb{F}_2$.

In that case, addition is very simple as it just consists of a bitwise XOR between the two summands. Multiplication can be performed by a succession of shifts and additions. Depending on the platform on which GCM is implemented, it is possible to improve this operation, as shown in [9] or [3] for instance.

One common feature among several of the previously reported methods for improving GCM is that all efforts are made to optimize the field multiplication only. In this work, we propose a way to improve the efficiency of the overall computation of $\mathrm{GHASH}_H(X) = X_1H^n + X_2H^{n-1} + \cdots + X_nH$.

3 Computing GHASH Using a Characteristic Polynomial

In order to easily present our method, here we consider a bit string, divided into n blocks: $X_1X_2\ldots X_n$ and an element H of $\mathbb{F}_{2^{128}}$. Our goal is to compute $\mathrm{GHASH}_H(X) = X_1H^n + X_2H^{n-1} + \cdots + X_nH$. As $H \in \mathbb{F}_{2^{128}}$ there exists a polynomial χ_H of degree 128 (at most), with coefficients in $\mathbb{F}_2$, such that $\chi_H(H) = 0$, called the characteristic polynomial of H in $\mathbb{F}_2$. This section shows how to use such a polynomial to limit the number of field multiplications involved in the polynomial evaluation.

3.1 Using χ_H

Let $H \in \mathbb{F}_{2^m}$ and $\chi_H(T) = \Pi_{i=0}^{m-1}(T + H^{2^i})$ be a polynomial in T of degree m. The polynomial $\chi_H(T)$ is called the characteristic polynomial of H and satisfies $\chi_H(H) = 0$ and has its coefficients in $\mathbb{F}_2$.

Hence $P = X_1H^n + X_2H^{n-1} + \cdots + X_nH = X_1H^n + X_2H^{n-1} + \cdots + X_nH \bmod \chi_H$ is considered as a polynomial in H with coefficients in $\mathbb{F}_{2^m}$. Let us now write χ_H as $\sum_{i=0}^{m} c_iT^i$. It can be shown that $c_m = c_0 = 1$. Using the fact that $H^m = \sum_{i=0}^{m-1} c_iT^i$, it is possible to reduce a polynomial in H of degree n to a polynomial of degree $m-1$. As an example, let $P = X_1H^{128} + X_2H^{127} + \cdots + X_{128}H$ with $X_i \in \mathbb{F}_{2^m}$, where $m = 128$. Then $P = (X_2 + c_{127}X_1)H^{127} + (X_3 + c_{126}X_1)H^{126} + \cdots + (X_{128} + c_1X_1)H + c_0X_1$. It is important to note that all the c_i's are in $\mathbb{F}_2$, which means that computing $(X_{m-i+1} + c_iX_i)$ is just an addition over $\mathbb{F}_{2^{128}}$ or a simple parameter assignment. Moreover, all these additions are independent and can all be performed in parallel. Figure 1 shows a diagram of a possible implementation of this operation in hardware. The output of the polynomial reduction unit (PRU) is a polynomial of degree $m-1$ in H. We can take the output of the PRU to an multiply-and-add unit to apply the Horner scheme and eventually obtain the desired GHASH value, which is an element of $\mathbb{F}_{2^{128}}$ (see Figure 2).

A generalized description of the operation of the structure of Figure 2 is given in Algorithm 2, where a polynomial P of degree $n \geq m$ can be computed using at most $m-1$ field multiplications, replacing each of the remaining $n-m$ multiplications by a maximum of m parallel field additions.

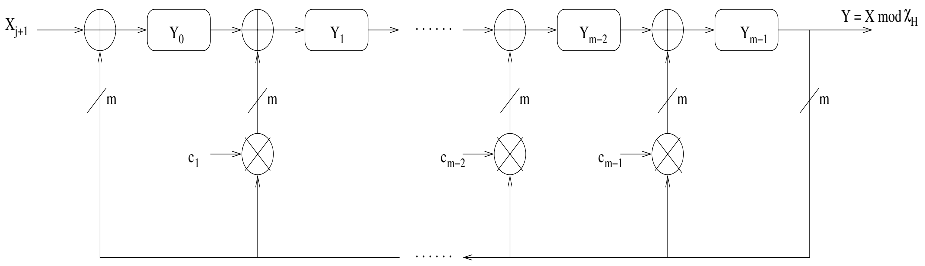


Fig. 1. Implementation of the Polynomial Reduction Unit (PRU)

Algorithm 2 computes P in two steps:

– first it computes the remainder of P modulo χ_H using a shift-and-add algorithm,
– then it effectively calculates P using the Horner scheme.

Complexity: For long messages, the value of n is expected to be much longer than that of m, e.g. if the size of X is 1 MBytes, then $n = 2^{16}$, and in Algorithm 2, the upper *for* loop will dominate in terms of computational cost. As noted previously, one important feature of Algorithm 2 is that all the additions involved in the upper *for* loop are independent. Moreover, the c_i's are elements of $\mathbb{F}_2$, which means that the operation $Y_{i-1} + c_iC$ is just a bitwise XOR operation. In the end, each step of this loop requires m bitwise XOR operations. Thus, our method trades $n-m$ multiplications by H over $\mathbb{F}_{2^m}$ against $m(n-m)(W_H-1)$ bit level XOR operations, where W_H is the number of non-zero coefficients of χ_H.

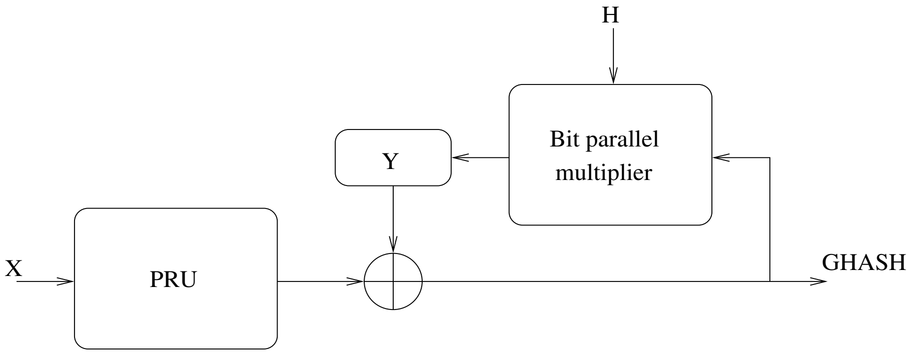


Fig. 2. Implementation of the GHASH function using a PRU

Algorithm 2. $GHASH_H(X)$ using χ_H

```
**Require:** $X = X_1X_2 \dots X_n$ and $\chi_H(T) = \sum_{i=0}^{m} c_i T^i$
**Ensure:** $GHASH_H(X) = X_1H^n + X_2H^{n-1} + \cdots + X_nH$
  $Y_{m-1} \dots Y_0 \leftarrow X_1 \dots X_m$, $GHASH = 0$
  **for** $j = m$ to $n$ **do**
    $C = Y_{m-1}$
    $Y_i \leftarrow Y_{i-1} + c_iC, 1 \leq i \leq m-1$  } (in parallel)
    $Y_0 = X_{j+1} + c_0C$                            }
  **end for**
  **for** $i = m-1$ down to 1 **do**
    $GHASH = (GHASH + Y_i) \times H$
  **end for**
  **return** $(GHASH + Y_0)$
```

In the case of a hardware implementation, on a parallel architecture, a step of the loop would require m^2 XOR gates, all used in parallel, and the gate delay is $D_X + D_A$ (delay of one XOR and one AND gates). In the end, the implementation of the GHASH function as shown in Figure 2 has a space complexity of $O(m^2)$ and the total time delay for the upper loop is $(n-m)(D_X + D_A)$. The final $m-1$ field multiplications can be performed using any of the already available methods as listed in Table 1.

Comparison: Let n be the length of the message to be treated (considered as a sequence of m bit blocks). Implementations based on Algorithm 1 performs, at each step of the algorithm, one field multiplication and one block bitwise XOR operation. Denoting D_M as the delay of a field multiplication, the total delay is:

$$n(D_M + D_X).$$

On the other hand, Algorithm 2 first requires the computation of the characteristic polynomial. We denote the cost of this computation as D_χ. It is important to remark that this cost is constant, i.e. independent of n. Then, Algorithm 2 performs m^2 parallel bitwise XOR operations in each pass of the upper loop.

Finally, the algorithm ends by computing $(m-1)$ field multiplications and m bitwise XOR operations on m bit blocks. Assuming that the m Y-registers of the PRU can be initialized in parallel, the total delay can then be approximated as:

$$D_\chi + (n-m)(D_X + D_A) + (m-1)(D_M + D_X).$$

Asymptotically (i.e. for large n), the new method reduces the critical path delay of the GHASH function to that of one XOR plus one AND operation (i.e. $O(1)$), whereas that of the traditional method is due to the multiplier, i.e. $O(\log m)$ for bit-parallel implementations. This reduction in the critical path is obtained at the cost of a PRU. However, the number of multiplications in the final Horner scheme being constant, using a smaller but slower multiplier can greatly reduce the additional cost, in terms of space, without changing the overall asymptotic complexity.

3.2 Delay Reduction with Multiple PRUs

Let $n = 2n'$ be an even integer (in order to simplify notations). Then one can write

$$\begin{aligned} P &= X_1 H^n + X_2 H^{n-1} + \cdots + X_n H & (1)\\ &= \left(X_1 (H^2)^{n'-1} + X_3 (H^2)^{n'-2} + \cdots + X_{n-1}\right) H^2 \\ &\quad + \left(X_2 (H^2)^{n'-1} + X_4 (H^2)^{n'-2} + \cdots + X_n\right) H \\ &\equiv P_1 H^2 + P_2 H. & (2) \end{aligned}$$

Now assume that we have two PRUs whose feedback connections are defined by the characteristic polynomial of H^2 over $\mathbb{F}_2$. Then these PRUs can process even and odd numbered blocks of X in parallel in $\frac{n}{2} - m$ steps. Referring to (2), the outputs of PRUs are $P_1 \mod \chi_{H^2}$ and $P_2 \mod \chi_{H^2}$, that we call partial GHASH. One can then simply multiply and add according to the equality $P \equiv P_1 H^2 + P_2 H$.

More generally, let us assume that we have r PRUs whose feedback connections are defined by the characteristic polynomial of H^r over $\mathbb{F}_2$. For simplicity, we also assume that n is a multiple of r (we pad zeros to X if needed). Then we can decompose X into r different sets $P_1, P_2, \ldots, P_r$ such that

$$P = P_1 H^r + P_2 H^{r-1} + \cdots + P_r H. \qquad (3)$$

where for $1 \le i \le r$ we have used $P_i = X_i (H^r)^{\frac{n}{r}-1} + X_{i+r} (H^r)^{\frac{n}{r}-2} + \cdots + X_{n-r+i} (H^r)^0$.

Let $P_i' = P_i \mod \chi_{H^r}$. Using r PRUs operating concurrently, we can reduce $P_i \mod \chi_{H^r}$ to obtain the corresponding P_i', $\forall i$, in $\frac{n}{r}$ iterations as shown in the left part of Fig. 3. (The number of iterations reduces to $\frac{n}{r} - m$ if each PRU can be initialized with m coefficients in parallel at the beginning of its operation.) We now write (3) in terms of P_i' as follows:

$$P \equiv P_1' H^r + P_2' H^{r-1} + \cdots + P_r' H. \qquad (4)$$

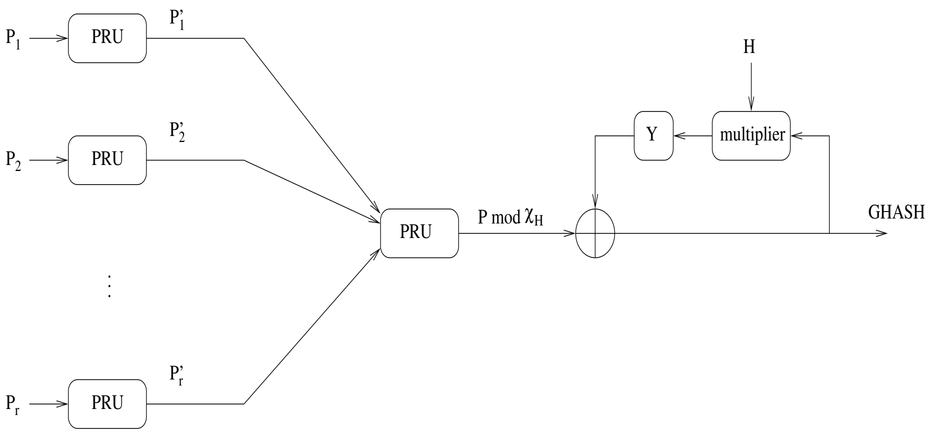


Fig. 3. Implementation of the GHASH function using multiple PRUs

Since each P'_i is a degree $m-1$ polynomial in H^r, one can verify that the sum of the products in (4) is nothing but a degree rm polynomial in H (with the constant term being zero). We can then use one more PRU whose feedback connection is defined by the characteristic polynomial of H and reduce the polynomial in H from degree rm to degree $m-1$. Note that computing this characteristic polynomial, unlike that of H^r, can be done in parallel with obtaining P'_i. Moreover, if r is a power of two, then H^r and H share the same characteristic polynomial and hence the latter does not need to be computed a second time. We obtain the final result by applying the Horner scheme to the output of the final PRU and this requires $m-1$ multiply-and-add operations.

Hence, given χ_{H^r} and using $r+1$ PRUs and one multiplier (see Fig. 3), the GHASH computation of X has the following time delay

$$\left(\frac{n}{r}+rm+1\right)(D_X+D_A)+(m-1)(D_M+D_X). \tag{5}$$

Comparison: As noted in [9], the conventional method of computing GHASH can also be parallelized. Assume that we have r field multipliers. Then all P_i, for $1 \le i \le r$, can be computed in $\frac{n}{r}-1$ iterations, assuming that we have $H, H^2, \ldots, H^r$, and the time delay of each iteration is D_M+D_X. With one additional delay of D_M, one can compute P_iH^{r-i+1}, $1 \le i \le r$. Finally P is obtained as $\sum_{i=1}^{r} P_iH^{r-i+1}$, which incurs a delay of $(\log_2 r)D_X$, assuming additions are done in parallel in a binary tree fashion. Thus, given $H^i, 1 \le i \le r$, using r field multipliers and $r-1$ adders, one can compute GHASH based on Algorithm 1 with the following time delay:

$$\left(\frac{n}{r}-1\right)(D_M+D_X)+D_M+(\log_2 r)D_X \tag{6}$$

In hardware implementation, D_X is normally larger than D_A, and more importantly, for bit parallel implementation we have $D_M \geq (\log_2 m)D_X$. For example, as mentioned in Table 1, the Karatsuba algorithm based bit-parallel multiplier over $\mathbb{F}_{2^{128}}$ has $D_M > 7D_X$. Thus for a large n (i.e., long messages), it is clear that (5) is considerably smaller than (6).

4 Computation of the Characteristic Polynomial over $\mathbb{F}_{2^{128}}$

In this section, we first recall a method from Gordon [5] that determines the characteristic polynomial of an element of a finite field and requires, in our context, 127 multiplications and 127 squarings. We then propose a new method taking advantage of the tower structure of $\mathbb{F}_{2^{128}}$ that can be faster than the first one depending on the representation of the finite field.

Gordon's method: Let $A \in \mathbb{F}_{2^m}$. Then the characteristic polynomial of A is given by

$$\chi_A(T) = \prod_{i=0}^{m-1} (T + A^{2^i}).$$

We want to find the polynomial in the form $\sum_{i=0}^{m} B_i T^i$, with $B_i \in \mathbb{F}_2$. Gordon's method is based on the observation that $\chi_A(x) = \prod_{i=0}^{m-1} (x + A^{2^i}) = \sum_{i=0}^{m} B_i x^i$. In other words, the coefficients of the representation of field element $\chi_A(x)$ correspond to those of the polynomial $\chi_A(T) \mod \chi_x(T)$. As $\chi_A(T)$ has degree 128, we have $\chi_A(T) = (\chi_A(T) \mod \chi_x(T)) + \chi_x(T)$. This can be easily evaluated by adding $\chi_A(x)$ to $\chi_x(x)$ modulo 2. Evaluating $\chi_A(x)$ can be done using 127 field multiplications and 127 fields squaring.

Gordon's method is quite general and can be applied to any binary field. We now propose a new method that takes into account that $\mathbb{F}_{2^{128}}$ has a very special structure. We will use the following lemma to compute the polynomial χ_A. Let $P = \sum_{i=0}^{d} p_i T^i$ be a polynomial in $\mathbb{F}_{2^n}[T]$ and k an integer. We denote

$$\sigma_k(P) = \sum_{i=0}^{d} p_i^{2^k} T^i.$$

Lemma 1. *Let $\mathbb{F}_{2^m}$ be a binary field, and let $\mathbb{F}_{2^{2m}}$ a degree 2 field extension of $\mathbb{F}_{2^m}$. The following assertions hold*

1. *If $P, Q \in \mathbb{F}_{2^m}[T]$ and k an integer then*

$$\sigma_k(PQ) = \sigma_k(P)\sigma_k(Q).$$

2. *If $P \in \mathbb{F}_{2^{2m}}[T]$ satisfies $P(A) = 0$, then the polynomial $Q = P\sigma_m(P)$ satisfies*

$$Q(A) = 0 \text{ and } Q \in \mathbb{F}_{2^m}[T].$$

Proof. 1. Let us write $P = \sum_{i=0}^{d} p_i T^i$ and $Q = \sum_{i=0}^{d'} q_i T^i$ then

$$PQ = \sum_{i=0}^{d} \sum_{j=0}^{d'} p_i q_i T^{i+j}.$$

Then, we apply σ_k to PQ

$$\begin{aligned} \sigma_k(PQ) &= \sigma_k \left(\sum_{i=0}^{d} \sum_{j=0}^{d'} p_i q_i T^{i+j} \right) \\ &= \sum_{i=0}^{d} \sum_{j=0}^{d'} (p_i q_i)^{2^k} T^{i+j} \\ &= \sum_{i=0}^{d} \sum_{j=0}^{d'} p_i^{2^k} q_i^{2^k} T^{i+j} \\ &= \sigma_k(P)\sigma_k(Q) \end{aligned}$$

which proves the assertion.

2. We now prove the second assertion of the lemma. Let $Q = P\sigma_{2^m}(P)$. We first check that $Q(A) = 0$, we have

$$Q(A) = P(A)\sigma_q(P)(A) = 0 \times \sigma_q(P)(A) = 0$$

In order to show that $Q \in \mathbb{F}_{2^m}[T]$ we have to prove that $\sigma_m(Q) = Q$, since this would mean that each coefficient of Q is in $\mathbb{F}_{2^m}$. We compute

$$\sigma_m(Q) = \sigma_m(P)\sigma_m(\sigma_m(P))$$

But $\sigma_m((\sigma_m(P)) = \sigma_{2m}(P) = P$ since $P \in \mathbb{F}_{2^{2m}}[T]$. Finally

$$\sigma_m(Q) = P\sigma_m(P) = Q$$

This completes the proof.

Let us now see, how to use the above result to compute the characteristic polynomial of an element $A \in \mathbb{F}_{2^8}$ over $\mathbb{F}_2$. We begin with the polynomial $P_0 = T + A$ which satisfies $P(A) = 0$ and $P \in \mathbb{F}_{2^8}[T]$.

– Now we apply the lemma for the field extension $\mathbb{F}_{2^8}/\mathbb{F}_{2^4}$ to obtain a polynomial $P_1 = P_0\sigma_4(P_0)$ which satisfies

$$P_1(A) = 0 \text{ and } P_1 \in \mathbb{F}_{2^4}[T]$$

Moreover we have $P_1 = (T + A)(T + A^{2^4})$.

– We apply again the lemma for the field extension $\mathbb{F}_{2^4}/\mathbb{F}_{2^2}$ to obtain a polynomial $P_2 = P_1\sigma_2(P_1)$ which satisfies

$$P_1(A) = 0 \text{ and } P_1 \in \mathbb{F}_{2^2}[T].$$

Moreover we also have $P_2 = (T + A)(T + A^{2^4})(T + A^{2^2})(T + A^{2^6})$.

– Finally we apply the lemma a third time for the field extension $\mathbb{F}_{2^2}/\mathbb{F}_2$ to the polynomial P_2. We get

$$\begin{aligned} P_3 &= P_2\sigma_1(P_2) \\ &= \underbrace{(T+A)(T+A^{2^4})(T+A^{2^2})(T+A^{2^6})}_{P_2} \\ &\quad \times \underbrace{(T+A^2)(T+A^{2^5})(T+A^{2^3})(T+A^{2^7})}_{\sigma_2(P_2)}, \end{aligned}$$

and $P_3 \in \mathbb{F}_2[T]$ and satisfies $P_3(A) = 0$. If we look at the expression of P_3 we can see that it is equal to the characteristic polynomial of A.

This method is generalized for binary field $\mathbb{F}_{2^m}$ where $m = 2^k$ is a power of 2 in Algorithm 3.

Algorithm 3. Computing the characteristic polynomial of A

```
Require: $A \in \mathbb{F}_{2^{2^k}}$
Ensure: $P_A$ (the minimal polynomial of $A$)
  $P \leftarrow T + A$
  for $i = k-1$ to 0 do
    $P \leftarrow P \times \sigma_{2^i}(P)$
  end for
  return $(P)$
```

Proposition 1. *Algorithm 3 returns the characteristic polynomial of A over $\mathbb{F}_2$.*

Proof. Applying Lemma 1, for successive extension field of degree 2 shows that the returned polynomial P satisfies $P(A) = 0$ and $P \in \mathbb{F}_{2^m}$. Let us now prove that the polynomial P is the characteristic polynomial of A, i.e., we show that the returned P satisfies

$$P(T) = \prod_{i=0}^{m-1}(T + A^{2^i}).$$

Specifically, we will show by induction on the index of the loop that the computed P in the jth loop is the characteristic polynomial of A over the field $\mathbb{F}_{2^{k-j}}$.

– For $j = 0$ this is clear that $P = T + A$ is the characteristic polynomial of A over $\mathbb{F}_{2^k}$.
– For $j = 1$ we have $P = (T + A)(T + A^{2^{k-1}})$, thus the assertion is true.
– Suppose that $P = \prod_{i=0}^{j-1}(T + A^{d^i})$ where $d = 2^{k-j}$, then

$$P\sigma_{j+1}(P) = \left(\prod_{i=0}^{j-1}(T + A^{d^{2i}})\right)\left(\prod_{i=0}^{j-1}(T + A^{d^{2i+1}})\right)$$

where $d = 2^{k-(j+1)}$. This completes the proof.

Complexity

Let us now evaluate the complexity of Algorithm 3. We suppose that each polynomial multiplication is done using the Karatsuba method, and each exponentiation to 2^k is done by computing k successive squares.

First, we make two observations:

– Let us first check that the degree of P after j iterations of the *for* loop has degree 2^j. This is true when $j = 0$ since P is initialized by $T + A$. In each loop the degree of P is multiplied by 2, thus after j loop, the degree of P must be equal to 1×2^j.

– At the end of the j-th iteration of the *for* loop, the polynomial P belongs to $\mathbb{F}_{2^{k-j}}$. Indeed, for $j = 1$, $P = (T + A)\sigma_{2^{2^{k-1}}}(T + A)$ with $A \in \mathbb{F}_{2^{2^k}}$, so $P \in \mathbb{F}_{2^{2^{k-1}}}[T]$ from Lemma 1. If we suppose that after j iterations $P \in \mathbb{F}_{2^{2^{k-j}}}[T]$, then, at step $j + 1$ we compute $P = P \times \sigma_{2^{k-(j+1)}}(P)$, which belongs to $P \in \mathbb{F}_{2^{2^{k-(j+1)}}}[T]$.

From the above two remarks, we can now evaluate the complexity of Algorithm 3.

– At the j-th iteration of the *for* loop, P is a monic polynomial of degree 2^j (and thus has 2^j coefficients). As its coefficients are in $\mathbb{F}_{2^{2^{k-j+1}}}$ and that we need to compute the 2^{k-j}-th power of each of them, we have to perform $2^j \times 2^{k-j}$ squarings over $\mathbb{F}_{2^{2^{k-j+1}}}$ to compute $\sigma_{2^{k-j}}(P)$. Let us denote S_{2^k} as the cost of a squaring over $\mathbb{F}_{2^{2^k}}$, we assume that $S_{2^k} = 2S_{2^{k-1}}$ (squaring is linear over binary fields). Then, the total complexity of computing $\sigma_{2^{k-j}}(P)$ is

$$\begin{aligned}\sum_{j=1}^{k} 2^j 2^{k-j} S_{2^{k-j+1}} &= \sum_{j=1}^{k} 2^k \frac{S_{2^k}}{2^{j-1}} \\ &= 2^k S_{2^k} \sum_{j=1}^{k} \frac{1}{2^{j-1}} \\ &= 2^k S_{2^k}(2 - 2^{-(k-1)}) \\ &= 2^{k+1} S_{2^k}(1 - 2^{-k}) \\ &\sim 2^{k+1} S_{2^k}\end{aligned}$$

– In the same manner, at the j-th iteration, after computing $\sigma_{2^{k-j}}(P)$, the algorithm computes $P\sigma_{2^{k-j}}(P)$. Thus, we multiply two degree 2^j polynomials. Performing this multiplication using the Karatsuba algorithm requires 3^j multiplications of fields elements. Let us denote M_{2^k} as the cost of a multiplication over $\mathbb{F}_{2^{2^k}}$, we assume that $M_{2^k} = 3M_{2^{k-1}}$. Then, the computational cost is equal to $\sum_{j=1}^{k} 3^j M_{2^{k-j+1}}$, so we obtain

$$\begin{aligned}\sum_{j=1}^{k} 3^j M_{2^{k-j+1}} &= \sum_{j=1}^{k} 3^j \frac{M_{2^k}}{3^{j-1}} \\ &= \sum_{j=1}^{k} 3M_{2^k} \\ &= 3kM_{2^k}\end{aligned}$$

In the end, the overall cost of computing the characteristic polynomial of an element of $\mathbb{F}_{2^{2^k}}$ over $\mathbb{F}_2$ is, in terms of number of operations over $\mathbb{F}_{2^{2^k}}$ is $3kM_{2^k} + 2^{k+1}S_{2^k}$.

In the case of GCM, $k = 7$ and thus the number of field operations is 21 multiplications and 256 squarings over $\mathbb{F}_{2^{128}}$.

Remark 1. It is not always suitable to decompose the field $\mathbb{F}_{2^{2^k}}$ into k extensions of degree 2. As a example, $\mathbb{F}_{2^{128}}$ can be seen as a degree 4 extension of $\mathbb{F}_{2^{32}}$ and $\mathbb{F}_{2^{32}}$ a degree 32 extension of $\mathbb{F}_2$. In that case, multiplying two elements of $\mathbb{F}_{2^{64}}$ is performed on the extension field $\mathbb{F}_{2^{128}}$, which means that $M_{2^7} = M_{2^6}$. In the end, the total complexity would be:

$$\begin{aligned}\sum_{j=1}^{2} 3^j M_{2^7} + \sum_{j=3}^{6} 3^j M_{2^5} + 3^j M_{2^7} &= 12M_{2^7} + 1080M_{2^5} + 2187M_2 \\ &= 135M_{2^7}\end{aligned}$$

This shows that depending on the representation, the computational cost might vary. However, this computation is done once and for all at the beginning of a GCM session. As such a session can involve thousands of field multiplications (the plain text can have up to 2^{39} bits, i.e. 2^{31} 128 bit blocks), this additional cost can be considered as negligible for long sessions.

Remark 2. For large values of n, the computation of GHASH using the characteristic polynomial is expected to be several times faster than traditional methods that use n field multiplications. For example, consider $n = 10,000$ and the FPGA based bit parallel multiplier from [12], which has a delay of $D_M = 6.637$ ns. This multiplier is based on the Karatsuba algorithm and hence $D_M = 3(\log_2(2^7))D_X + D_A$, i.e., ignoring the delay due to the single level of AND gates, we have $D_M/D_X \approx 21$. Thus, the traditional method for computing GHASH will require $10,000 \times (D_X + D_M) \approx 69.6$ μs. On the other hand, the characteristic polynomial based GHASH using the same multiplier will require approximately $10,000 \times (D_X + D_A) + 128 \times (D_M + D_X) + 135 \times D_M \approx 8.1$ μs, resulting in more than 8 fold reduction in the computation time.

5 Conclusions

In this paper we have proposed a new way to improve the performance of the GHASH function of GCM. Our method is based on the use of the characteristic polynomial of the authentication data. It has allowed us to trade most of the field multiplications involved during the authentication tag computation by a series of 128 independent fields additions. This is very attractive for high performance implementations where all such additions can be performed in parallel. This allows us to reduce the delay of each of the first $n - 128$ multiplications over $\mathbb{F}_{2^{128}}$ to that of one XOR and one AND operation. To illustrate the effectiveness of the proposed method, we have considered $n = 10,000$ and the Karatsuba algorithm based bit parallel multiplier on FPGA from [12], which has a delay of $D_M = 6.637$ ns and $D_M/D_X \approx 21$, and we have estimated that compared to the traditional method, the new method can significantly reduce the GHASH computation time, that it to say, eight times.

In this paper, we have also shown the flexibility of our method in terms of parallelization. Using multiple polynomial reduction units allows us to efficiently parallelize the computations and improve the performance of GHASH even further. Finally, we have also proposed a method, specific to $\mathbb{F}_{2^{128}}$, to compute the initial characteristic polynomial efficiently.

References

1. Bajard, J.-C., Imbert, L., Jullien, G.A.: Parallel Montgomery multiplication in GF(2^k) using trinomial residue arithmetic. In: Proc. 17th IEEE Symposium on Computer Arithmetic (ARITH), pp. 164–171 (2005)
2. Bulens, P., Standaert, F.-X., Quisquater, J.-J., Pellegrin, P., Rouvroy, G.: Implementation of the AES-128 on virtex-5 FPGAs. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 16–26. Springer, Heidelberg (2008)
3. Fan, H., Hasan, M.A.: A new approach to subquadratic space complexity parallel multipliers for extended binary fields. IEEE Transactions on Computers 56(2), 224–233 (2007)
4. Good, T., Benaissa, M.: AES on FPGA from the fastest to the smallest. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 427–440. Springer, Heidelberg (2005)
5. Gordon, J.A.: Very simple method to find the minimum polynomial of an arbitrary nonzero element of a finite field. Electronics Letters 12(25), 663–664 (1976)
6. Jarvinen, K.U., Tommiska, M.T., Skyttae, J.O.: A fully pipelined memoryless 17.8 Gbps AES-128 encryptor. In: International symposium on Field programmable gate arrays - FPGA, pp. 207–215. ACM, New York (2003)
7. Lemsitzer, S., Wolkerstorfer, J., Felber, N., Braendli, M.: Multi-gigabit GCM-AES architecture optimized for FPGAs. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 227–238. Springer, Heidelberg (2007)
8. Mastrovito, E.D.: VLSI Architectures for Computation in Galois Fields. PhD thesis, Dept. of Electrical Eng., Link ping Univ., Sweden (1991)
9. McGrew, D.A., Viega, J.: The Galois/Counter Mode of Operation (GCM) (2005)
10. NIST. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC (November 2007)
11. Paar, C.: A new architecture for a parallel finite field multiplier with low complexity based on composite fields. IEEE Transactions on Computers 45(7), 856–861 (1996)
12. Patel, P.: Parallel multiplier designs for the Galois/counter mode of operation. Master's thesis, Electrical and Computer Engineering, University of Waterloo (2008)

13. Satoh, A.: High-speed hardware architectures for authenticated encryption mode gcm. In: IEEE International Symposium on Circuits and Systems - ISCAS, pp. 4831–4834 (2006)
14. Satoh, A.: High-speed parallel hardware architecture for Galois counter mode. In: IEEE International Symposium on Circuits and Systems - ISCAS, pp. 1863–1866 (2007)
15. Satoh, A., Sugawara, T., Aoki, T.: High-speed pipelined hardware architecture for Galois counter mode. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 1863–1866. Springer, Heidelberg (2007)
16. Standaert, F.X., Rouvroy, G., Quisquater, J.-J., Legat, J.-D.: Efficient implementation of Rijndael encryption in reconfigurable hardware: Improvements and design tradeoffs. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 334–350. Springer, Heidelberg (2003)

Principles on the Security of AES against First and Second-Order Differential Power Analysis[⋆]

Jiqiang Lu[1], Jing Pan[1,2], and Jerry den Hartog[1]

[1] Department of Mathematics and Computer Science,
Eindhoven University of Technology,
5600 MB Eindhoven, The Netherlands
lvjiqiang@hotmail.com, {j.pan,j.d.hartog}@tue.nl
[2] Riscure BV, 2628 XJ Delft, The Netherlands
pan@riscure.com

Abstract. The Advanced Encryption Standard (AES) is a 128-bit block cipher that is currently being widely used in smartcards. Differential Power Analysis (DPA) is a powerful technique used to attack a cryptographic implementation in a resource-limited application environment like smartcards. Despite the extensive research on DPA of AES, it seems none has explicitly addressed the fundamental issue: How many rounds of the beginning and end parts of an AES implementation should be protected in order to resist practical DPA attacks, namely first and second-order DPA attacks? Implementation designers may think that it is sufficient to protect the first and last one (or one and a half) rounds of AES, leaving the inner rounds unprotected or protected by simple countermeasures. In this paper, we show that power leakage of some intermediate values from the more inner rounds of AES can be exploited to conduct first and/or second-order DPA attacks by employing techniques such as fixing certain plaintext/ciphertext bytes. We give five general principles on DPA vulnerability of unprotected AES implementations, and then give several general principles on DPA vulnerability of protected AES implementations. These principles specify which positions of AES are vulnerable to first and second-order DPA. To justify the principles, we attack two recently proposed AES implementations that use two kinds of countermeasures to achieve a high resistance against power analysis, and demonstrate that they are even vulnerable to DPA. Finally, we conclude that at least the first two and a half rounds and the last three rounds should be secured for an AES implementation to be resistant against first and second-order DPA in practice.

Keywords: Side channel cryptanalysis, Advanced Encryption Standard, Differential power analysis.

1 Introduction

Nowadays, smartcards are widely used in many real-life security applications as a medium of authenticating the identity of a user and/or executing cryptographic

[⋆] This work was supported by the Dutch Sentinels project PINPASJC TIF.6687.

operations. Power analysis [1] is well known as a practical threat to the security of a cryptographic implementation running in a resource-constrained application environment such as a smartcard. It is based on the fact that the power consumption during an execution of the implementation reveals some information about the data being processed. Primarily, there are two types of power analysis: Simple Power Analysis (SPA) and Differential Power Analysis (DPA). An n-th order DPA attack extracts secret key information by analysing the correlation between the secret key and n points of the power consumption, and it works under the fundamental hypothesis [2]: *There exists a set of n intermediate variables that appear during execution of the algorithm, such that guessing a few key bits (in practice less than 32 bits) allows us to decide whether or not two inputs give the same value for a known function of these n variables.* Though the computing power has been increasing, guessing 32 or more bits is still considered to be unpractical in the context of DPA on smartcards. Up to now, only first and second-order DPA attacks have been experimentally demonstrated to be feasible and efficient in practice, see [3,1,4,5]. Thus, it is a minimum requirement for a cryptographic implementation to be resistant against first and second-order DPA in practice.

The Advanced Encryption Standard (AES) [6] is a 128-bit block cipher with a user key of 128, 192 or 256 bits. It was released by NIST in 2001 as the new-generation data encryption standard for use in the USA, and was adopted as an ISO [7] international standard in 2005. Being used more and more widely in reality, AES is one of the most promising cryptographic algorithms for smartcards, and thus how to design a secure and efficient AES smartcard implementation is of great interest to both industry and academia. A variety of solutions have been presented during the past several years, for example, [10,11,12,8,13,14,9]. One of the most widely used software countermeasures is the Boolean masking method, as followed in [10,15,16,12,17,14]. The main idea of this method is to mask any sensitive intermediate value by XORing it with one or more randomly generated values called masks. Another kind of software countermeasures is hiding, such as randomizing the operations of an algorithm [18].

It has been known that an attacker can exploit the SubBytes operation of the first or last round of AES to conduct a first-order DPA attack. In 2006, Jaffe [19] described a first-order DPA attack exploiting the SubBytes operation of Round 2. Despite the extensive research on the protection of AES against DPA, to our best knowledge, it seems that none has explicitly addressed the fundamental issue: How many rounds of the beginning and end parts of an AES implementation should be protected in order to resist practical DPA attacks, namely first and second-order DPA attacks? To achieve a good tradeoff between security and efficiency, some implementation designers may think that it is enough to protect the first and last one (or one and a half) rounds of AES to thwart first and second-order DPA attacks, leaving the inner rounds unprotected or protected by simple countermeasures, e.g. [8,9]. The reason that the beginning and end parts of AES are particularly protected is that they are more vulnerable to DPA attacks, because an intermediate value from there depends on a relatively small fraction of the key, and an intermediate value from the inner rounds depends on a group of 32 or more key bits, due to the diffusion and confusion properties of the MixColumns operation.

In this paper, we focus on the security of the AES with 128 key bits against DPA. Taking advantage of several simple techniques, such as fixing certain plaintext/ciphertext bytes, we exploit some intermediate values from the inner rounds of AES to conduct first and second-order DPA attacks. We summarise them as five general principles for first and second-order DPA on unprotected AES implementations; some simple cases in the principles have already been known to the public, but more cases are presented for the very first time. We then derive several general principles for first and second-order DPA on protected AES implementations. The principles suggest that when securing an AES implementation against first and second-order DPA one should well protect at least the first two and a half rounds and the last three rounds, i.e. from the beginning until the MixColumns operation of the third round and from the beginning of the eighth round to the end. As examples, we apply the general principles to attack Herbst et al.'s and Tillich et al.'s AES software implementations [8,9], and show that they are not secure against first and/or second-order DPA attacks. An innovative point for Herbst et al.'s and Tillich et al.'s implementations is that they use two kinds of countermeasures, namely randomization (operation shuffling) and masking, to make themselves highly resistant against power analysis.

The remainder of this paper is organised as follows. In the next section we give the notation and briefly review the AES cipher. In Section 3, we present the general principles on first and second-order DPA of unprotected and protected AES implementations, and explain these principles in detail in Section 4. In Section 5, we provide justifications by applying the general principles to Herbst et al.'s and Tillich et al.'s AES implementations in practice. In Section 6 we compare our result with related work. In Section 7 we give several principles on the protection of AES against first and second-order DPA. Section 8 concludes the paper.

2 Preliminaries

In this section we describe some notation and the AES block cipher.

2.1 Notation

In the following descriptions, we assume that a number without a prefix is in decimal (base 10) notation, and a number with prefix $0x$ is in hexadecimal (base 16) notation. The sixteen bytes of a 4×4 byte array are numbered using the conventional row-column fashion, starting with (0,0) (i.e. (0,0), (0,1), $\cdots$, (3,3)). We use the following notation.

$\oplus$	bitwise logical exclusive OR (XOR) of two bit strings of the same length
$*$	polynomial multiplication modulo the polynomial $x^8 + x^4 + x^3 + x + 1$

2.2 The AES Block Cipher

The AES [6] cipher takes as input a 128-bit plaintext block P, represented as a 4×4 byte array, and has a total of 10 rounds. It uses the following four elementary operations to construct the round function.

– The AddRoundKey operation XORs a $4{\times}4$ byte array with a 16-byte subkey.
– The SubBytes operation applies the same $8{\times}8$-bit bijective S-box (denoted below by S) 16 times in parallel to a $4{\times}4$ byte array.
– The ShiftRows operation cyclically shifts the jth row of a $4{\times}4$ byte array to the left by j bytes, $(0 \leq j \leq 3)$.
– The MixColumns operation pre-multiplies a $4{\times}4$ byte array by a fixed $4{\times}4$ byte matrix $MC = (m_{i,j})$; see [6] for the specifications of the matrix MC and its inverse $MC^{-1} = (m'_{i,j})$.

The encryption procedure is, where X is a 16-byte variable, K_0, K_i and K_{10} are 16-byte round keys, and C denotes the ciphertext.

```
1. $X = \text{AddRoundKey}(P, K_0)$.
2. For $i = 1$ to 9:
     $X = \text{SubBytes}(X)$,
     $X = \text{ShiftRows}(X)$,
     $X = \text{MixColumns}(X)$,
     $X = \text{AddRoundKey}(X, K_i)$.
3. $X = \text{SubBytes}(X), X = \text{ShiftRows}(X)$, $C = \text{AddRoundKey}(X, K_{10})$.
```

The ith iteration in Step 2 in the above description is referred to below as Round i, and the transformation in Step 3 is referred to below as the final round (i.e. Round 10). We write $K^i_{j,h}$ for byte (j, h) of K_i, $(0 \leq j, h \leq 3, 0 \leq i \leq 10)$.

3 Principles on First and Second-Order DPA of AES

In this section, we first give five general principles for DPA attacks on unprotected AES implementations, and then derive a few general principles on protected AES implementations. These principles are obtained under the fundamental hypothesis given in Section 1. Recall that a DPA attack is, under the fundamental hypothesis, considered to be unpractical if it requires guessing 32 bits or more. We remind the reader that more inner rounds could be similarly attacked if it was feasible to guess 32 or more bits in some environments.

3.1 Principles for Unprotected AES Implementations

Below are the five general principles for first and second-order DPA on plain AES implementations without countermeasure, which are categorized by the location of the intermediate byte(s) exploited. Details of the principles are given in Section 4.

i. Any intermediate byte before the MixColumns operation of Round 3 can be exploited to conduct a first-order DPA attack. The attack uses a number of plaintexts with 0, 3 or 15 bytes fixed, depending on the location of the exploited byte.
ii. Any intermediate byte after the AddRoundKey operation of Round 7 can be exploited to conduct a first-order DPA attack. The attack uses a number of ciphertexts with 0, 3 or 15 bytes fixed, depending on the location of the exploited byte.
iii. Any two intermediate bytes before the SubBytes operation of Round 3 can be exploited to conduct a second-order DPA attack, if their XOR value is plaintext-dependent[1]. The attack uses a number of plaintexts with 0, 1, 2, 3, 4, 7, 11 or 15 bytes fixed, depending on the location of the two exploited bytes.
iv. Any two intermediate bytes after the SubBytes operation of Round 8 can be exploited to conduct a second-order DPA attack, if their XOR value is ciphertext-dependent. The attack uses a number of ciphertexts with 0, 2, 3, 4, 7 or 15 bytes fixed, depending on the location of the two exploited bytes.
v. Any intermediate byte before the MixColumns operation of Round 2 and any intermediate byte after the MixColumns operation of Round 8 can be exploited to conduct a second-order DPA attack. The attack uses a number of plaintexts with 0, 3 or 4 bytes fixed and their ciphertexts with 0, 4 or 3 bytes fixed, depending on the location of the two exploited bytes.

To apply these DPA attacks in practice one needs to acquire the power traces of plaintexts and/or ciphertexts with the required properties, which depends on specific application environments. These DPA attacks work under four attack scenarios, namely, a ciphertext-only scenario, a known-plaintext scenario, a chosen-plaintext scenario, and an adaptive chosen-ciphertext scenario. The simple cases in the principles, like those exploiting the SubBytes operation of the first or last round, requires a known-plaintext or ciphertext-only scenario. Obtaining plaintexts with a few bytes fixed is simple, which can be easily done under a chosen-plaintext scenario. Obtaining ciphertexts with a few bytes fixed is a little sophisticated. It usually requires an (adaptive) chosen-ciphertext scenario. For instance, an attacker can first choose a number of ciphertexts with certain bytes fixed, and then decrypts them under the attacked key. This usually requires an AES decryption implementation available, and in this case the attacker may choose to attack the decryption implementation using the chosen ciphertexts. Anyway, as demonstrated by Bleichenbacher [20], an adaptive chosen-ciphertext attack is feasible in some real world applications. To obtain plaintexts with a few bytes fixed and their ciphertexts with certain bytes fixed, an attacker can use the following way under a chosen-plaintext scenario: first encrypt a number of plaintexts with certain bytes fixed, and then look for only those ciphertexts whose concerned bytes are identical (plaintexts not meeting this property can be discarded, which causes a high data complexity). Since a

[1] The notion 'plaintext-dependent' means that the XOR value involves some plaintext byte(s). Similar for the following notion 'ciphertext-dependent'.

large amount of computations are needed in order to obtain the useful texts, this attack may only be applicable to cryptographic devices that have a strong computation ability. In the rest of the paper we mainly focus on the possibility of launching an effective DPA attack when the required texts are available.

The principles also consider attacks where 15 bytes of the plaintexts or ciphertexts are fixed and the remaining byte is random. Thus there remain only 256 different plaintexts/ciphertexts available, and a remaining problem is how to obtain sufficient power traces. A simple way to do so is to encrypt/decrypt every available plaintext/ciphertext many times, and then to use the obtained power traces. This is applicable for some devices in practice, particularly considering that a first-order DPA attack can break an AES S-box implementation using as few as 100 or even 30 power traces (see [21, 22, 23]). The technique of fixing a few plaintext and/or ciphertext bits has been used in differential power analysis of some implementations for the DES [24] block cipher [16,26,25] and AES [19], as well as in block cipher cryptanalysis, [27] say.

3.2 Principles for Protected AES Implementations

From Principles (i)–(v), we can easily get the following two principles for a protected AES implementation:

vi. If a byte concerned by Principle (i) or (ii) is unprotected, then it can be exploited to conduct a first-order DPA attack, no matter how well the other parts are protected.
vii. If two bytes concerned by Principle (iii), (iv) or (v) are unprotected or protected only by the same mask, then they can be exploited to conduct a second-order DPA attack, no matter how well the other parts are protected.

Therefore, a protected AES implementation would be vulnerable to a first or second-order DPA attack if it has either of the above weaknesses, no matter how well the other parts are protected. Herbst et al.'s and Tillich et al.'s AES implementations are such examples as to be attacked in Section 5.

4 Principle Details

In this section we explain the principles in Section 3.1 in detail.

4.1 Explaining Principles (i) and (ii)

Simple cases in Principles (i) and (ii) have been well known, such as those exploiting a byte immediately before the SubBytes operation of Round 10. Here we will explain two representative cases: a moderate case in Principle (ii) which exploits a byte immediately before the AddRoundKey operation of Round 8, and the most difficult case in Principle (i) which exploits a byte immediately after the SubBytes operation of Round 3. The reasoning is similar for the other cases of Principles (i) and (ii).

Attacking the AddRoundKey Operation in Round 8. Let v be the exploited byte immediately before the AddRoundKey operation in Round 8. There is no MixColumns operation in the last round of AES, and a byte immediately before the AddRoundKey operation of Round 8 depends on a group of 9 key bytes (one from K_8, four from K_9, and the remaining four from K_{10}) when we express it using relevant ciphertext bytes. Thus, a naive first-order DPA attack exploiting the byte v would need to guess more than 32 secret key bits, which is infeasible under the fundamental hypothesis. Nevertheless, as described subsequently, we find that this limitation can be circumvented by using the technique of fixing a few ciphertext bytes.

The XOR of v and the corresponding byte of K_8, denoted by k_0, is input to an S-box in Round 9; and let u be the output of the S-box. Thus, we have $v = k_0 \oplus S^{-1}(u)$. The following ShiftRows operation does not change the value of u. Then, u is used as a part of the input to the MixColumns operation in Round 9, and we refer to the resulting output column (four bytes) as (u_0, u_1, u_2, u_3). Hence, u can be expressed as follows:

$$u = m'_0 * u_0 \oplus m'_1 * u_1 \oplus m'_2 * u_2 \oplus m'_3 * u_3 \,,$$

where m'_0, m'_1, m'_2, m'_3 are relevant elements from MC^{-1}. Subsequently, (u_0, u_1, u_2, u_3) is used to generate four ciphertext bytes, where four bytes of K_9 and four bytes of K_{10} are involved. We denote the involved four bytes of K_9 as (k_1, k_2, k_3, k_4), the involved four bytes of K_{10} as (k_5, k_6, k_7, k_8), and the resulting four ciphertext bytes as (c_0, c_1, c_2, c_3). Thus, the four bytes u_0, u_1, u_2, u_3 can be expressed as,

$$u_0 = S^{-1}(c_0 \oplus k_5) \oplus k_1, \qquad u_1 = S^{-1}(c_1 \oplus k_6) \oplus k_2,$$
$$u_2 = S^{-1}(c_2 \oplus k_7) \oplus k_3, \qquad u_3 = S^{-1}(c_3 \oplus k_8) \oplus k_4 \,.$$

To simplify our explanations, we write the value of $m'_0 * k_1 \oplus m'_1 * u_1 \oplus m'_2 * u_2 \oplus m'_3 * u_3$ as θ. Thus, we have $u = m'_0 * S^{-1}(c_0 \oplus k_5) \oplus \theta$. Therefore, the exploited byte v can now be represented as

$$v = k_0 \oplus S^{-1}(m'_0 * S^{-1}(c_0 \oplus k_5) \oplus \theta) \,. \tag{1}$$

Observe that θ is related to only three ciphertext bytes, namely (c_1, c_2, c_3). Thus, if we have a number of ciphertexts such that bytes (c_1, c_2, c_3) are fixed to arbitrary values (and the other bytes are random), then the value of θ will be constant for these ciphertexts. In order to use these ciphertexts to conduct a first-order DPA attack exploiting v, we need to guess only the 24 unknown bits for (k_0, k_5, θ) to predict the value of v. This is much less than the limitation of 32 bits in the fundamental hypothesis. As a consequence, the DPA attack can reveal the values of the two key bytes k_0 and k_5, guessing only 24 unknown bits. The whole round key K_8 or K_{10} can be revealed by performing additional 15 similar attacks.

Further, we can conduct a single-bit (first-order) DPA attack using Eq. (1), by guessing only the 16 unknown bits for (k_5, θ). In such an attack, one bit of the intermediate byte v is targeted, and to predict the target bit we first make a hypothesis on the value of the corresponding bit of k_0, and then guess for (k_5, θ). The difference between the average of the power traces for which the target bit is 1 and the average of the power traces for which the target bit is 0 is used to determine whether or not the guess for (k_5, θ) is correct. The difference will be the largest only when the guess for (k_5, θ) is correct, because the bit from k_0 does not affect the magnitude and just changes the sign of the difference. Therefore, the single-bit DPA attack can reveal k_5, guessing only 16 unknown bits (k_5, θ).

Attacking the SubBytes Operation in Round 3. The most difficult case in Principle (i) is to attack a byte immediately after the SubBytes operation in Round 3. Let v denote such a byte. Different from the intermediate byte exploited above, any intermediate byte between the MixColumns operation of Round 2 and the MixColumns operation of Round 8 depends on the whole 128 key bits and the whole 128-bit plaintext or ciphertext. Thus, we need more tricks to deal with this case. v is dependent on all the 16 plaintext bytes, due to the diffusion and confusion properties of the MixColumns and ShiftRows operations of Rounds 1 and 2. Following a reasoning similar to that described above, we learn that the intermediate byte v is an expression of the form:

$$v = S(m_0 * S(m_1 * S(p_0 \oplus k_0) \oplus \theta) \oplus \delta)\,, \tag{2}$$

where m_0, m_1 are relevant elements from MC, p_0 is a plaintext byte, k_0 is the byte of K_0 XORed with p_0, θ is a function of 3 plaintext bytes and 4 key bytes, and δ is a function of 12 plaintext bytes and 13 key bytes. After a simple analysis, we can get the details of θ and δ.

If we choose a number of plaintexts with the fifteen bytes except p_0 fixed, then θ and δ are constant for these plaintexts. Thus, using these plaintexts, we can conduct a first-order DPA attack using Eq. (2), by guessing the 24 unknown bits for (k_0, θ, δ), and obtain the 8 key bits for k_0. Note that unlike in the previous attack, we now have to guess for all the 24 unknown bits even when a single-bit DPA is conducted, for each bit of v depends on all the 24 unknown bits.

4.2 Explaining Principles (iii) and (iv)

Second-order DPA exploits two intermediate values. We first explain a moderate case of Principle (iii), and then explain a most difficult case of Principle (iii). Similar for the remaining cases of Principles (iii) and (iv).

Attacking the SubBytes Operation in Round 2. The two exploited bytes are from the output of the SubBytes operation in Round 2. Let's first investigate how the targeted intermediate bytes are calculated. Let v be the output of one exploited S-box in Round 2, and let u be the input of the S-box, that is, $v = S(u)$. This input u equals the XOR of a byte of K_1, denoted by k_0, and an output byte of the MixColumns operation in Round 1. This output byte of the MixColumns operation is calculated based on a column of 4 input bytes, and let (u_0, u_1, u_2, u_3) denote this column. Then, u can be expressed as:

$$u = k_0 \oplus m_0 * u_0 \oplus m_1 * u_1 \oplus m_2 * u_2 \oplus m_3 * u_3 ,$$

where m_0, m_1, m_2, m_3 are relevant elements from MC. The four bytes u_0, u_1, u_2, u_3 are calculated based on 4 plaintext bytes and 4 bytes of K_0; we denote by (p_0, p_1, p_2, p_3) the 4 plaintext bytes and by (k_1, k_2, k_3, k_4) the 4 bytes of K_0. Hence, we have

$$u_0 = S(p_0 \oplus k_1), \quad u_1 = S(p_1 \oplus k_2), \quad u_2 = S(p_2 \oplus k_3), \quad u_3 = S(p_3 \oplus k_4).$$

Let $\theta = k_0 \oplus m_1 * u_1 \oplus m_2 * u_2 \oplus m_3 * u_3$, then v can be rewritten as $v = S(m_0 * S(p_0 \oplus k_1) \oplus \theta)$.

Now, given the other exploited output byte w of the SubBytes operation in Round 2, we have two situations to consider: (A) v and w are dependent on different sets of four plaintext bytes; and (B) v and w are dependent on the same set of four plaintext bytes. We first consider situation A. Compute the XOR of v and w as:

$$v \oplus w = S(m_0 * S(p_0 \oplus k_1) \oplus \theta) \oplus w . \tag{3}$$

It requires 4 plaintext bytes and 5 key bytes to calculate v or w. Hence, a straightforward second-order DPA attack needs to guess the 10 key bytes in order to predict the value $v \oplus w$, which is impossible in practice under the fundamental hypothesis. Nevertheless, observe that θ and w involve a total of 7 plaintext bytes. If we have a number of plaintexts with the seven bytes involved in θ and w fixed, then w and θ are constant for these plaintexts, and thus we can conduct a second-order DPA attack using Eq. (3), by guessing only the 24 unknown bits for (k_1, θ, w). Further, as in the single-bit first-order DPA attack in Section 4.1, if a single-bit second-order DPA attack is conducted using Eq. (3), we need to guess only the 16 unknown bits for (k_1, θ), and finally reveal the key byte k_1.

Next we consider situation B, where v and w are dependent on the same set of four plaintext bytes. Similarly, w can be expressed as $w = S(m_4 * S(p_0 \oplus k_1) \oplus \theta')$, where m_4 is an element from MC, and θ' is a function of the three plaintext bytes (p_1, p_2, p_3) and four key bytes. Thus, the XOR of v and w is

$$v \oplus w = S(m_0 * S(p_0 \oplus k_1) \oplus \theta) \oplus S(m_4 * S(p_0 \oplus k_1) \oplus \theta') . \tag{4}$$

As a result, if we have a number of plaintexts with the three bytes involved in θ and θ' fixed, then θ and θ' are constant for these plaintexts, and thus we can conduct a second-order DPA attack using Eq. (4) to obtain k_1, by guessing only the 24 unknown bits for (k_1, θ, θ').

Attacking the AddRoundKey Operation in Round 2. The two exploited bytes v and w are from the output of the AddRoundKey operation of Round 2. Obviously, either of v and w is dependent on all the 16 plaintext bytes and 21 key bytes. After a simple analysis, we learn that v and w can be respectively expressed as the following form.

$$\begin{aligned} v &= m_0 * S(m_1 * S(p_0 \oplus k_0) \oplus \theta) \oplus \delta\,, \\ w &= m_2 * S(m_3 * S(p_0 \oplus k_0) \oplus \theta') \oplus \delta'\,, \end{aligned}$$

where m_0, m_1, m_2, m_3 are elements from MC, p_0 is a plaintext byte, k_0 denotes the byte of K_0 XORed with p_0, either of θ and θ' is a function of 3 plaintext bytes and 4 key bytes, and either of δ and δ' is a function of 12 plaintext bytes and 16 key bytes. Consequently, the XOR of v and w is

$$\begin{aligned} &v \oplus w = \\ &\quad m_0 * S(m_1 * S(p_0 \oplus k_0) \oplus \theta) \oplus m_2 * S(m_3 * S(p_0 \oplus k_0) \oplus \theta') \oplus \delta \oplus \delta'\,. \end{aligned} \quad (5)$$

Observe that θ' involves the same set of 3 plaintext bytes as θ, and δ' involves the same set of 12 plaintext bytes as δ. Thus, if we have a number of plaintexts with the 15 plaintext bytes except p_0 fixed, then θ, θ', δ and δ' will be constant for these plaintexts. Thus, by guessing the 24 unknown bits for (k_0, θ, θ') we can conduct a single-bit second-order DPA attack using Eq. (5) to obtain k_0.

Furthermore, $v \oplus w$ involves only a total of 12 plaintext bytes in a few special situations, and thus we need a number of plaintexts with only 11 bytes fixed. For example, v and w are the output bytes $(0, i)$ and $(1, i)$ of the AddRoundKey operation of Round 2, where $0 \leq i \leq 3$. As bytes (0,3) and (1,3) of the MC matrix are identical (i.e. $0x01$), XORing v and w will cancel the last input byte of the i-th column of the MixColumns operation of Round 2, and $v \oplus u$ involves a total of 12 plaintext bytes only. As a result, we only need to fix 11 plaintext bytes to get constant θ, θ', δ and δ'. Other special situations include bytes $(0, i)$ and $(3, i)$, bytes $(1, i)$ and $(2, i)$, and bytes $(2, i)$ and $(3, i)$, which can be easily spotted from the MC matrix. These special situations are because every column of the MC matrix contains two identical elements. Similar special cases do not hold for the MC^{-1} matrix, for the four elements of a column of MC^{-1} are different one another.

4.3 Explaining Principle (v)

We briefly explain a difficult case in Principle (v): one exploited byte v is immediately after the SubBytes operation of Round 2, and the other exploited byte w is immediately before the AddRoundKey operation of Round 8. We know that v is dependent on 4 plaintext bytes and 5 round key bytes, and w is dependent on 4 ciphertext bytes and 9 round key bytes. A similar analysis reveals that $v \oplus w$ can be expressed as the following form:

$$v \oplus w = S(m_0 * S(p_0 \oplus k_0) \oplus \theta) \oplus w\,, \quad (6)$$

where m_0 is an element from MC, p_0 is a plaintext byte, k_0 is the byte of K_0 XORed with p_0, and θ is a function of 3 plaintext bytes and 4 key bytes. If we have a number of plaintexts and ciphertexts such that the 3 plaintext bytes involved in θ and the 4 ciphertext bytes involved in w are fixed, then θ and w are constant for these plaintexts and ciphertexts. Consequently, we can conduct a second-order DPA attack using Eq. (6) to obtain k_0, by guessing the 24 unknown bits for (k_0, θ, w). Similarly, a single-bit second-order DPA attack guessing only the 16 unknown bits for (k_0, θ) is feasible using Eq. (6).

Alternatively, $v \oplus w$ can also be expressed as the following form:

$$v \oplus w = S^{-1}(m'_0 * S^{-1}(c_0 \oplus k_0) \oplus \theta) \oplus \delta\,, \tag{7}$$

where m'_0 is an element from MC^{-1}, c_0 is a ciphertext byte, k_0 denotes the byte of K_{10} XORed with c_0, θ is a function of 3 ciphertext bytes and 7 key bytes, and δ is a function of 4 plaintext bytes and 6 key bytes. Therefore, if we have a number of plaintexts and ciphertexts such that the 4 plaintext bytes involved in δ and the 3 ciphertext bytes involved in θ are fixed, we can conduct a second-order DPA attack using Eq. (7) to obtain k_0 by guessing the 24 unknown bits for (k_0, θ, δ), and conduct a single-bit second-order DPA attack to obtain k_0 by guessing only the 16 unknown bits for (k_0, θ).

5 Experimental Results

In 2006, Herbst, Oswald and Mangard [8] presented an AES software implementation for 8-bit smartcards; see [8] for its specifications. Herbst et al.'s implementation uses randomization and masking for (roughly) the first round and the last one and a half rounds, and uses only masking for the inner rounds. Building on Herbst et al.'s idea, in 2007 Tillich, Herbst and Mangard [9] presented an AES implementation for 32-bit smartcards; see [9] for its specifications. Tillich et al.'s implementation uses both the countermeasures for (roughly) the first one and a half rounds and the last one and a half rounds, and uses no countermeasures for the inner rounds.

In this section, we take Herbst et al.'s and Tillich et al.'s AES implementations as examples, and conduct representative experiments to justify the principles in Section 3. We program the two implementation schemes in the EEPROM of an AVR-based 8-bit micro-controller. The micro-controller is clocked at 3.57 Mhz, and the power signals are sampled at a rate of 200M samples per second. Note that we use correlation to distinguish correct and incorrect key guesses in the experiments, but similar results hold when using the classical "difference of means" way [1].

5.1 Practical Attacks on Herbst et al.'s Implementation

From [8] we observe that the output bytes of the SubBytes operation of Round 2 or 9 are protected only by the same mask M' in Herbst et al.'s implementation, and the output bytes of the AddRoundKey operation of Round 1 or 8 are protected only by the same mask M. Thus, by Principle (vii) we learn that Herbst et al.'s implementation is vulnerable to second-order DPA attacks.

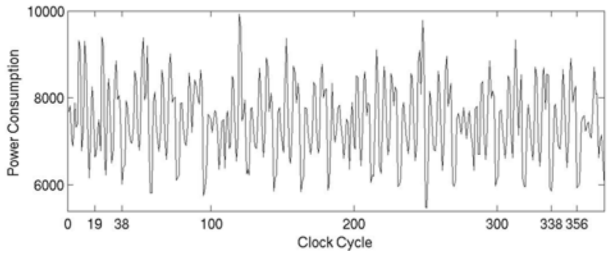


Fig. 1. The SubBytes operation in Round 2

We perform a second-order attack that exploit two output bytes of the SubBytes operation of Round 2. The attack recovers a key byte by guessing only 16 unknown bits. The two exploited bytes are the first byte $X_{0,0}$ and the last byte $X_{3,3}$. Following the descriptions in Section 4.2, we know that their XOR is,

$$X_{0,0} \oplus X_{3,3} = S(m_{0,0} * S(P_{0,0} \oplus K^0_{0,0}) \oplus \theta) \oplus X_{3,3}\,.$$

We choose 16384 plaintexts such that the seven bytes $(1,1)$, $(2,2)$, $(3,3)$, $(1,0)$, $(2,1)$, $(3,2)$ and $(0,3)$ are respectively fixed to 5, 10, 15, 1, 6, 11 and 12, and the other bytes are random, and we use the secret key $K = (K_{i,j})$ with $K_{i,j} = i + j \times 4$. Thus, $\theta = 115$ and $X_{3,3} = 94$ hold for every plaintext.

We then obtain the power traces for encrypting the 16384 plaintexts on the micro-controller. We use the attack strategy given in Chapter 10.3 of [21]. In order to reduce the computational workload of the attack, we compress the power traces by combining (adding up) all the signals during each clock cycle. We next make an educated guess for the time frames when $X_{0,0}$ and $X_{3,3}$ are computed during every execution by optically inspecting the power traces. Fig. 1 plots a power trace segment that corresponds to the SubBytes operation in Round 2, where the 16 sequential S-box substitutions are clearly distinguishable from each other. The first S-box substitution occurs in clock cycles 19–38, and the last S-box substitution occurs in clock cycles 338–356. Subsequently, for every power trace we calculate the absolute-of-difference of each signal in clock cycles 19–38 and each signal in clock cycles 338–356. The resulting traces, which contains 380 signals each, are referred to as the preprocessed power traces. At last, a first-order attack is applied using the preprocessed power traces. In this step, we correlate the power traces to the leftmost bit of the exploited variole $X_{0,0} \oplus X_{3,3}$, where all the possible values for $(K^0_{0,0}, \theta)$ are tested. In our experiment the advantages of a single-bit attack over a multi-bit attack are two-fold: (1) The testing space is reduced by $\frac{1}{3}$; and (2) The ghost peaks caused by incorrect hypotheses of $X_{3,3}$ in case of a multi-bit attack (due to the linearity of XOR) can be avoided.

The results of the second-order attack are plotted in Fig. 2, where the black curve corresponds to the correct guess (i.e., when $(K^0_{0,0}, \theta) = (0, 115)$), and the gray curves correspond to the incorrect guesses. The figure shows that the correct guess leads to the highest peak at point 300. This point corresponds to clock cycles 34 and 352 in the original traces, during which $X_{0,0}$ and $X_{3,3}$ are manipulated on the micro-controller.

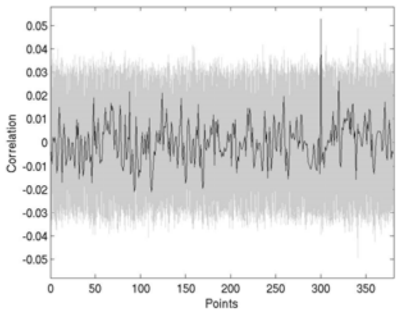


Fig. 2. Results for all the 65536 guesses in a second-order attack

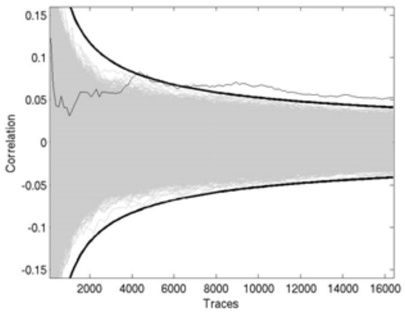


Fig. 3. Results at point 300 with an increasing number of traces

We analyze the number of power traces required by such an attack. We perform 128 attacks using 128, 256, 384, . . . , 16384 power traces, respectively. Different from the above experiment, we only test the interesting point in time, i.e., point 300 in the preprocessed trace. Fig. 3 depicts the evolution of the results over an increasing number of power traces used, where the outer thicker black curves mark the expected region of the incorrect key hypotheses with a confidence of 0.9999. Again, the result for the correct guess is plotted in black, and the rest are plotted in gray. The point where the curve for the correct key hypothesis leaves this region gives an estimation of the number of traces required by a successful attack. From Fig. 3 we get that the attack can almost always succeed with approximately 8000 traces.

5.2 Practical Attacks on Tillich et al.'s Implementation

From [9] we observe that there is no protection between the AddRoundKey operation in Round 2 and the MixColumns operation in Round 3 and between the MixColumns operation in Round 7 and the AddRoundKey operation in Round 8. Thus, by Principle (vi) we can conduct first-order DPA attacks on Tillich et al.'s implementation. Observe that all the output bytes of the AddRoundKey operation in Round 8 are concealed by the same mask M, and thus by Principle (vii), Tillich et al.'s implementation is vulnerable to second-order DPA attacks. The second-order DPA attacks are similar to that described above for Herbst et al.'s implementation.

We perform a first-order attack exploiting an input byte of the AddRoundKey operation in Round 8. The attack reveals a key byte of K_{10} by guessing only 16 unknown bits. The exploited byte is the first input $X_{0,0}$, and by the guideline in Section 4.1 we have,

$$X_{0,0} = K^8_{0,0} \oplus S^{-1}(m'_{0,0} * S^{-1}(C_{0,0} \oplus K^{10}_{0,0}) \oplus \theta).$$

We generate 16384 plaintexts which cause 16384 ciphertexts with bytes $(1, 3)$, $(2, 2)$ and $(3, 1)$ being equal respectively to 13, 10 and 7 when encrypted using

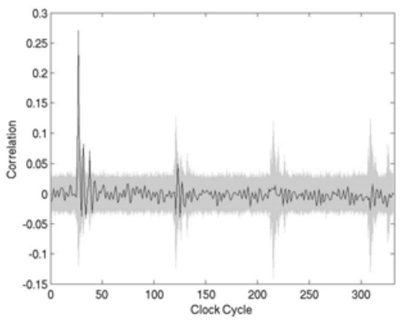


Fig. 4. Results for all 65536 guesses in a first-order attack

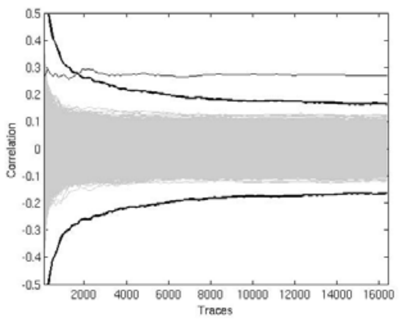


Fig. 5. Results at clock cycle 27 with an increasing number of traces

the secret key $K = (K_{i,j})$ with $K_{i,j} = i + j \times 4$. Hence, $\theta = 124$ for all the ciphertexts, and $K^{10}_{0,0} = 19$.

We collect the power traces for encrypting the plaintexts on the micro-controller. Again, to reduce the computational complexity of the subsequent analysis we perform the same compression of traces and optical inspection as in the attack in Section 5.1. Finally, we successfully find the key byte $K^{10}_{0,0}$ after performing an attack on the leftmost bit of $X_{0,0}$ by guessing for $(K^{10}_{0,0}, \theta)$. The results of this attack are depicted in Fig. 4, where the black curve represents the result for the correct guess of $(K^{10}_{0,0}, \theta)$ and the gray curves represent the results for the incorrect guesses. A distinctive peak happens at clock cycle 27, indicating that $X_{0,0}$ is manipulated at clock cycle 27 during the encryptions. To analyze the number of required power traces, we performe 128 such attacks using 128, 256, 384, . . . , 16384 power traces respectively and calculate the results only for clock cycle 27; the results are shown in Fig. 5, which indicates 2000 traces are adequate for a successful attack.

6 Comparison with Related Work

As mentioned above, this paper concerns solely the security of an AES implementation against the primary type of power analysis — DPA. During the past few years, other types of power analysis have been proposed, including the template attack [28], the side-channel collision analysis [29] and those obtained by combining the existing power analytic techniques [30]. Compared with DPA, these attack techniques require more assumptions on the ability of an attacker and are much harder to conduct in practice.

Following the side-channel collision analysis approach, in 2006 Handschuh and Preneel [31] applied differential cryptanalysis [32] for power attacks on DES, which exploit intermediate values immediately after the first four rounds of DES. Handschuh and Preneel's result differs from ours in several aspects: (I) Their attacks require high probability differentials for reduced rounds of the cipher concerned, and are hard to apply to the first two and a half rounds of AES (i.e. from the beginning until the MixColumns operation of the third round). For DES there are 4-round differentials with large probabilities (e.g. 3.8×10^{-4}); but as the maximum differential probability for the AES S-box is 2^{-6} and the MixColumns operation has a branch number of 5 [33], thus for AES a differential characteristic operating on the first two and a half rounds has a probability of at most 2^{-54}. Park et al. [34] gave the upper bound 1.144×2^{-111} for the probability of a 4-round AES differential. Therefore, these differentials are not useful for power analysis in practice because their probabilities are too small; (II) Handschuh and Preneel's attacks depend on a direct measurement of the Hamming weight of an intermediate value, and like the side-channel collision analysis, are harder to perform than DPA attacks, but our attacks are DPA attacks; and (III) Our result also includes attacks that work for protected implementations where the exploited intermediate state is masked or partially masked.

Motivated by Handschuh and Preneel's work, in 2007 Biryukov and Khovratovich [35] applied two other traditional symmetric-key cryptanalytic methods to obtain new techniques of side-channel cryptanalysis, namely the impossible collision attack and the multiset collision attack, which exploit intermediate values immediately after the first three or four rounds of AES. This is better than our result in terms of the numbers of attacked rounds. However, their attacks require more and stronger assumptions on the ability of an attacker than ours, and have much larger data and computation complexities, e.g., $2^{19} - 2^{32}$ measurements and $2^{27} - 2^{54}$ off-line computations; and another important difference is like that described in the above (III): our result can be applicable in some implementations with the exploited intermediate values being masked.

In 2008, using some sophisticated techniques [9] obtained by combining (high-order) DPA attacks and the windowing technique [3], Tillich and Herbst [36] presented several complicated power analysis attacks on Herbst et al.'s implementation. Their attacks aim to demonstrate how to break the countermeasures; they use the assumption that the attacker has the knowledge of the generating or storing time of a mask (as well as its power consumption), and use the windowing technique to deal with the effect of randomization. By contrast, we aim to demonstrate how far intermediate values can be exploited. Our attacks do not use the assumption, and do not need to deal with the effect of randomization, for they exploit intermediate values beyond the randomization zones; and they require much less plaintexts, and are very simple and easy to conduct in practice.

7 Principles on the Protection of AES against First and Second-Order DPA

From the principles in Section 3, we can easily get the following principles for protecting AES against first and second-order DPA.

(1) Any one or two bytes concerned by Principles (i)–(v) should not be left unprotected.
(2) Any two bytes concerned by Principles (iii)–(v) should not be protected only by the same mask; in other words, if they are protected by the same mask, other countermeasure(s) should be present.

How far should we protect the beginning and end parts of AES in order to resist first and secon-order DPA attacks (under the fundamental hypothesis)? Now the general principles answer this question to some extent: In summary, in order to make an AES implementation secure against first and second-order DPA, one should sufficiently protect at least the first two and a half rounds and the last three rounds of AES, i.e. from the beginning until the MixColumns operation of Round 3 and from the beginning of Round 8 to the end. Note that this result is towards a general application environment, and aims to provide a necessary security level against first and second-order DPA, and sophisticated attacks are not taken into account. In addition, it is based on the techniques we have known currently. In practice, implementation designers can decide the protected rounds according to application environments, for example, if it is not possible for an attacker to obtain plaintexts then the first two and a half rounds might be left unprotected.

Various countermeasures or their combinations can be applied to make an AES implementation secure against first and second-order DPA. If using the randomization and masking techniques as followed in [8,9], one should apply masking to those operations that are vulnerable to first-order DPA, and should apply both countermeasures to those operations that are vulnerable to second-order DPA. If the sophisticated attacks were considered, enhanced protection mechanisms should be adopted.

8 Conclusions

In this paper, we have extensively studied the security of AES against first and second-order DPA. A few general principles have been presented for attacking an AES implementation. We have discovered that some values from the inner rounds of AES can be exploited to conduct a first or second-order DPA attack. As examples, we have demonstrated that Herbst et al.'s and Tillich et al.'s AES implementations are even vulnerable to first and second-order DPA attacks, although they are designed to achieve a high protection against power analysis using two kinds of countermeasures. We have given several principles for protecting an AES implementation against first and second-order DPA attacks. In general, for the time being, the first two and a half rounds and the last three rounds of an AES implementation should be well protected in order to thwart first and second-order DPA in practice.

Acknowledgments. The authors are very grateful to Jaap de Vos for providing the power traces and to Jasper van Woudenberg and the anonymous referees for their comments.

References

1. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
2. Goubin, L., Patarin, J.: DES and differential power analysis — the "duplication" method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)

3. Clavier, C., Coron, J.S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
4. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
5. Oswald, E., Mangard, S., Herbst, C., Tillich, S.: Practical second-order DPA attacks for masked smart card implementations of block ciphers. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 192–207. Springer, Heidelberg (2006)
6. National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), FIPS-197 (2001)
7. International Organization for Standardization (ISO), ISO/IEC 18033-3:2005: Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers (2005)
8. Herbst, C., Oswald, E., Mangard, S.: An AES smart card implementation resistant to power analysis attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
9. Tillich, S., Herbst, C., Mangard, S.: Protecting AES software implementations on 32-bit processors against power analysis. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 141–157. Springer, Heidelberg (2007)
10. 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)
11. Blömer, J., Guajardo, J., Krummel, V.: Provably secure masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004)
12. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
13. Oswald, E., Schramm, K.: An efficient masking scheme for AES software implementations. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 292–305. Springer, Heidelberg (2006)
14. Schramm, K., Paar, C.: Higher order masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006)
15. Akkar, M.-L., Goubin, L.: A generic protection against high-order differential power analysis. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 192–205. Springer, Heidelberg (2003)
16. Akkar, M.L., Bévan, R., Goubin, L.: Two power analysis attacks against one-mask method. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 332–347. Springer, Heidelberg (2004)
17. 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)
18. Daemen, J., Rijmen, V.: Resistance against implementation attacks: a comparative study of the AES proposals. In: Proceedings of The Second Advanced Encryption Standard Candidate Conference, NIST(1999)
19. Jaffe, J.: Introduction to differential power analysis. In: ECRYPT Summer School on Cryptographic Hardware, Side Channel and Fault Analysis (2006)
20. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)

21. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks: revealing the secrets of smart cards. Springer, Heidelberg (2007)
22. Pan, J., den Hartog, J., de Vink, E.: An operation-based metric on CPA resistance. In: Jajodia, S., Samarati, P., Cimato, S. (eds.) SEC 2008. International Federation for Information Processing, vol. 278, pp. 429–443. Springer, Boston (2008)
23. Prouff, E., Ciraud, C., Aumonier, 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)
24. National Institute of Standards and Technology (NIST), Data Encryption Standard (DES), FIPS-46 (1977)
25. Lv, J., Han, Y.: Enhanced DES implementation secure against high-order differential power analysis in smartcards. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 195–206. Springer, Heidelberg (2005)
26. Lv, J.: On two DES implementations secure against differential power analysis in smart-cards. Information and Computation 204(7), 1179–1193 (2006)
27. Gilbert, H., Minier, M.: A collision attack on 7 rounds of Rijndael. In: Proceedings of The Third Advanced Encryption Standard Candidate Conference, NIST (2000)
28. 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)
29. Schramm, K., Wollinger, T.J., 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)
30. Pan, J., den Hartog, J., Lu, J.: You cannot hide behind the mask: Power analysis on a provablely secure S-box implementation. In: Youm, H.Y., Yung, M. (eds.) WISA 2009. LNCS, vol. 5932, pp. 178–192. Springer, Heidelberg (2009)
31. Handschuh, H., Preneel, B.: Blind differential cryptanalysis for enhanced power attacks. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 163–173. Springer, Heidelberg (2007)
32. Biham, E., Shamir, A.: Differential cryptanalysis of the Data Encryption Standard. Springer, Heidelberg (1993)
33. Daemen, J., Rijmen, V.: AES proposal: Rijndael. In: Proceedings of The First Advanced Encryption Standard Candidate Conference, NIST (1998)
34. Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 247–260. Springer, Heidelberg (2003)
35. Biryukov, A., Khovratovich, D.: Two new techniques of side-channel cryptanalysis. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 195–208. Springer, Heidelberg (2007)
36. Tillich, S., Herbst, C.: Attacking state-of-the-art software countermeasures—A case study for AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 228–243. Springer, Heidelberg (2008)

Adaptive Chosen-Message Side-Channel Attacks

Nicolas Veyrat-Charvillon* and François-Xavier Standaert**

Université catholique de Louvain, Crypto Group, Belgium
nicolas.veyrat, fstandae@uclouvain.be

Abstract. Most side-channel attacks that have been published in the open literature assume known- or chosen-message adversarial scenarios. In this paper, we analyze the increase of the attacks' efficiencies that can be obtained by adaptively selecting the messages. For this purpose, we first describe a generic strategy that allows an adversary to take advantage of this capability. We show that it can be applied to any differential power or electromagnetic analysis attack, against unprotected or protected devices and exploiting profiled or non-profiled leakage models. Then, we provide various experiments to quantify these improvements. Finally, we discuss the optimality of our strategy and its implications for the security evaluation of leakage-resilient cryptographic hardware.

1 Introduction

In classical cryptanalysis, the adaptive selection of the inputs to a cryptographic primitive is known to be a powerful ability for the adversaries. For example, blockwise-adaptive chosen-message attacks have been used to show the insecurity of different encryption modes in [9]. Similarly, Bleichenbacher has demonstrated in [3] that chosen-ciphertext attacks can be used to attack the RSA Encryption Standard PKCS #1. And in the symmetric setting, boomerang attacks are an example of how the adaptivity can be exploited to reduce the complexity of certain categories of attacks [24]. Quite surprisingly, and although it is frequently suggested as a possible improvement, very few related works have been performed in the context of side-channel attacks. In [20], Schindler presented a timing attack against RSA with the Chinese remainder theorem that requires some form of adaptivity. And more recently, Köpf and Basin provided a careful model and analysis of such an attack scenario. But the investigations in [12] are carried out in a restricted context of noiseless leakage. This typically applies to timing attacks such as [10], but is of limited interested in the case of power or electromagnetic side-channel attacks, in which noise is a typical issue adversaries have to deal with [1,11,19]. To the best of the authors's knowledge, the generalization of this previous work, from the context of deterministic leakages to the one of probabilistic (or noisy) leakages was left as an open question.

In this paper, we consequently tackle this problem and propose a careful investigation of adaptive chosen-message side-channel attacks. We describe a generic

* Work supported by the Walloon region through the project SCEPTIC.
** Associate researcher of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

strategy that can be applied to improve the efficiency of any distinguisher. As an illustration, we detail its impact for correlation and template attacks [4,5], both from simulations and actual experiments. Our evaluations show significant increases of the side-channel key-recovery success rates. We additionally evaluate the application of adaptive strategies against implementations protected with masking [8] and observe very similar improvements. Eventually, we discuss the optimality of our approach and compare it with the one in [12].

These results imply direct consequences for the good security evaluations of leaking cryptographic devices. They show that when applicable, adaptive strategies require to take larger security margins than for random-message attacks. In other words, the attacks described in this paper indicate how to best exploit the physical information leakage, in the standard DPA setting formalized in [15]. Hence, they can be used to determine the worst-case number of measurements required to performed a successful key-recovery in this context. We note that considering key-recovery attacks in security evaluations can appear as too weak from a theoretical point of view. But as demonstrated in [2], there is a strong relationship between distinguishing attacks and key-recovery attacks in the context of block ciphers. Hence, this situation is not very different than the one in classical (*e.g.* linear, differential) cryptanalysis, in which one considers the best available attacks in order to approximate the security of a cipher. And the adaptive strategies presented in this paper are part of these evaluation tools.

Note also that our results have interesting connections with recent works in the area of leakage resilient cryptography. Indeed, one of the assumptions in, *e.g.* [6,16], is that the leakage function can be adaptively selected by the adversary. But as discussed in [23], this is a quite strong requirement that is rarely observed: it would require that the adversary can change his measurement setup in a constructive manner. In practice, most attacks rather rely on a fixed leakage function and measure this function for different plaintexts. It is the combination of several plaintexts that allows increasing the information leakage in such a way that the key is eventually revealed. Hence, our adaptive selection of the messages provides a more realistic counterpart of the adversarial capabilities.

2 Terminology and Notations

In a side-channel attack, an adversary tries to recover some secret information from a leaking implementation, *e.g.* a software program or an IC computing a cryptographic algorithm. In this paper, we focus on the divide-and-conquer strategies that are most frequently considered in the literature [14] and are formalized as "standard DPA attacks" in [15]. In the context of a block cipher implementation (that will be our running example), one typically targets small pieces of the master key - called subkeys in the following - one by one. The attacks then follow the different steps illustrated in Figure 1.

Namely, we consider a device performing several cryptographic computations $\mathsf{E}_k(x_i)$ on different plaintexts x_i drawn from the text space $\mathcal{X}$, using some fixed key k drawn from the key space $\mathcal{K}$. While computing $\mathsf{E}_k(x_i)$, the device handles some intermediate values that depend on the known input x_i and the unknown key k (defined as sensitive variables in [18]). In practice, the interesting sensitive variables in a DPA attack are the ones that depend on an enumerable subkey s: we denote them as v_i^s, for a plaintext x_i. Any time such a sensitive intermediate value is computed, the device generates some physical leakage, denoted as l_i^k (where the k superscript indicates that the leakage potentially depend on all the key k, including the subkey s). Hence, in order to perform a key-recovery, an adversary first has to select a sensitive value. Given that this variable only depends on a subkey s, he can then predict its result for the plaintexts x_i that generated l_i^k and enumerate every possible subkey candidate $s^* \in \mathcal{S}$. This leads to different hypothetical intermediate variables $v_i^{s^*}$. Afterwards, the adversary exploits a leakage model to map these values from their original space $\mathcal{V}$ towards a modeled leakage space $\mathcal{M}$. As a result, he obtains $|\mathcal{S}|$ different models, denoted

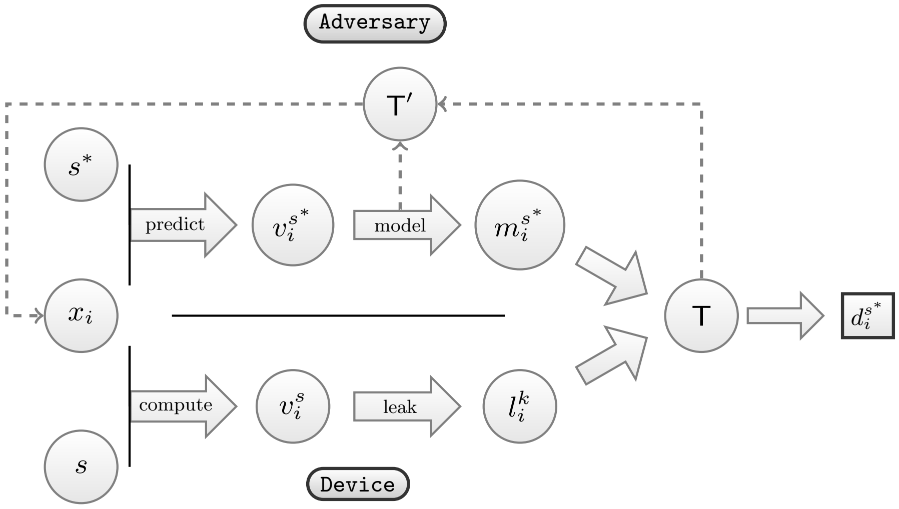


Fig. 1. Schematic description of a side-channel key-recovery attack

as $m_i^{s^*}$, again corresponding to the different subkey candidates. Eventually, he uses a statistical test T to compare the different models $m_i^{s^*}$ with the actual leakages l_i^k. If the attack is successful, the highest value for this test should occur for the correct subkey candidate $s^* = s$. This procedure can be repeated for different subkeys in order to recover the complete key k.

In view of this description, there are several important parameters that determine the efficiency of a DPA. First, the choice of an intermediate computation and leakage model have a significant impact. For example, it is well known that predicting the first round S-boxes' outputs in a block cipher leads to a better discrimination of the subkeys than predicting their inputs [17]. As for the leakage models, it mainly relates to the a-priori knowledge of the adversary about the device he targets. One generally distinguishes profiled and non-profiled attacks. In the first ones (*e.g.* template attacks [5]), the adversary can characterize the leakage probability density functions (pdf for short) prior to the online attack. In the second ones, he exploits simpler models (*e.g.* predicting only certain moments of the leakage pdf, as in correlation attacks [4]) or performs the profiling "on-the-fly" [7]. Second, and closely related, the choice of a statistical test is usually determined by the type of models available to the adversary.

Another parameter that is less frequently considered (and evaluated) in the literature is the selection of the plaintexts. That is, in most experimental settings, one generally considers attacks with random input messages. But as illustrated in Figure 1, a more powerful scenario is to adaptively select the plaintexts, in function of the prior knowledge about the secret subkey and an hypothetical leakage model. In this paper, we consequently investigate the statistical tests T' that can be used in order to best exploit the available leakage.

3 Adaptive Template Attacks

In this section, we present the principles of our adaptive chosen-message strategy. We first describe it in the (profiled) context of template attacks. Then, we discuss how to generalize our solution to non-profiled distinguishers.

3.1 Template Attacks

Template attacks, first published in [5], are usually considered as the most powerful type of side-channel attacks, in an information theoretic sense. They work in two main steps. In a first profiling phase, the adversary builds key-dependent templates, *i.e.* he estimates the leakage pdf for different internal configurations of his target device. Then, in a second (online attack) phase, he uses these templates to perform a maximum-likelihood key-recovery. In this paper, we focus on the (most frequently considered) case of Gaussian templates.

Templates construction. Gaussian template attacks assign a Gaussian distribution to a number of different configurations of the target device. In their most generic form, they perform this assignment exhaustively. For example, if an adversary targets the 8 first bits of an AES master key, he will use one Gaussian for any pair (x_i, s^*), out of the 2^{16} possible ones. In practice, different tricks can be used to reduce this number of templates, in order to increase the efficiency of the profiling, *e.g.* by taking advantage of symmetry properties and stochastic models [21]. In this section, we describe the generic approach for simplicity. Suppose that the adversary is provided with N_p traces to estimate the pdf corresponding to a state (x_i, s). He will then assume that the leakage traces $\{l_i^{k,j}\}_{j=1}^{N_p}$ are drawn from the multivariate normal distribution:

$$\mathcal{N}(l_i^{k,j}|\boldsymbol{\mu}_{x_i}^s, \boldsymbol{\Sigma}_{x_i}^s) = \frac{1}{(2\pi)^{\frac{N}{2}}|\boldsymbol{\Sigma}_{x_i}^s|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(l_i^{k,j} - \boldsymbol{\mu}_{x_i}^s)^\top(\boldsymbol{\Sigma}_{x_i}^s)^{-1}(l_i^{k,j} - \boldsymbol{\mu}_{x_i}^s)\right\},$$

where the mean $\boldsymbol{\mu}^s_{x_i}$ and the covariance matrix $\boldsymbol{\Sigma}^s_{x_i}$ specify completely the noise distribution associated to the leakage trace of each pair (x_i, s). The templates are built by estimating the sets of parameters $\boldsymbol{\mu}^s_{x_i}$ and $\boldsymbol{\Sigma}^s_{x_i}$ for $x_i \in \mathcal{X}$ and $s \in \mathcal{S}$. Maximum likelihood estimators can be used for this purpose: $\hat{\boldsymbol{\mu}}^s_{x_i} = \frac{1}{N_p}\sum_{j=1}^{N_p} l_i^{k,j}$, and $\hat{\boldsymbol{\Sigma}}^s_{x_i} = \frac{1}{N_p}\sum_{j=1}^{N_p}(l_i^{k,j} - \hat{\boldsymbol{\mu}}_{x_i,s})(l_i^{k,j} - \hat{\boldsymbol{\mu}}_{x_i,s})^{\mathsf{T}}$.

Online attack. Assume now that there are $|\mathcal{S}|$ possible subkeys. To determine which one is the most likely to have generated a new trace l_i^k, we compute:

$$\tilde{s} = \underset{s^*}{\operatorname{argmax}}\ \hat{\Pr}[s^*|l_i^k] = \underset{s^*}{\operatorname{argmax}}\ \hat{\Pr}[l_i^k|s^*, x_i] \cdot \hat{\Pr}^{(0)}[s^*],$$

where $\hat{\Pr}[l_i^k|s^*, x_i] = \mathcal{N}(l_i^k|\hat{\boldsymbol{\mu}}^{s^*}_{x_i}, \widehat{\boldsymbol{\Sigma}}^{s^*}_{x_i})$ and $\hat{\Pr}^{(0)}[s^*]$ is the a priori probability of the subkey candidate s^*, that we assume to be uniform in the following (*i.e.* equal to $1/|\mathcal{S}|$, $\forall s^*$). In other words, the classification rule assigns l_i^k to the candidate s^* with the highest a posteriori probability. Since in practice, a single trace is usually not enough to recover the subkey with high confidence, the adversary finally combines several plaintexts and computes $\tilde{s} = \underset{k^*}{\operatorname{argmax}}\ \hat{\Pr}^{(q)}[s^*]$, with:

$$\hat{\Pr}^{(q)}[s^*] = \frac{\prod_{i=1}^{q} \Pr[s^*|l_i^k]}{\sum_{s' \in \mathcal{S}} \prod_{i=1}^{q} \Pr[s'|l_i^k]},$$

and q the number of traces used in the online attack. Note that in the following sections, we will denote as univariate (*resp.* multivariate) the attacks in which the the traces l_i^k contain one (*resp.* several) samples.

3.2 Adaptive Selection of the Plaintexts

Let us now assume that a template attack has been performed with i traces, corresponding to different plaintexts x_1 to x_i, and giving rise to a certain knowledge about the subkey candidates summarized as $\hat{\Pr}^{(i)}[s^*]$. The objective of this paper, illustrated in Figure 1, is to select the next plaintext x_{i+1} in such a way that it will best discriminate the correct subkey. Ideally, this plaintext could be obtained by computing the success rate of the adversary in step $i+1$ or, similarly, by computing the residual entropy of this correct subkey s (*i.e.* one of the metrics in [22]). But while running an attack, the adversary obviously does not know the value of this correct subkey yet. As a consequence, the only applicable strategy is to exploit a criteria that can be estimated "on-the-fly".

Following the previous section, it appears that a natural criteria is to look at the entropy of the subkey candidates rather than the one of the correct subkey. Indeed, in a successful attack, the entropy of these subkey candidates should eventually be null (*i.e.* we should determine only the correct subkey with probability one). For example, at step i, this entropy can be estimated as:

$$\hat{\mathrm{H}}^{(\mathbf{x}_i)}[S^*] = -\sum_{s^*} \hat{\Pr}^{(i)}[s^*] \cdot \log_2 \hat{\Pr}^{(i)}[s^*],$$

where $\mathbf{x}_i = [x_1, x_2, \ldots, x_i]$ is the vector of plaintexts used in the attack. Using this entropy as a criteria for our adaptive chosen-message attacks implies selecting the plaintext x_{i+1} as the one minimizing $\hat{\mathrm{H}}^{(\mathbf{x}_{i+1})}[S^*]$. This can be done as follows. First, let us observe that for every plaintext candidate x^*_{i+1} and subkey candidate s^*, one can define a random variable $\hat{L}^{s^*}_{x^*_{i+1}}$, corresponding to the simulated leakage trace that perfectly follows the leakage model obtained from the templates construction phase (*i.e.* a normal curve with mean vector $\hat{\boldsymbol{\mu}}^{s^*}_{x_{i+1}}$ and covariance matrix $\widehat{\boldsymbol{\Sigma}}^{s^*}_{x^*_{i+1}}$). We can then construct a random variable $\hat{L}^{S^*}_{x^*_{i+1}}$, as a mixture of $\hat{L}^{s^*}_{x^*_{i+1}}$'s, for different plaintext candidates x^*_{i+1}, with probability:

$$\Pr[\hat{L}^{S^*}_{x^*_{i+1}}] = \sum_{s^*} \hat{\Pr}^{(i)}[s^*] \cdot \Pr[\hat{L}^{s^*}_{x^*_{i+1}}]. \tag{1}$$

That is, we have one $\hat{L}^{S^*}_{x^*_{i+1}}$ per plaintext candidate x^*_{i+1}. Exemplary mixtures are represented in Figure 2, for two different plaintexts and in a simple context with only four possible subkeys. The definition of this variable is motivated by the fact that at step i in an attack, the only available knowledge about the subkeys is stored in $\hat{\Pr}^{(i)}[s^*]$. Hence, Equation (1) is the best available estimation of the leakage pdf at this step. For a given mixture and a fixed (simulated) leakage value $\hat{l}^{S^*}_{x^*_{i+1}}$, it is possible to compute the conditional entropy $\hat{\mathrm{H}}^{(\mathbf{x}^*_{i+1})}[S^*|\hat{l}^{S^*}_{x^*_{i+1}}]$, as illustrated in Figure 2 for three exemplary leakage values l_0, l_1 and l_2.

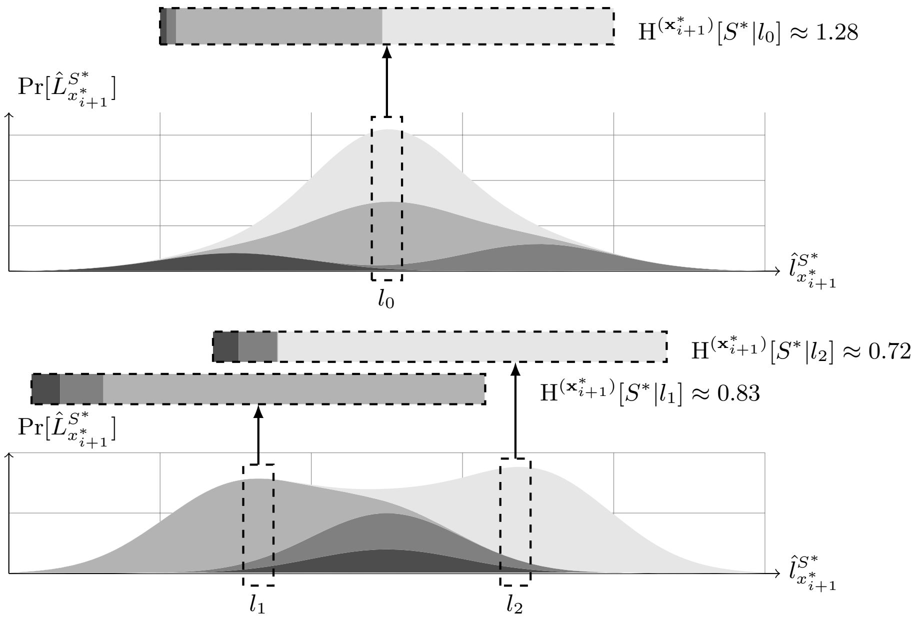


Fig. 2. Adaptive selection of the plaintexts in a simplified context with $|\mathcal{S}| = 4$

Integrating this entropy over the leakages yields the estimations:

$$\hat{\mathrm{H}}^{(\mathbf{x}^*_{i+1})}[S^*] = \int \Pr[\hat{l}^{S^*}_{x^*_{i+1}}] \cdot \hat{\mathrm{H}}^{(\mathbf{x}^*_{i+1})}[S^* | \hat{l}^{S^*}_{x^*_{i+1}}] \, dl^{S^*}_{x^*_{i+1}}.$$

And since we have one such entropy value for every possible choice of x^*_{i+1} in $|\mathcal{X}|$, we finally obtain the following rule to select the plaintexts:

$$\tilde{x}_{i+1} = \underset{x^*_{i+1}}{\operatorname{argmin}} \, \hat{\mathrm{H}}^{\mathbf{x}^*_{i+1}}[S^*]$$

Summarizing, we use the available a-priori subkey information at step i and the leakage model (*i.e.* the templates) to predict how the entropy of the subkey candidates would evolve at step $i+1$, for different plaintext candidates x^*_{i+1}.

3.3 Generalization to Non-profiled Attacks

As detailed in the previous section, an important requirement when applying an adaptive strategy (*e.g.* in the case of template attacks) is the availability of a good leakage model. Therefore, an interesting question is to know if such strategies can still help in the context of non-profiled side-channel attacks, where the model is usually less precise. As an illustration, we discuss this problem for the frequently considered correlation power analysis using Pearson's coefficient.

Correlation attacks, as described in [4], use the following distinguisher:

$$\hat{\rho}(\mathbf{M}_q^{s^*}, \mathbf{L}_q) = \frac{\hat{\mathbf{E}}\Big(\big(l_i - \hat{\mathbf{E}}(\mathbf{L}_q)\big) \cdot \big(m_i^{s^*} - \hat{\mathbf{E}}(\mathbf{M}_q^{s^*})\big)\Big)}{\hat{\sigma}(\mathbf{L}_q) \cdot \hat{\sigma}(\mathbf{M}_q^{s^*})},$$

where $\hat{\mathbf{E}}$ and $\hat{\sigma}$ denote the sample means and standard deviations of a random variable, respectively. In this context, the models $m_i^{s^*}$ are not the complete leakage pdf (as in template attacks) but only their mean values (*i.e.* the first-order moments of the pdf). In general, these mean values are not estimated with profiling, but rather taken from engineering intuition. For example, a usual assumption is to use the so-called Hamming Weight or distance leakage models [14].

Adaptive correlation. When trying to apply the strategy of the previous section to correlation attacks, two main problems arise, that we now detail. First, the subkey probability estimation is not straightforward. Whereas template attacks rate these subkey candidates using their probabilities, correlation attacks return a set of scores, corresponding to the value of Pearson's coefficient. In order to mount an adaptive attack, the adversary consequently needs to use heuristics in order to estimate the subkey distribution $\hat{\Pr}^{(i)}[s^*]$, *e.g.* by:

– using the absolute value of the estimated coefficient $\hat{p}_i^{s^*} = |\hat{\rho}(\mathbf{M}_q^{s^*}, \mathbf{L}_q)|$,
– applying Fisher's transform on this correlation coefficient (in order to get a normal distribution), *i.e.* computing $\hat{p}_i^{s^*} = |\operatorname{arctanh}(\hat{\rho}(\mathbf{M}_q^{s^*}, \mathbf{L}_q))|$,

– computing the p-values associated with each correlation in a hypothesis test. For example, one could estimate the p-value obtained when stating that the subkey candidate in not correlated with the model.

In each case, we then need to normalize the $\hat{p}_i^{s^*}$'s in order to get an estimated probability distribution, as the values we obtain are not actual probabilities, and some wrong subkeys may give a non-zero score (*aka* ghost peaks [4]):

$$\hat{\Pr}^{(i)}[s = s^*] = \frac{\hat{p}_i^{s^*}}{\sum_{s' \in \mathcal{S}} \hat{p}_i^{s'}}$$

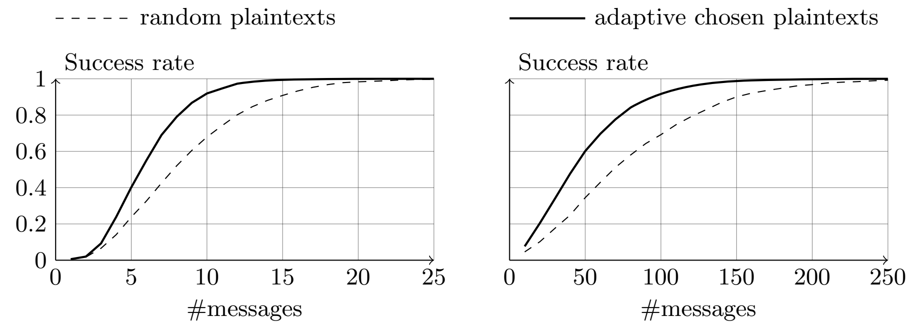
Second, and more critically, the selection procedure of Section 3.2 requires to build a random variable $\hat{L}_{x^*_{i+1}}^{S^*}$ as a mixture of $\hat{L}_{x^*_{i+1}}^{s^*}$, that estimates the leakage distribution given the subkey probabilities at step i in an attack. This requires an estimate of the leakage pdf that is given if the leakage model is probabilistic (as in template attacks), but is not directly available in a correlation attack. Again, a number of heuristics are possible. The simplest one, that we considered in this work, is to combine the (Hamming weight or distance) power models with a Gaussian assumption, *i.e.* to paste a Gaussian curve to the different Hamming weights, of which the variance is estimated "on-the-fly" during the attack.

4 Simulated Experiments

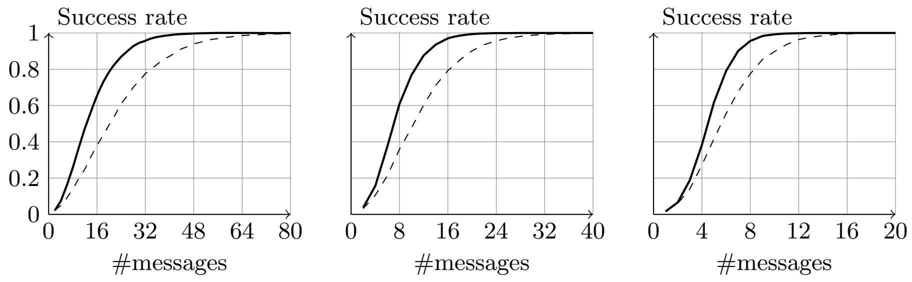
In order to validate our adaptive message selection, we first conducted software simulations. These attacks target the output of a single AES S-box in the first encryption round. Excepted if mentioned otherwise, physical leakages are simulated as the Hamming weight of the S-box outputs, to which is added a normally distributed noise with standard deviation σ_n. The efficiency of an attack is then measured with the success rate, averaged over 1000 independent key recoveries. The results of our experiments are in Figure 3 from which we observe:

1. In all cases, the adaptive strategy leads to increased success rates. It noticeable that the impact of this adaptivity becomes significant as soon as a slight a-priori knowledge is known about the target subkey. Also, and as illustrated in Figure 3.**(a)**, this improvement holds for different noise levels.
2. The same observation also holds for different leakage functions. For example, Figure 3.**(b)** shows the success rates of attacks exploiting three different side-channels of the form: $\mathsf{L}(x) = \sum_i \alpha_i x[i] + n$, where $x[i]$ is the ith bit of the target S-box output and n a Gaussian noise. Interestingly, these examples directly connect with the framework in [22]. They show that as in a non-adaptive context, a more informative leakage function (measured with the conditional entropy $\mathrm{H}[S|\mathbf{L}_1]$) leads to more efficient attacks.
3. Although more computationally intensive (because they require to deal with mixtures of probability distributions, *e.g.* as described in [13]), attacks against masked implementations exhibit similar improvements (see Figure 3.**(c)**).

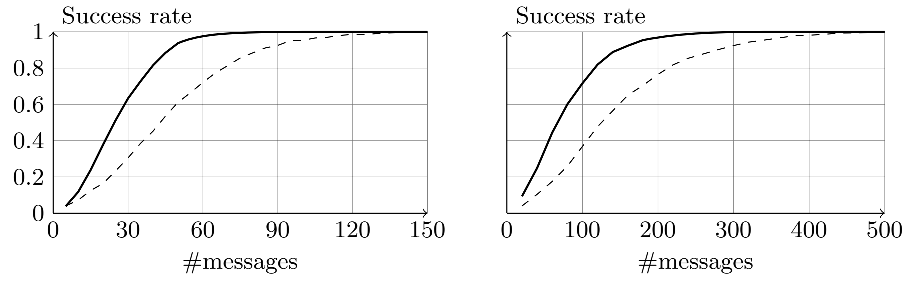
(a) Hamming weight leakage function, $\sigma_n = 1$ (left) and $\sigma_n = 4$ (right).



(b) Different leakage functions with the same noise level $\sigma_n = 1$ and the conditional entropy $\mathrm{H}[S|\mathbf{L}_1] = 7.7$ (left), $\mathrm{H}[S|\mathbf{L}_1] = 7.4$ (middle), $\mathrm{H}[S|\mathbf{L}_1] = 6.9$ (right).



(c) Masked S-box, Hamming weight leakages, $\sigma_n = 0.5$ (left) and $\sigma_n = 1$ (right).



(d) Correlation attacks, Hamming weight leakages, $\sigma_n = 0.5$ (left) and $\sigma_n = 1$ (right).

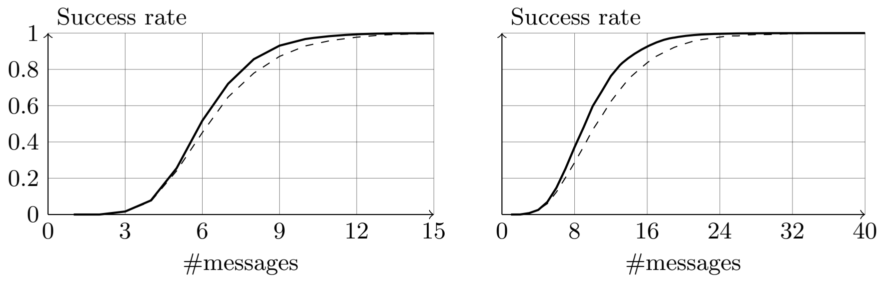


Fig. 3. Success rates of different simulated experiments

4. Eventually, the results of the heuristics proposed to exploit adaptivity in the context of correlation power analysis are given in Figure 3.**(d)**. As expected, the imperfect approximations of the pdf imply smaller improvements.

This last point implies interesting scopes for further research. For example, it would be interesting to apply adaptive strategies to other non-profiled tools such as the MIA [7], in which an estimation of the leakage pdf is computed as part of the attack. In the same line, it could also be possible to exploit stochastic models in order to obtain a leakage model "on-the-fly". In this respect, it is worth recalling that such distinguishers can also be used for profiling a device, without a-priori knowledge of the key (*i.e.* to obtain templates in a flexible way).

5 Experiments Using Actual Measurements

In order to confirm the previous simulations, we additionally performed actual experiments against an implementation of the AES Rijndael in an Atmel Atmega 644p chip. Such actual measurements are interesting because they allow exploiting the leakage of several time samples, contrary to the simulated case where a single point of interest was considered. In other words, actual experiments allow easily evaluating the impact of multivariate templates. In practice, we compared attacks with up to three samples, for adaptive and random message selection. The points of interest were selected as part of the profiling phase, two of them corresponding to the S-box computation, and one to the first key addition. Again, we estimated the success rates over 1000 independent key recoveries, excepted for the trivariate attack which was only launched against 50 different keys. The smaller number of attacks in this case is due to their computational cost, that grows exponentially with the number of dimensions, and makes the exhaustive analysis of Section 3.2 too intensive to be performed.

The results of these experiments are in Figure 4. They show that the adaptive strategy holds for real world implementations. That is, the leakage models built during profiling can be precise enough[1] so that the estimation of the "next-step entropy" $\hat{\mathrm{H}}^{\mathbf{x}^*_{i+1}}[S^*]$ leads to a meaningful selection of the next plaintext x_{i+1}. It is worth noting the large difference between univariate random-message attacks and trivariate chosen-message ones. It illustrates the variability that can be observed between different attack scenarios in physically observable cryptography.

6 Discussion and Concluding Remarks

Is our strategy optimal?. Following the previous sections, a first natural question is to know if the proposed strategy is optimal. For this purpose, it is interesting to relate our work with the one of [12]. The authors estimate the number of queries required for a key-recovery, in the context of deterministic side-channel leakages. For each encryption step, the key candidates are partitioned in

[1] We used a 1000 traces to characterize each template.

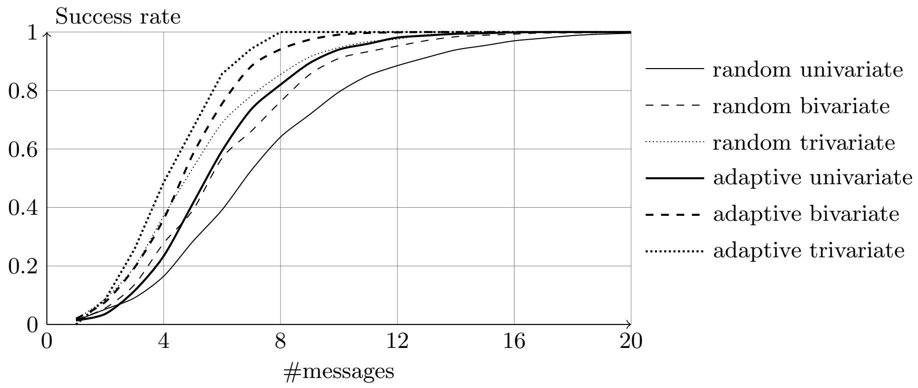


Fig. 4. Success rates of experiments carried out against an AES implementation

r sets, and the side-channel leakage allows the adversary to discriminate one set containing the correct key. The optimal strategy minimizes the number of steps required to reduce the number of key candidates to one. The main limitation is that its computational cost is doubly exponential in the number of attack steps. This is because in this optimal strategy, it is in fact several next plaintexts (x_{i+1}, x_{i+2}, ...) that have to be predicted in order to minimize the entropy of the key candidates. Hence, this strategy is hardly applicable, even for small parameters size. In order to get rid of this limitation, Köpf and Basin propose an alternative greedy heuristic, which predicts only one next plaintext at a time.

The procedure presented in this paper can be seen as the extension of such a greedy strategy, from the deterministic case towards the more general probabilistic case. The main difference is that deterministic leakages allow the adversary to effectively eliminate subkeys, whereas probabilistic leakages only help the adversary to update the subkey candidates' distribution. This extension allows an application of adaptive strategies to a broader class of attacks, including power and electromagnetic leakages, typically. But it comes at a computational cost, since we had to turn deterministic sums into integrals (that are multidimensional in the case of multivariate attacks). Summarizing, our strategy is not optimal. But as indicated in [12], greedy heuristics can provide close to (or even equal to) optimal results in practice. The exact evaluation of the greedy approach with respect to the optimal one and the investigation of alternative solutions to reduce the computational cost of adaptive attacks is a scope for further research.

Implications. Next to optimality, another important question is to determine whether the application of adaptive strategies may have practical impact in certain applications. Looking at the figures in the previous sections indicates that the improvements are not huge, but can be significant. For example, Table 1 shows that the number of measurements required to reach a certain success rate is improved, in particular when combining adaptive attacks with trivariate leakages. But in fact, the consequences of adaptivity are best observed with respect to the global success rates of the attacks. That is, because standard DPA

Table 1. Approximated data complexities for different attacks against an 8-bit subkey

Target success rate	$> 20\%$	$> 40\%$	$> 60\%$	$> 80\%$	$\approx 100\%$
random messages - 1D	5	7	8	11	20
adaptive messages - 1D	4	5	7	8	16
adaptive strategy - 3D	3	4	5	6	8

Table 2. Approximated success rates for different attacks against a 128-bit key

Number of messages	2	3	4	5	6	7	8
random messages - 1D	2^{-69}	2^{-55}	2^{-42}	2^{-29}	2^{-21}	2^{-15}	2^{-10}
adaptive messages - 1D	2^{-64}	2^{-50}	2^{-33}	2^{-20}	2^{-12}	2^{-7}	2^{-5}
adaptive strategy - 3D	2^{-58}	2^{-32}	2^{-16}	2^{-9}	2^{-4}	2^{-2}	-

attacks exploit a divide-and-conquer strategy, the overall success rate against the full AES master key can be estimated by simply raising the success rate against an 8-bit byte to the power 16. This assumes that all key bytes are equally difficult to recover, which is reasonable in most applications, in particular software ones as in Section 5. In the case of adaptive attacks, it also means that the selection of all the plaintext bytes are performed concurrently. Table 2 shows these estimated success rates in function of the number of messages in the attack. It clearly illustrates the strong impact that adaptive strategies may have. For example, one can imagine a re-keying scheme where the secret is updated every four encryptions. Our results suggest that the resulting security level would differ by a factor of 2^9 depending on the use or not of adaptive messages. This factor increases to 2^{26} if multivariate leakages are considered. And in the case of attacks against the AES-256, these factors would additionally be squared.

It is worth mentioning that targeting hardware implementations, in which all the subkeys are manipulated in parallel, would imply additional questions. For example, in a context where a single key byte has to be recovered with high efficiency, one could also take advantage of chosen plaintexts so that the remaining input bits are constant, in order to reduce the algorithmic noise. But an adaptive strategy would still apply to the target key byte. Extending the experiments of this paper towards more devices and countermeasures against side-channel attacks is anyway another interesting direction for further research.

Eventually, and as discussed in [23], the success rates of adaptive attacks can, when applicable, be used as rough (but only available ones) estimations of the bounded leakage[2] that is necessary to prove the security of certain leakage resilient constructions. Our results can also be directly integrated in the evaluation framework of Eurocrypt 2009 [22]: they exhibit a new type of distinguisher that can take advantage of the information leakage in a close to optimal manner. Summarizing, this paper brings an important contribution to the exploitation of side-channel leakages in both theoretical and practical settings.

[2] Given that the plaintext selection is granted to adversaries. As previously said, it is anyway a more reasonable abstraction than the adaptivity of the leakage function.

References

1. Agrawal, D., Archambeault, B., Rao, J., 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. Baignères, T.: Quantitative Security of Block Ciphers: Design and Cryptanalysis Tools. PhD Thesis, EPFL, Lausanne, Switzerland (November 2008)
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, pp. 1–12. Springer, Heidelberg (1998)
4. 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)
5. Chari, S., Rao, J., 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)
6. Dziembowski, S., Pietrzak, K.: Leakage-Resilient Cryptography. In: The proceedings of FOCS 2008, Washington, DC, USA, October 2008, pp. 293–302 (2008)
7. 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. 396–410. Springer, Heidelberg (2008)
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. Joux, A., Martinet, G., Valette, F.: Blockwise-Adaptive Attackers: Revisiting the (In)Security of Some Provably Secure Encryption Models: CBC, GEM, IACBC. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 17–30. Springer, Heidelberg (2002)
10. 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)
11. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–412. Springer, Heidelberg (1999)
12. Köpf, B., Basin, D.A.: An Information-Theoretic Model for Adaptive Side-Channel Attacks. In: The proceedings of the ACM Conference on Computer and Communications Security, Alexandria, Virginia, USA, October 2007, pp. 286–296 (2007)
13. Lemke-Rust, K., Paar, C.: Gaussian Mixture Models for Higher-Order Side Channel Analysis. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 14–27. Springer, Heidelberg (2007)
14. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)
15. Mangard, S., Oswald, E., Standaert, F.-X.: One for All, All for One: Unifying Standard DPA Attacks, Cryptology ePrint Archive, Report 2009/449
16. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2010)
17. Prouff, E.: DPA Attacks and S-Boxes. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 424–441. Springer, Heidelberg (2005)
18. Rivain, M., Dottax, E., Prouff, E.: Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 127–143. Springer, Heidelberg (2008)

19. 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)
20. Schindler, W.: A Timing Attack against RSA with the Chinese Remainder Theorem. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 109–124. Springer, Heidelberg (2000)
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. 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 (2010); extended version available on the Cryptology ePrint Archive, Report 2006/139
23. Standaert, F.-X., Pereira, O., Yu, Y., Quisquater, J.-J., Yung, M., Oswald, E.: Leakage Resilient Cryptography in Practice. Cryptology ePrint Archive, report 2009/341
24. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

Secure Multiplicative Masking of Power Functions

Laurie Genelle[1], Emmanuel Prouff[1], and Michaël Quisquater[2]

[1] Oberthur Technologies
{l.genelle,e.prouff}@oberthur.com
[2] University of Versailles
michael.quisquater@prism.uvsq.fr

Abstract. Side Channel Analysis (SCA) is a powerful key recovery attack that efficiently breaks block ciphers implementations. In software, it is usually counteracted by applying a technique called masking, that combines the key dependent variables with random values. When the block cipher to protect mixes affine functions and power functions, a natural strategy is to additively mask the first category of functions and to multiplicatively mask the second one. Several works that follow this strategy have been proposed in the literature, but all of them have been proved to be flawed or very costly. The main difficulty comes from the multiplicative masking of the zero value in a finite field. In this paper, we propose a scheme to multiplicatively mask power functions in such a way that the security against first-order SCA is maintained. We moreover show how to securely combine additive masking of affine transformations with multiplicative masking of power functions. We then apply our method to protect the AES implementation and we show that our proposal offers good timing/memory performances.

1 Introduction

Originally, cryptographic algorithms were designed to provide resistance against logical attacks. These attacks try to recover the key from ciphertexts related to known or unknown plaintexts. While an algorithm may be considered as perfect from a logical point of view, its implementation may leak information. This very old idea was actually already applied to the *one-time pad* by distinguishing the electrical trace of a zero resulting from the addition of two one's and of two zero's. In the late 90's, the development of the smart card industry instigated the research community to develop attacks against software (and hardware) implementations of algorithms. Those attacks take advantage of the correlation between the manipulated secret key and physical measures such as the running time, the power consumption or the electromagnetic emanation of the algorithm processing. Cryptanalyses based on physical measures are today commonly referred to as *side channel attacks*. For such attacks, the idea consists in targeting internal processed values from which it is often possible to derive information on the key. The family of side channel analyzes can be split into two main categories: the *Simple Power Analysis* (SPA) and the *Differential Power Analysis*

(DPA). SPA consists in directly interpreting power consumption measurements and in identifying the execution sequence. In a DPA, the attacker focuses on the power consumption of a single instruction and performs statistical tests to reveal some correlations between the distribution of the measurement values and the *sensitive data* (*i.e.* depending on a secret value) manipulated by the instruction. Since the publication of the first SPA and DPA, many papers describing either countermeasures or attack improvements have been published (see [2,4,5,13] for example). Among these improvements, *higher-order SCA attacks* are of particular interest. They extend the SPA and DPA by considering a set of several instructions instead of a single one. The number d of instructions targeted by the attack is called *order* of the SCA. Today's implementations are expected to be resistant against first-order SCA (1O-SCA for short), higher order being difficult to mount in practice. That's why only resistance against SCA of order one (including the classical DPA) is considered in this paper. We investigate this problematic for software implementations of block ciphers that mix affine transformations with power functions (*e.g.* the AES).

1.1 Related Work

A way to thwart SCA involves random values (called masks) to de-correlate the leakage signal from the sensitive data that are manipulated. This way of securing the implementation is called *masking*. It is today considered as the most effective one and is therefore privileged by the smart card industry. The masking countermeasures that have been proposed in the literature may be divided into two global strategies.

The first one consists in masking additively internal values whatever the transformation of the block cipher is applied to. For linear operations, dealing with additive masks propagation and correction is straightforward. In contrast, dealing with additive masking for the non-linear steps of the algorithm (*e.g.* the S-box transformations) is more difficult. This issue was addressed in several ways. Table re-computation [5,12] enables to mask any function at the cost of a high memory complexity. It may be a good solution when the device on which is embedded the implementation is not too limited in `RAM` memory. In the context of the S-boxes of the AES, Courtois and Goubin [7] have improved the memory performances of the method by using the compact representation of *homographic functions*. The main other methods concern S-boxes with a simple polynomial representation and they are essentially based on the decomposition of the evaluation of those functions. Blömer *et al.* [3] proposed to evaluate them using a *square-and-multiply* algorithm while propagating additive mask at each step. In the same spirit, several researchers have applied the so-called *tower field methods* [19,10,14,15,16,18] that evaluate functions defined on quadratic extensions of a finite field from the elements of the subfields while propagating the mask from one representation to another.

The second global strategy was initially proposed by Akkar and Giraud [2] and is dedicated to block ciphers that mix additive operations with multiplicative ones. The core of the strategy is to take the best part of each world using additive masking to secure linear operations and multiplicative masking to deal with power functions. Originally, the tricky part of this approach was believed to be the conversion of additive masked values into multiplicative ones and Akkar and Giraud [2] proposed a solution to this problematic. However Golić and Tymen [9] exhibited a flaw in the scheme, observing that the sensitive variable is not masked when it equals zero. Indeed, in this case the multiplicatively masked variable equals zero whatever the value of the mask. As a result and as confirmed afterwards in several papers, it turned out that the main difficulty of the second strategy is to multiplicatively mask the value zero in a finite field.

In order to bypass this difficulty, several solutions were proposed:

- Golić *et al.* [9] proposed to shift the computation of the power function up to a ring embedding the finite field on which the power function is defined. This procedure enables to map the zero element of the finite field to non-zero elements of the ring. This procedure doubles the size of the elements and thus induces both a memory and a computational overhead. Moreover, the scheme does not perfectly protect the data against first-order SCA.
- Trichina *et al.* [20] simplified Akkar and Giraud's scheme and proposed some tricks to correct the zero-value flaw. Unfortunately, the whole scheme was shown to be vulnerable to DPA attacks [1] because some masks are biased.
- Another solution would be to detect the zero value masking processing and to apply a particular treatment in this case. As for instance suggested in [8], this could be done by using conditional branches. However, this solution does not give satisfaction because it is vulnerable to SPA. Trichina and Korkishko [21] proposed a trick based on pre-computed tables in order to avoid the use of conditional branches. However, with such a procedure the processing of the AES S-box is simply wrong when applied to zero or one as noticed by Oswald and Schramm [16]. The latter authors tried to repair the schema but conditional branches were always necessary.

Eventually none of the techniques proposed in the literature to apply multiplicative masking to sensitive data is perfectly secure against first-order SCA. This led people to work in the direction of the first strategy even if the second one is actually more natural for block ciphers such as AES.

1.2 Our Results

The method we describe in this paper circumvents the difficulties encountered so far to combine additive masking with multiplicative one. Our solution may be outlined as follows. We first mask the sensitive variable additively and we compute the propagation of this mask through the affine transformations. We then convert this additive masking into a multiplicative masking, mapping (masked) sensitive data equal to zero into non-zero values. We keep track of this modification if applied and we compute the propagation of the multiplicative mask of this non-zero masked value through the power function. The multiplicative mask is then converted into an additive mask taking into account the potential modification of a sensitive variable at the preceding step. These steps are repeated to eventually obtain both the additively masked ciphertext and the corresponding mask. In the paper we propose two algorithms to convert an additive masking into a multiplicative masking and conversely. Those algorithms result in two methods to secure the implementation of block ciphers. We compared their performances with those of the main techniques proposed in the literature [5, 12, 14, 15, 18, 21]. The timing complexity of our first proposal is ranked second after the table re-computation method [5, 12] and is at least 2.5 times faster than the others [14, 15, 18, 21]. It requires as much memory as the re-computation table method, but it enables to change the masks frequently during the processing with only little overhead. This is a strong advantage since this specificity can be mandated to ensure that no simple higher-order SCA is possible [9]. The second method we propose is an optimization of the first one for devices with little `RAM` and with at least 32 bytes of bit-addressable memory (as for instance the `80C251` architecture microcontrollers). In memory constraints environment, the re-computation method can no longer be used and our solution is the fastest first-order secure alternative to it. We may further conclude that this second proposal achieves the best timing/memory trade-off while having the advantage to enable frequent change of the masks during the computation with only little overhead.

1.3 Paper Organization

The paper is organized as follows. Section 2 deals with the definitions and the concepts we use in the paper. Section 3 and 4 respectively describe the core idea and the algorithms enabling to convert additive masks to multiplicative ones and conversely. Section 5 compares our solution to other implementations in the case of the AES. Section 6 concludes the paper.

2 Theoretical Framework for Security Analysis

We briefly give in what follows some definitions and concepts used to describe our proposal and to analyze its security.

We shall view an implementation of a cryptographic algorithm as the processing of a sequence of *intermediate data*, as defined by Blömer *et al.* [3]. Those data will be associated with *random variables* (r.v. for short) denoted by capital letters, X for instance, while lowercase letters, x for instance, will denote a particular value. The *probability* of the event $\{X \in B\}$ shall be denoted by $\mathrm{P}(X \in B)$. When the event is the singleton $\{x\}$, we will write $\mathrm{P}(X = x)$. If the random variable X has the law of probability $\mathcal{L}(S)$ on the set S, we will write $X \sim \mathcal{L}(S)$. In particular, the uniform law will be denoted by $\mathcal{U}(S)$. We introduce below the concept of *dependency* with respect to $\mathrm{P}(\cdot)$.

Definition 1 (Independency). *Two discrete random variables X_1 and X_2, defined over finite sets S_1 and S_2 respectively, are independent if and only if*

$$\mathrm{P}(X_1 = x_1, X_2 = x_2) = \mathrm{P}(X_1 = x_1) \cdot \mathrm{P}(X_2 = x_2)$$

for any pair $(x_1, x_2) \in S_1 \times S_2$.

Throughout this paper we will often use the following results (see Theorems 3.3.1 and 3.3.2 in [6]).

Proposition 1. *Let $X_1, X_2, \cdots, X_n$ be independent random variables taking respectively their values in the finite sets $S_1, S_2, \cdots, S_n$. If $T_1, T_2, \cdots, T_n$, U and V denote arbitrary sets, then*

1. *$g_1(X_1), g_2(X_2), \cdots, g_n(X_n)$ are independent for any applications $g_i : S_i \to T_i$, $i = 1...n$,*
2. *$f(X_1, \cdots, X_k)$ and $g(X_{k+1}, \cdots, X_n)$ are independent for any applications $f : S_1 \times \cdots \times S_k \to U$ and $g : S_{k+1} \times \cdots \times S_n \to V$.*

We now state a proposition that will allow us to derive the next results. It generalizes analyzes published in [3,15].

Proposition 2. *Consider a finite group $(G, \star)$, a finite set S and an application $g : S \to G$. Let X_1 and X_2 be two independent random variables over G and S, respectively. If X_1 is uniform, then $X_3 = X_1 \star g(X_2)$ is a uniform random variable over G which is independent of X_2.*

Proof. See Appendix A.1. □

In the next proposition, we apply Proposition 2 to the additive group $(\mathrm{GF}(2^n), \oplus)$ and the multiplicative group $(\mathrm{GF}(2^n)^\star, \cdot)$. These results will be useful to prove the security of our scheme in Sect. 4.

Proposition 3. *In what follows, $\mathcal{L}(S)$ denotes a law of probability on S which is not specified.*

1. *If $X_1 \sim \mathcal{U}(\mathrm{GF}(2^n))$ and $X_2 \sim \mathcal{L}(\mathrm{GF}(2^n))$ are independent r.v's and g is an application from $\mathrm{GF}(2^n)$ to $\mathrm{GF}(2^n)$, then*
$$X_3 = X_1 \oplus g(X_2) \sim \mathcal{U}(\mathrm{GF}(2^n)) \text{ and } X_3 \text{ is independent of } X_2\,.$$
2. *If $X_1 \sim \mathcal{U}(\mathrm{GF}(2^n)^\star)$ and $X_2 \sim \mathcal{L}(\mathrm{GF}(2^n))$ are independent r.v's and g is an application from $\mathrm{GF}(2^n)$ to $\mathrm{GF}(2^n)^\star$, then*
$$X_3 = X_1 \cdot g(X_2) \sim \mathcal{U}(\mathrm{GF}(2^n)^\star) \text{ and } X_3 \text{ is independent of } X_2\,.$$
3. *If $X_1 \sim \mathcal{L}(\mathrm{GF}(2^n)^\star)$, $X_2 \sim \mathcal{U}(\mathrm{GF}(2^n))$ and $X_3 \sim \mathcal{L}(\mathrm{GF}(2^n))$ are independent r.v's and g is an application from $\mathrm{GF}(2^n)$ to $\mathrm{GF}(2^n)$, then*
$$X_4 = X_1 \cdot (X_2 \oplus g(X_3)) \sim \mathcal{U}(\mathrm{GF}(2^n)) \text{ and } X_4 \text{ is independent of } X_3\,.$$

Proof. See Appendix A.2. □

In the rest of the paper, an intermediate variable shall be said to be *sensitive* if it is dependent on the secret key. The following definition gives a formal definition of the security against first-order SCA.

Definition 2 (First-Order SCA Security). *A cryptographic algorithm is said to be secure against first-order SCA if every intermediate variable is independent of every sensitive variable.*

In the next sections, we present the core idea of our proposal and we prove that it is first-order SCA resistant.

3 Core Idea of Our Proposal

The core idea of our contribution deals with the first-order SCA resistant implementation of block ciphers mixing affine transformations and power functions in a finite field. In what follows, Op denotes either an affine transformation or a power function, depending on the context.

In what follows, we assume that input/output of the affine operations of the block cipher (*e.g.* the AES individual affine transformations) are always masked additively. Namely, every such operation Op is implemented such that it takes as input the masked data $x \oplus m_{\text{in}} \in \text{GF}(2^8)$, where m_{in} is randomly generated over $\text{GF}(2^8)$ and x is a sensitive value. Since Op is affine, then the corresponding output equals $Op(x) \oplus Op(m_{\text{in}}) \oplus c$, where c is a constant and $Op(m_{\text{in}}) \oplus c$ is the new mask. In this case, it is obvious that neither the input, nor the output, nor any intermediate data during the computation is sensitive.

Let us now consider the case when Op is a power function (not linear) defined over $\text{GF}(2^8)$. To take advantage of the multiplicative structure of Op, we would like to secure its implementation with multiplicative masks. For such a purpose, since Op is likely to operate on the output of additively masked affine transformations, we first need to convert an additively masked value $x \oplus m_{\text{in}}$ into a multiplicatively one in the form $x \cdot b$, where b is a random value in $\text{GF}(2^8)^{\star}$. Secondly, we need to define a scheme such that Op operates securely on $x \cdot b$, and outputs data in the form $Op(x) \cdot b'$. Eventually, since an affine transformation is likely to operate on the multiplicatively masked output of Op, we need to convert it back into an additively masked value in the form $Op(x) \oplus m_{\text{out}}$, where m_{out} is randomly generated over $\text{GF}(2^8)$. The masking conversions and the secure scheme for the power function must be defined by taking into account the so-called *zero-value problem* identified in several papers [7,9,20,21]: the value $x = 0$ cannot be masked multiplicatively.

The core idea of this paper is to convert additive masking into multiplicative masking *via* an algorithm AMToMM in such a way that the sensitive value $x = 0$ is mapped into 1, keeping trace of this transformation. The power function is then applied to this non-zero multiplicatively masked value. The result is eventually converted into an additive masked value *via* an algorithm MMToAM, taking into account the potential mapping of the zero sensitive value in the first step.

For any y, let us denote by δ_y the function defined by $\delta_y(x) = 1$ if $x = y$ and $\delta_y(x) = 0$ otherwise. The most tricky part in this method is to define AMToMM such that it computes $b \cdot (x \oplus \delta_0(x))$ in a secure way from $x \oplus m_{\text{in}}$, m_{in} and b. This indeed implies the SCA-secure computation of $\delta_0(x)$ from $x \oplus m_{\text{in}}$ and m_{in}. It can first be noticed that we have $\delta_0(x) = \delta_{m_{\text{in}}}(x \oplus m_{\text{in}})$. Based on this remark, we propose in the following section a masked implementation of $\delta_{m_{\text{in}}}(x \oplus m_{\text{in}})$ designed such that:

$$\widetilde{\delta}_{m_{\text{in}}}(x \oplus m_{\text{in}}) = \begin{cases} r \oplus 1 & \text{if } x = 0 \\ r & \text{otherwise} \end{cases}$$

where r is a random value in GF(2^8) (Proposal 1 in Sect(s). 4.1 and 4.3) or in GF(2) (Proposal 2 in Sect. 4.4). To implement it securely and efficiently we choose to tabulate the function thanks to a pre-computed table T defined such that $T[i] = r \oplus 1$ if $i = m_{\text{in}}$ and $T[i] = r$ otherwise. Indeed, for such a table T we have $\widetilde{\delta}_{m_{\text{in}}}(x \oplus m_{\text{in}}) = T[x \oplus m_{\text{in}}]$. The whole procedure is summarized in Fig. 1.

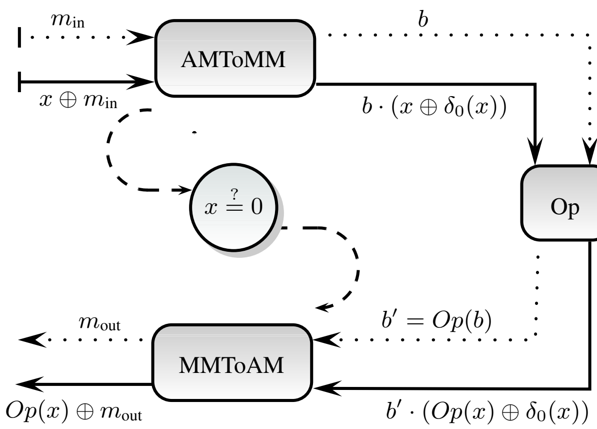


Fig. 1. Multiplicative Masking of a Power Function *Op*

In the next section the three algorithms AMToMM, *Op* and MMToAM are described and their efficiency and security are both discussed.

4 Algorithmic of Our Proposal

In the following description of Alg(s). AMToMM, *Op* and MMToAM we will keep the same notations involved above. Namely, we shall denote by x a sensitive value, by m_{in} an additive mask, by $b \in \text{GF}(2^8)^\star$ a multiplicative mask, by r a random value over GF(2^8) and by T a table of 256 bytes/bits such that $T[x \oplus m_{\text{in}}]$ takes the value $r \oplus 1$ if $x = 0$ and r otherwise. The value r and the table T are regenerated at each execution of the algorithm. Values m_{in} and b can be regenerated several times per algorithm processing. We shall denote $(x)_{m_{\text{in}}} = x \oplus m_{\text{in}}$ and $[x]_b = b \cdot (x \oplus \delta_0(x))$. The output $Op(x) \oplus \delta_0(x)$ shall be denoted by x' and the associated multiplicative mask $Op(b)$ shall be denoted by b'.

4.1 From Additive Masking (AM) to Multiplicative Masking (MM)

To transform every $(x)_{m_{\text{in}}}$ into $[x]_b$ without leaking information about x, we suggest to use the following algorithm which has been decomposed into several elementary operations (left column), each manipulating an intermediate result (right column) computed from $x \oplus m_{\text{in}}$, m_{in}, b and r.

Algorithm 1. SCA-resistant AMToMM

INPUTS: The table T, the random value r, the additively masked value $(x)_{m_{\text{in}}}$, the additive mask m_{in}, the multiplicative mask b
OUTPUT: The multiplicatively masked value $[x]_b$ and the updating of the global variable *mem* with $T[(x)_{m_{\text{in}}}]$

Pseudo-Code	Data
1. $res \leftarrow r$	$res = r$
2. $res \leftarrow res \oplus (x)_{m_{\text{in}}}$	$res = r \oplus x \oplus m_{\text{in}}$
3. $res \leftarrow res \oplus m_{\text{in}}$	$res = r \oplus x$
4. $res \leftarrow b \cdot res$	$res = b \cdot (r \oplus x)$
5. $tmp \leftarrow (x)_{m_{\text{in}}}$	$tmp = x \oplus m_{\text{in}}$
6. $mem \leftarrow T[tmp]$	$mem = r \oplus \delta_0(x)$
7. $tmp \leftarrow b \cdot mem$	$tmp = b \cdot (r \oplus \delta_0(x))$
8. $res \leftarrow res \oplus tmp$	$res = b \cdot (x \oplus \delta_0(x))$

Correctness of Alg. 1. Algorithm 1 processes the following computations. Additions are performed in the order from left to right to insure that the calculation is first-order SCA resistant:

$$b \cdot (r \oplus (x)_{m_{\text{in}}} \oplus m_{\text{in}} \oplus T[(x)_{m_{\text{in}}}]) .$$

Substituting the data according to their definitions, this computation simplifies to:

$$b \cdot (x \oplus \delta_0(x)) = [x]_b .$$

Moreover we can see that Alg. 1 keeps the track of the possible mapping into 1 of $x = 0$ by saving $T[(x)_{m_{\text{in}}}]$ in a global variable denoted by *mem*.

Security Analysis. In the following, we denote by X, M_{in}, B and R the random variables modeling the values x, m_{in}, b and r respectively. From the description of Alg. 1, we obtain in Table 1 the list of all intermediate variables $I_1, \ldots, I_7$ that involve X in their construction. We prove below that each of them is independent of X.

We remind that M_{in} and R are uniformly distributed over GF(2^8), and that B is uniformly distributed over GF$(2^8)^\star$. Moreover M_{in}, B, R and X are by definition mutually independent.

Proposition 4. *Algorithm 1 is first-order SCA resistant.*

Proof. We have $I_1 = (R \oplus M_{\text{in}}) \oplus X$, $I_2 = R \oplus X$, $I_4 = X \oplus M_{\text{in}}$ and $I_5 = R \oplus \delta_0(X)$, where R, M_{in} and X are mutually independent random variables. We moreover have $M_{\text{in}} \sim \mathcal{U}(\text{GF}(2^8))$ and $R \sim \mathcal{U}(\text{GF}(2^8))$. Thus, the random variable $R \oplus M_{\text{in}}$ satisfies $R \oplus M_{\text{in}} \sim \mathcal{U}(\text{GF}(2^8))$ according to the first statement

Table 1. Intermediate variables of Alg. 1

j	I_j
1	$R \oplus X \oplus M_{\text{in}}$
2	$R \oplus X$
3	$B \cdot (R \oplus X)$
4	$X \oplus M_{\text{in}}$
5	$R \oplus \delta_0(X)$
6	$B \cdot (R \oplus \delta_0(X))$
7	$B \cdot (X \oplus \delta_0(X))$

of Proposition 3 and it is independent of X according to Statement 2 of Proposition 1. The three r.v's M_{in}, R and $R \oplus M_{\text{in}}$ being uniform over $\mathcal{U}(\text{GF}(2^8))$ and independent of X, it follows from Statement 1 of Proposition 3 that I_1, I_2, I_4 and I_5 are also independent of X.

We observe now that $I_7 = B \cdot (X \oplus \delta_0(X))$ is the product of a uniform random variable B defined over $\text{GF}(2^8)^*$ with a function of X taking its values in $\text{GF}(2^8)^*$. Noting that B and X are independent, it follows from the second statement of Proposition 3 that I_7 is independent of X.

Eventually, we have $I_3 = B \cdot (R \oplus X)$ and $I_6 = B \cdot (R \oplus \delta_0(X))$, where $B \sim \text{GF}(2^8)^*$, $R \sim \mathcal{U}(\text{GF}(2^8))$ and X are mutually independent random variables. We conclude that I_3 and I_6 are independent of X according to the third statement of Proposition 3.

We have proved that all intermediate variables in Alg. 1 are independent of X. From Definition 2, we can then conclude that Alg. 1 is first-order SCA resistant. $\square$

4.2 Multiplicative Masking of Power Functions

We describe hereafter how to secure the processing of a power function Op with multiplicative masking.

Algorithm 2. SCA-resistant Power Function Op

Inputs: The multiplicative masked value $[x]_b$ and the multiplicative mask b
Output: The masked value $[Op(x)]_{b'}$ and the output mask b'

```
1. output ← $Op([x]_b)$
2. $b'$ ← $Op(b)$
```

Correctness of Alg. 2. We apply the power function Op to the multiplicatively masked value $[x]_b$ and we obtain:

$$Op([x]_b) = Op(b) \cdot Op(x \oplus \delta_0(x)) = b' \cdot x' \, ,$$

where we recall that x' and b' respectively denote $Op(x \oplus \delta_0(x))$ and $Op(b)$. Observing that x' is always non-zero, we deduce that $\delta_0(x') = 0$ and we have:

$$Op([x]_b) = b' \cdot (x' \oplus \delta_0(x')) = [x']_{b'} .$$

Security Analysis. In Alg. 2, the following intermediate variables appear: $I_1 = [x]_b$, $I_2 = Op([x]_b)$ and $I_3 = Op(b)$. Among them only I_1 and I_2 involve X in their definition and we prove below that each of them is independent of X.

Proposition 5. *Algorithm 2 is first-order SCA resistant.*

Proof. Let g denote the function defined from $\mathrm{GF}(2^8)$ into $\mathrm{GF}(2^8)^\star$ by $g(X) = X \oplus \delta_0(X)$. The r.v. I_1 is the product of $g(X)$ with a random variable $B \sim \mathcal{U}(\mathrm{GF}(2^n)^\star)$ which is independent of X. The second statement of Proposition 3 thus implies that I_1 and X are independent. Since I_2 is a function of the r.v. I_1 which is independent of X, it is itself independent of X according to Statement 1 of Proposition 1. We eventually conclude that all the intermediate variables of Alg. 2 are independent of X. Definition 2 thus implies that the latter one is secure against first-order SCA. □

4.3 From Multiplicative to Additive Masking

We recall that we have $x' = Op(x \oplus \delta_0(x))$, that is $x' = Op(x) \oplus \delta_0(x)$ since Op is a power function (and thus $Op(0) = 0$ and $Op(1) = 1$). We assume that m_{out} is a random value generated over $\mathrm{GF}(2^8)$.

The following algorithm describes a method to securely convert every $[x']_{b'}$ into $(Op(x))_{m_{\mathrm{out}}}$.

Algorithm 3. SCA-resistant MMToAM

INPUTS: The multiplicatively masked value $[x']_{b'}$, the multiplicative mask $b' \in \mathrm{GF}(2^8)^\star$, the additive mask $m_{\mathrm{out}} \in \mathrm{GF}(2^8)$, the global variable $mem = T[(x)_{m_{\mathrm{in}}}]$
OUTPUT: The additively masked value $(Op(x))_{m_{\mathrm{out}}}$

Pseudo-Code	Data
1. $res \leftarrow mem$	$res = r \oplus \delta_0(x)$
2. $res \leftarrow res \oplus m_{\mathrm{out}}$	$res = r \oplus \delta_0(x) \oplus m_{\mathrm{out}}$
3. $res \leftarrow res \oplus r$	$res = \delta_0(x) \oplus m_{\mathrm{out}}$
4. $res \leftarrow b' \cdot res$	$res = b' \cdot (\delta_0(x) \oplus m_{\mathrm{out}})$
5. $tmp \leftarrow [x']_{b'}$	$tmp = b' \cdot (Op(x) \oplus \delta_0(x))$
6. $res \leftarrow res \oplus tmp$	$res = b' \cdot (Op(x) \oplus m_{\mathrm{out}})$
7. $res \leftarrow b'^{-1} \cdot res$	$res = Op(x) \oplus m_{\mathrm{out}}$

Correctness of Alg. 3. Algorithm 3 processes the following computations where the operation order is defined from the left to the right between each pair of brackets:

$$b'^{-1} \cdot (b' \cdot (T[(x)_{m_{\mathrm{in}}}] \oplus m_{\mathrm{out}} \oplus r) \oplus [x']_{b'}) .$$

Substituting the variables according to their definitions and observing that $Op(x \oplus \delta_0(x)) = Op(x) \oplus \delta_0(x)$, this computation simplifies to:

$$\begin{aligned}\delta_0(x) \oplus r \oplus m_{\text{out}} \oplus r \oplus x' \oplus \delta_0(x') &= \delta_0(x) \oplus m_{\text{out}} \oplus Op(x \oplus \delta_0(x)) \\ &= m_{\text{out}} \oplus Op(x) \\ &= (Op(x))_{m_{\text{out}}}\end{aligned}$$

Security Analysis. We use the same notations as introduced in Sect. 4.1. Additionally we denote by M_{out} and B' the random variables corresponding respectively to the values m_{out} and b'. Table 2 lists all the intermediate variables $I_1, \ldots, I_7$ of Alg. 3. We prove below that each of them is independent of X.

Table 2. Intermediate variables of Alg. 3

j	I_j
1	$R \oplus \delta_0(X)$
2	$R \oplus \delta_0(X) \oplus M_{\text{out}}$
3	$\delta_0(X) \oplus M_{\text{out}}$
4	$B' \cdot (\delta_0(X) \oplus M_{\text{out}})$
5	$B' \cdot (Op(X) \oplus \delta_0(X))$
6	$B' \cdot (Op(X) \oplus M_{\text{out}})$
7	$Op(X) \oplus M_{\text{out}}$

By definition M_{out} is uniformly distributed over GF(2^8), and B' is uniformly distributed over GF$(2^8)^\star$. Moreover we remind that M_{out}, B' and X are mutually independent.

Proposition 6. *Algorithm 3 is first-order SCA resistant.*

Proof. We have, $I_1 = R \oplus \delta_0(X)$, $I_2 = (M_{\text{out}} \oplus R) \oplus \delta_0(X)$, $I_3 = M_{\text{out}} \oplus \delta_0(X)$ and $I_7 = Op(X) \oplus M_{\text{out}}$ where R, M_{out} and X are mutually independent. After noting that $M_{\text{out}} \sim \mathcal{U}(\text{GF}(2^8))$ and $R \sim \mathcal{U}(\text{GF}(2^8))$, we deduce respectively from the first Statement of Proposition 3 and the second Statement of Proposition 1 that $R \oplus M_{\text{out}}$ satisfies $R \oplus M_{\text{out}} \sim \mathcal{U}(\text{GF}(2^8))$ and is independent of X. We conclude from Statement 1 of Proposition 3 that I_1, I_2, I_3 and I_7 are independent of X.

Let us observe now that $I_5 = B' \cdot (Op(X) \oplus \delta_0(X))$ is the product of a uniform random variable defined over GF$(2^8)^*$ and a function of X with values in GF$(2^8)^\star$. It follows from Statement 2 of Proposition 3 that I_5 is independent of X.

Eventually, we have $I_4 = B' \cdot (\delta_0(X) \oplus M_{\text{out}})$ and $I_6 = B' \cdot (Op(X) \oplus M_{\text{out}})$, where $B' \sim \mathcal{U}(\text{GF}(2^8)^*)$, $M_{\text{out}} \sim \mathcal{U}(\text{GF}(2^8))$ and X are mutually independent. These intermediate variables are independent of X according to Statement 3 of Proposition 3.

All intermediate variables of Alg. 3 are independent of X. We conclude that Alg. 3 is secure against first-order SCA according to Definition 2. $\square$

4.4 Optimization

Here we propose an alternative algorithm to convert an additive masking into a multiplicative masking. Compared with Alg. 1, it involves a RAM-table T of 256 bits (stored in 32 bytes) instead of a RAM-table T of 256 bytes, at the cost of only two additional bitwise additions (Steps 3 and 9 in Alg. 4). This version is therefore of particular interest when the device on which is implemented the countermeasure makes it easy to manipulate bits in memory (*e.g.* has bit-addressable memory).

In what follows, we denote by γ a random bit and we define T by $T[i] = \gamma \oplus 1$ if $i = m_{\text{in}}$ and $T[i] = \gamma$ otherwise.

Algorithm 4. SCA-resistant AMToMM using a bit-table

INPUTS: The table T, the random bit γ, the additively masked value $(x_i)_{m_{\text{in}}}$, the additive mask m_{in}, the multiplicative mask b
OUTPUT: The multiplicatively masked value $[x]_b$ and the updating of the global variable *mem* with $T[(x)_{m_{\text{in}}}]$

Pseudo-Code	Data
1. $res \leftarrow \gamma$	$res = \gamma$
2. $rand \leftarrow$ RNG	$rand$ is random value
3. $res \leftarrow res \oplus rand$	$res = \gamma \oplus rand$
4. $res \leftarrow res \oplus (x)_{m_{\text{in}}}$	$res = \gamma \oplus rand \oplus x \oplus m_{\text{in}}$
5. $res \leftarrow res \oplus m_{\text{in}}$	$res = \gamma \oplus rand \oplus x$
6. $res \leftarrow b \cdot res$	$res = b \cdot (\gamma \oplus rand \oplus x)$
7. $tmp \leftarrow (x)_{m_{\text{in}}}$	$tmp = x \oplus m_{\text{in}}$
8. $mem \leftarrow T[tmp]$	$mem = \gamma \oplus \delta_0(x)$
9. $tmp \leftarrow mem \oplus rand$	$tmp = \gamma \oplus \delta_0(x) \oplus rand$
10. $tmp \leftarrow b \cdot tmp$	$tmp = b \cdot (\gamma \oplus \delta_0(x) \oplus rand)$
11. $res \leftarrow res \oplus tmp$	$res = b \cdot (x \oplus \delta_0(x))$

We have described the optimization for Alg. AMToMM only but the modification must obviously also be done for Alg. MMToAM. Since adapting the optimization above to the latter algorithm is straightforward, we chose to not describe it.

5 Application to the AES

To compare the efficiency of our proposals with that of other methods in the literature, we applied them to protect an implementation of the AES-128 algorithm in encryption mode. We wrote the codes in assembly language for an 8051 based 8-bit architecture without bit-addressable memory. This context was not ideally suitable for our Proposal 2, but as it can be observed in Table 3, its performances are already good for this architecture and actually achieve the best timing/memory trade-off. For this reason, we didn't chose to move to a bit-addressable target[1] for our comparison.

[1] After a rough analysis of the assembly code for our Proposal 2, we think that a 10% loss of performances is due to the fact that no bit-addressable memory is available.

In Table 3, we have listed the timing/memory performances of the different implementations. Memory performances correspond to the number of bytes allocations and cycles numbers correspond to multiple of 10^3. We moreover have dissociated `RAM` consumption inherent to the method from `RAM` consumption related to the implementation choice (in brackets) which can highly vary from a code to another. A column has been added to alert on the fact that some of the listed countermeasures can be easily adapted (without significant timing/memory performances overhead) to involve different masks to secure each S-box calculation. This specificity is referred to as MM (for *Multi-Masking*) in the table and is discussed in more details in [9,18]. We also added a column to point out that some of the listed countermeasures do not achieve first-order SCA resistance as defined in Definition 2. This fact is discussed in further details in the next paragraph.

Table 3. Comparison of AES implementations

	Method	Ref.	Cycles	RAM	ROM	MM	1O-SCA
	Unprotected Implementation						
1.	No Masking	Na.	2	0 (+32)	1150	Na.	No.
	Masking by Addition *vs* First-Order SCA						
2.	Re-computation	[12]	10	256 (+35)	1553	No.	Yes.
3.	Tower Field in GF(2^2)	[14,15]	77	0 (+42)	3195	Yes.	Yes.
4.	Tower Field in GF(2^4)	[16]	29	0 (+36)	3554	Yes.	No.
5.	Masking *on-the-fly*	[18]	82	0 (+39)	2948	Yes.	Yes.
	Masking by Addition-and-Multiplication *vs* First-Order SCA						
6.	Log-ALog Tables	[21]	55	256 (+44)	1900	No.	No.
7.	Proposal 1	Na.	26	256 (+40)	2795	Yes.	Yes.
8.	Proposal 2	Na.	28	32 (+40)	2960	Yes.	Yes.

Outlines of the Methods and Main Differences. The AES implementations listed in Table 3 only differ in their approaches to protect the S-box access. The linear steps and the key-scheduling of the AES have been implemented in the same way: the key-scheduling has not been masked and the internal sensitive data manipulated during the linear steps have been protected by bitwisely adding a random value. We moreover chose to protect all the rounds of the AES processing.

Since the linear steps are protected by additive masking, the AES S-Box denoted by S must be modified to securely deal with such a masking of its inputs/outputs. To answer this issue, the re-computation method computes the table of the function $x \mapsto S[x \oplus m_{\text{in}}] \oplus m_{\text{out}}$ for two pre-defined values of input/output masks and stores it in `RAM`. For the other methods, S is split into its affine part and its non-linear part Op which is the power function $x \in \mathrm{GF}(2^8) \mapsto x^{254} \in \mathrm{GF}(2^8)$. Under this representation, dealing with the mask propagation for S amounts to deal with the mask propagation for Op. The methods of Oswald *et al.* [14,15] and of Prouff *et al.* [18] start by representing

Op over an extension field of GF(2^4) of degree 2 (such a technique is usually called *tower field method*) and they only differ in the ways of securely computing an inversion in GF(2^4). In [15, 14], the inversion is performed by going down to GF(2^2) where this operation is linear. In [18], the inversion is performed for every element of GF(2^4) and the correct result is saved in a special location in memory according to the value of a comparison test. This method is referred to as the *Masking on-the-fly* method in Table 3. All the methods mentioned above achieve perfect security against first-order side channel analysis, which is not the case of those of Trichina-Korkishko [21] and Oswald-Schramm [16] that respectively correspond to the 6th and 4th method presented in Table 3.

In Trichina *et al.*'s method, a *primitive element* of GF(2^8) is computed and every non-zero element of GF(2^8) is expressed as a power of that element. To resolve the zero-value problem, Trichina *et al.* use slightly modified discrete logarithm and exponentiation tables that are pre-computed at the beginning of the processing to evaluate the AES power function. As argued in [16], the method has a faulty behavior when some intermediate values are null and no sound correction of the method has been published until now in the literature. As in [14,15], the method proposed by Oswald-Schramm [16] is based on the tower field method but some operations are processed by accessing look-up tables at addresses that depend on both the mask and the masked data. Since such a variable is dependent on the sensitive variable, the scheme cannot be considered as first-order SCA resistant with respect to Definition 2 and attacks such as those exhibited in [17,22] are possible. Despite the imperfect security of the two methods discussed above, we chose to implement them for comparison since they counteract almost all first-order SCA when no pre-processing is performed (which is for instance the case for the classical DPA [11] and CPA [4]).

Discussion about the Implementations Results. First, since the performances have been measured for a particular implementation on a particular architecture, we draw reader's attention that Table 3 does not aim at arguing that a method is better than another but aims at enlightening the main particularities (timing performances and `ROM`/`RAM` requirements) of each method.

As expected, when 256 bytes of `RAM` memory are available, then the re-computation method achieves the best timing performances. In this context, our first proposal is also promising (ranked second behind the re-computation method). It can even be a valuable alternative to the re-computation method, when the input/output masks of the S-box calculations are required to change during the AES processing. As argued for instance by Golić and Tymen [9] or by Prouff and Rivain [18], such a security requirement can be laid down in order to increase the resistance against simple higher-order SCA. In this case, the re-computation becomes prohibitive whereas the performances of our first proposal stay almost unchanged (the values taken by the input/output mask in Alg. $1-4$ are not assumed to be fixed during the AES processing).

As `RAM` memory is a very sensitive resource in the area of embedded devices, it is often preferable to value memory allocation reduction over timing reduction. Except for our first proposal, all the countermeasures listed in Table 3 are less efficient than the re-computation method but they all require much less RAM allocation. In memory constrained devices, they therefore are preferred to the re-computation method. Among them, our second proposal is the most efficient one. Only the method of Oswald-Schramm has close performances, but at the cost of imperfect security *versus* 1O-SCA. When compared to the methods achieving perfect resistance against first-order SCA, our second proposal is at least 2.5 times faster.

To conclude about the experiment results reported in Table 3, the choice between the implementations that offer perfect resistance against first-order SCA essentially depends on two parameters: the size of the RAM memory available on the device and the necessity to change masks frequently. We sum up our conclusions in Table 4, where we give for different contexts (amount of RAM available and chosen masking methods) the method(s) which is(are) the most efficient one(s).

Table 4. Distribution of the 1O-SCA-resistant methods *vs* the available RAM memory

Method	Cycles ($\times 10^3$)	ROM (in bytes)	Multi-Masking
Device with Large RAM memory (> 256 bytes)			
Re-computation Table	10	1553	No.
Proposal 1	26	2795	Yes.
Device with Medium RAM memory (between 40 and 256 bytes)			
Proposal 2	28	2960	Yes.
Device with Small RAM memory (< 40 bytes)			
Tower Field in GF(2^2)	77	3195	Yes.
Masking *on-the-fly*	82	2948	Yes.

6 Conclusion

In this paper, we have proposed a solution to the zero-value problem in the multiplicative masking. By introducing a scheme that embeds the multiplicative masking in GF(2^8) into a multiplicative masking in GF(2^8)* we obtained a secure method to protect the implementation of power functions. We proposed two methods with different timing/memory trade-offs to implement our scheme and we proved the security of both methods against first-order SCA. We moreover compared the new solutions with the existing ones for the AES. Based on our experiments, we argued that our solutions offer very valuable timing/memory performances. When a large amount of RAM memory can be used, our proposal is ranked second among the implemented methods and offers better resistance to simple higher-order SCA than the first ranked method. In memory constrained devices, our second proposal is ranked first. To the best of our knowledge, it has the best timing/memory overhead and is therefore a valuable alternative to the existing methods.

References

1. Akkar, M.-L., Bévan, R., Goubin, L.: Two Power Analysis Attacks against One-Mask Methods. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 332–347. Springer, Heidelberg (2004)
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. Blömer, J., Merchan, J.G., Krummel, V.: Provably Secure Masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004)
4. 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)
5. Chari, S., Jutla, C., Rao, J., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
6. Chung, K.L.: A Course in Probability Theory. Academic Press, London (2001)
7. Courtois, N., Goubin, L.: An Algebraic Masking Method to Protect AES against Power Attacks. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 199–209. Springer, Heidelberg (2006)
8. Damgard, M., Keller, M.: Secure Multiparty AES. In: Financial Cryptography (to appear, 2010)
9. Golić, J., Tymen, C.: Multiplicative Masking and Power Analysis of AES. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 198–212. Springer, Heidelberg (2003)
10. Gueron, S., Parzanchevsky, O., Zuk, O.: Masked Inversion in GF(2^n) Using Mixed Field Representations and its Efficient Implementation for AES. In: Nedjah, N., Mourelle, L.M. (eds.) Embedded Cryptographic Hardware: Methodologies and Architectures, pp. 213–228. Nova Science Publishers, Bombay (2004)
11. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 388–397. Springer, Heidelberg (1999)
12. Messerges, T.: Securing the AES Finalists against Power Analysis Attacks. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001)
13. Messerges, T.: Using Second-order Power Analysis to Attack DPA Resistant Software. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
14. Oswald, E., Mangard, S., Pramstaller, N.: Secure and Efficient Masking of AES – A Mission Impossible? Cryptology ePrint Archive, Report 2004/134 (2004)
15. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A Side-Channel Analysis Resistant Description of the AES S-box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
16. Oswald, E., Schramm, K.: An Efficient Masking Scheme for AES Software Implementations. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 292–305. Springer, Heidelberg (2006)
17. Prouff, E., McEvoy, R.P.: First-Order Side-Channel Attacks on the Permutation Tables Countermeasure. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 81–96. Springer, Heidelberg (2009)

18. 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)
19. Rudra, A., Bubey, P.K., Jutla, C.S., Kumar, V., Rao, J., 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)
20. Trichina, E., DeSeta, D., Germani, L.: Simplified Adaptive Multiplicative Masking for AES. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 187–197. Springer, Heidelberg (2003)
21. Trichina, E., Korkishko, L.: Secure and Efficient AES Software Implementation for Smart Cards. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 425–439. Springer, Heidelberg (2005)
22. 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 Appendix

A.1 Proof of Proposition 2

Proof. We first observe that for any $x_2 \in S$ and $x_3 \in G$,

$$\mathrm{P}(X_1 \star g(X_2) = x_3, X_2 = x_2) = \mathrm{P}(X_1 \star g(x_2) = x_3, X_2 = x_2)\,.$$

Since X_1 and X_2 are independent and X_1 is uniform over G,

$$\mathrm{P}(X_1 \star\ g(x_2) = x_3, X_2 = x_2) = \mathrm{P}(X_1 \star\ g(x_2) = x_3) \cdot \mathrm{P}(X_2 = x_2) = \frac{1}{\mid G \mid} \cdot \mathrm{P}(X_2 = x_2)\,,$$

for any $x_2 \in S$ and $x_3 \in G$. It follows that for any $x_2 \in S$ and $x_3 \in G$,

$$\mathrm{P}(X_1 \star\ g(X_2) = x_3, X_2 = x_2) = \frac{1}{\mid G \mid} \cdot \mathrm{P}(X_2 = x_2)\,. \tag{1}$$

Let us now prove that $X_3 = X_1 \star\ g(X_2)$ is a uniform random variable over G. According to the law of total probability, for any $x_3 \in G$

$$\mathrm{P}(X_1 \star\ g(X_2) = x_3) = \sum_{x_2 \in S} \mathrm{P}(X_1 \star\ g(X_2) = x_3 \mid X_2 = x_2) \cdot \mathrm{P}(X_2 = x_2)\,.$$

By definition of the conditional probability and (1), for any $x_3 \in G$

$$\mathrm{P}(X_1 \star\ g(X_2) = x_3) = \sum_{x_2 \in S} \frac{1}{\mid G \mid} \cdot \mathrm{P}(X_2 = x_2)\,.$$

Noting that $\sum_{x_2 \in S} \mathrm{P}(X_2 = x_2) = 1$, we conclude that $X_3 = X_1 \star\ g(X_2)$ is a uniform random variable over G. According to (1) and the uniformity of X_3, we have for every $(x_2, x_3) \in S$:

$$\mathrm{P}(X_1 \star\ g(X_2) = x_3, X_2 = x_2) = \mathrm{P}(X_1 \star\ g(X_2) = x_3) \cdot \mathrm{P}(X_2 = x_2)\,.$$

This means that $X_3 = X_1 \star\ g(X_2)$ and X_2 are independent. □

A.2 Proof of Proposition 3

Proof. The first and the second statement follow immediately from Proposition 2 particularized to the groups $G = (\mathrm{GF}(2^n), \oplus)$ and $G = (\mathrm{GF}(2^n)^*, \cdot)$ respectively. Let us prove the third statement. Let us first prove that if we have two independent random variables $A \sim \mathcal{L}(\mathrm{GF}(2^n)^*)$ and $B \sim \mathcal{U}(\mathrm{GF}(2^n))$ then $C = A \cdot B \sim \mathcal{U}(\mathrm{GF}(2^n))$. Let us denote by A^{-1} the inverse of A in $\mathrm{GF}(2^n)^*$. For any $c \in \mathrm{GF}(2^n)$, we have

$$\mathrm{P}(C = c) = \mathrm{P}(A \cdot B = c) = \mathrm{P}(B = c \cdot A^{-1}) = \mathrm{P}(B \oplus c \cdot A^{-1} = 0)\,.$$

Note that $B \oplus c \cdot A^{-1}$ is the sum of uniform random variable over $\mathrm{GF}(2^n)$ and a random variable that may be considered over $\mathrm{GF}(2^n)$. According to the first statement we have $B \oplus c \cdot A^{-1} \sim \mathcal{U}(\mathrm{GF}(2^n))$. It follows that $C \sim \mathcal{U}(\mathrm{GF}(2^n))$.

Consider now the independent random variables $X_1 \sim \mathcal{L}(\mathrm{GF}(2^n)^*)$, $X_2 \sim \mathcal{U}(\mathrm{GF}(2^n))$ and $X_3 \sim \mathcal{L}(\mathrm{GF}(2^n))$. Note that $X_2 \oplus g(X_3)$ is uniform over $\mathrm{GF}(2^n)$ according to the first statement of Proposition 3. Applying the result above to $A = X_1$ and $B = X_2 \oplus g(X_3)$, we deduce that

$$X_4 = X_1 \cdot (X_2 \oplus g(X_3)) \sim \mathcal{U}(\mathrm{GF}(2^n))\,.$$

We are left to prove that $X_4 = X_1 \cdot (X_2 \oplus g(X_3))$ is independent of X_3. For any $x_3, x_4 \in \mathrm{GF}(2^n)$, we have:

$$\mathrm{P}(X_1 \cdot (X_2 \oplus g(X_3)) = x_4, X_3 = x_3) = \mathrm{P}(X_1 \cdot (X_2 \oplus g(x_3)) = x_4, X_3 = x_3)\ \ .$$

Since X_1, X_2 and X_3 are independent random variables, it follows that $X_1 \cdot (X_2 \oplus g(x_3))$ is independent of X_3 for any $x_3 \in \mathrm{GF}(2^n)$ (see for example Theorem 3.3.2 in Chung [6] p.54). Therefore, we have for any $x_3, x_4 \in \mathrm{GF}(2^n)$

$$\mathrm{P}(X_1 \cdot (X_2 \oplus g(X_3)) = x_4, X_3 = x_3) = \mathrm{P}(X_1 \cdot (X_2 \oplus g(x_3)) = x_4) \cdot \mathrm{P}(X_3 = x_3)\,. \tag{2}$$

Now, applying the result above to the random variables $A = X_1$ and $B = X_2$, we have $X_1 \cdot X_2 \sim \mathcal{U}(\mathrm{GF}(2^n))$. Therefore, considering $X_1 \cdot g(x_3)$ as a random variable over $\mathrm{GF}(2^n)$, we have according to the first statement of this proposition that

$$X_1 \cdot (X_2 \oplus g(X_3)) = X_1 \cdot X_2 \oplus X_1 \cdot g(x_3) \sim \mathcal{U}(\mathrm{GF}(2^n))\,.$$

We conclude that

$$\mathrm{P}(X_1 \cdot (X_2 \oplus g(x_3)) = x_4) = \mathrm{P}(X_1 \cdot (X_2 \oplus g(X_3) = x_4)$$

for any $x_3, x_4 \in \mathrm{GF}(2^n)$. Due to (2), we have for any $x_3, x_4 \in \mathrm{GF}(2^n)$,

$$\mathrm{P}(X_1 \cdot (X_2 \oplus g(X_3)) = x_4, X_3 = x_3) = \mathrm{P}(X_1 \cdot (X_2 \oplus g(X_3)) = x_4) \cdot \mathrm{P}(X_3 = x_3)\,.$$

The result follows. $\square$

Batch Groth–Sahai

Olivier Blazy[1], Georg Fuchsbauer[1], Malika Izabachène[2], Amandine Jambert[3,4], Hervé Sibert[5], and Damien Vergnaud[1]

[1] École normale supérieure-CNRS-INRIA, 45 rue d'Ulm, 75320 Paris Cedex 05, France
[2] Université de Versailles, 45 avenue des États-Unis, 78035 Versailles, France
[3] Orange Labs R&D, 42 rue des Coutures, BP6243, 14066 Caen Cedex, France
[4] IMB, Université Bordeaux 1, 351 cours de la Libération, 33405 Talence, France
[5] ST-Ericsson, 9-11 rue Pierre-Felix Delarue, 72100 Le Mans Cedex 9, France

Abstract. In 2008, Groth and Sahai proposed a general methodology for constructing non-interactive zero-knowledge (and witness-indistinguishable) proofs in bilinear groups. While avoiding expensive NP-reductions, these proof systems are still inefficient due to the number of pairing computations required for verification. We apply recent techniques of *batch verification* to the Groth-Sahai proof systems and succeed to improve significantly the complexity of proof verification. We give explicit batch-verification formulas for generic Groth-Sahai equations (whose cost is less than a tenth of the original) as well as for specific popular protocols relying on their methodology (namely Groth's group signatures and the P-signatures by Belenkiy, Chase, Kohlweiss and Lysyanskaya).

Keywords: Pairing-based cryptography, Batch verification, Groth-Sahai proof system.

1 Introduction

In a zero-knowledge proof system, a prover convinces a verifier *via* an interactive protocol that a mathematical statement is true, without revealing anything other than the validity of the assertion. In 1988, Blum, Feldman and Micali [BFM90] showed that the use of a common random string shared between the prover and the verifier permits to design a zero-knowledge proof system for all NP-languages that does not require interaction. These proofs, called non-interactive zero-knowledge (NIZK), turned out to be a particularly useful tool in constructing cryptographic primitives. Unfortunately, their work (as well as subsequent results) does not yield efficient proofs. Until recently, the only way to construct efficient proofs was to rely on the random-oracle model (ROM) [BR93], which has been subject to a series of criticisms starting with [CGH98].

In 2008, Groth and Sahai [GS08] proposed a way to produce efficient and practical NIZK and non-interactive witness-indistinguishable (NIWI) proofs for (algebraic) statements related to groups equipped with a bilinear map. In particular, they give proofs for the simultaneous satisfiability of a set of equations. They proposed three instantiations of their system based on different (mild) computational assumptions: the subgroup decision problem, the symmetric external Diffie-Hellman problem (SXDH) and the decision linear problem (DLIN). Each one of these has already given rise to many applications such as [BW06, BW07, CGS07, Gro07, GL07, BCKL08, BCC$^+$09, FPV09]. Although it is much more efficient than all previous proposals, their proof system still lacks in practicality compared to the ROM, since the verification of a single equation requires the computation of dozens of bilinear-map evaluations by the verifier.

The aim of this paper is to optimize the verification procedure at the expense of slightly weakening the soundness of the proof system.

Prior Work. In the last twenty years, there has been a lot of work in cryptography in which expensive tasks are processed in batch rather than individually to achieve better efficiency. Batch cryptography was first introduced by Fiat [Fia90], who proposed an algorithm to compute several private RSA key operations (with different exponents) through one full exponentiation and several small exponentiations. Batch cryptography is particularly relevant in settings where many exponentiations need to be verified together: many schemes were proposed to achieve batch verification of digital signatures - e.g. [NMVR94] for DSA signatures, and it seems natural to apply such techniques to the verification of Groth-Sahai proofs, which require expensive evaluations of pairings. In 1998, Bellare, Garay and Rabin [BGR98] took the first systematic look at batch verification and described several techniques for conducting batch verification of exponentiations with high confidence. They proposed three generic methods called the *random-subset test*, the *small-exponents test* and the *bucket test*. More recently, Ferrara, Green, Hohenberger and Pedersen [FGHP09] presented a detailed study on how to securely batch-verify a set of pairing-based equations and some applications on existing signatures schemes.

Our Results. The main result of the paper is a significant reduction of the cost of Groth-Sahai proof systems by using batch-verification techniques. In particular, we give efficient explicit verification procedures for the three[1] instantiations proposed in [GS08]. The essence of our approach is a trade-off between soundness and efficiency: if the verification algorithm returns valid, the verifier is assured that all proved statements are indeed valid with overwhelming probability. The best improvements are for the proofs based on SXDH and DLIN, which are the ones with most practical relevance (see Sections 5 and 6). Table 1 summarizes the number of dominant pairing operations required to verify the different algebraic statements in Groth-Sahai terminology (see Section 3 for details).

In [CHP07], Camenisch *et al.* explicitly mentioned as an "exciting" open problem the development of fast batching schemes for various forms of anonymous authentication, such as group signatures and anonymous credentials. This paper is the first to address this issue in the standard security model by considering two schemes based on the Groth-Sahai methodology.

[1] The results for the (least practical) instantiation based on the subgroup decision problem are deferred to the full version of the paper [BFI$^+$10].

Table 1. Number of pairings per proof verification, where n and m stand for the number of different types of variables

	Naive computation	Batch computation
SXDH		
Pairing-product	$5m + 3n + 16$	$m + 2n + 8$
Multi-scalar multiplication in $\mathbb{G}_1$	$8m + 2n + 14$	$\min(2n + 9, 2m + n + 7)$
Multi-scalar multiplication in $\mathbb{G}_2$	$8n + 2m + 14$	$\min(2m + 9, 2n + m + 7)$
Quadratic	$8m + 8n + 12$	$2 \min(m, n) + 8$
DLIN		
Pairing-product	$12n + 27$	$3n + 6$
Multi-scalar multiplication	$9n + 12m + 27$	$3n + 3m + 6$
Quadratic	$18n + 24$	$3n + 6$

The first scheme we consider was proposed by Groth in 2007 [Gro07]. It is a constant-size group-signature scheme whose security can be proved in the standard model, *i.e.* without relying on the random oracle heuristic. For illustrative purposes, we concentrate on the simpler variant of the scheme that provides CPA anonymity only. Even this variant does not achieve satisfactory efficiency—the verification of a signature requires the computation of 68 expensive pairing operations. In Section 7, we propose an improved verification procedure in which the total number of bilinear-map evaluations drops to 11. In addition, if $n \geq 2$ signatures (for the same group) have to be verified at once, we manage to further decrease this number from $11n$ to $4n + 7$.

In Section 8, we study the *P-signature* scheme[2] proposed by Belenkiy, Chase, Kohlweiss and Lysyanskaya [BCKL08]. Since anonymous credentials are an immediate consequence of P-signatures, we thereby apply our techniques to privacy-preserving authentication mechanisms. Belenkiy *et al.* proposed two instantiations of their protocol (based on SXDH and DLIN). They evaluated that the verification of a proof of possession of a signature would involve respectively 68 and 128 pairing evaluations. We show that this can be reduced to 15 and 12, respectively. Moreover, the number of pairing operations required to verify $n \geq 2$ signatures is reduced to $2n + 13$ and $3n + 9$, respectively, by using our techniques.

2 Preliminaries

2.1 Bilinear Groups

Since Groth-Sahai proof systems apply to group-dependent languages, we summarize the basics of bilinear groups and pairing-based assumptions. In the sequel,

[2] A *P-signature* scheme is a digital-signature scheme with an additional non-interactive proof of signature possession.

we consider an algorithm $\mathcal{G}$ that, on input a security parameter λ, outputs a tuple $(N, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order N, g_1 and g_2 generate $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and e is an admissible bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, which means that it is efficiently computable, $e(g_1, g_2)$ generates $\mathbb{G}_T$, and that $e(u^a, v^b) = e(u, v)^{ab}$ for all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_N$.

Definition 1. *Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be a bilinear group with p prime. The* Symmetric eXternal Decision Diffie-Hellman (SXDH) assumption [ACHdM05] *states that the decision Diffie-Hellman assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$,* i.e. *the distributions (u, u^x, u^y, u^z) and $(u, u^x, u^y, u^{x \cdot y})$ are computationally indistinguishable for a random group element $u \in \mathbb{G}_i$ and random scalars $x, y, z \in \mathbb{Z}_p$ (for $i \in \{1, 2\}$).*

Definition 2. *Let $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a bilinear group where p is prime (and $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$). The* decision linear (DLIN) assumption [BBS04] *states that the two distributions (u, v, w, u^a, v^b, w^c) and $(u, v, w, u^a, v^b, w^{a+b})$ are computationally indistinguishable for random group elements $u, v, w \in \mathbb{G}$ and random $a, b, c \in \mathbb{Z}_p$.*

2.2 Notation

We let "$\cdot$" denote the product of two elements either in $\mathbb{Z}_N$, in $\mathbb{G}$ or in $\mathbb{G}_T$. For equal-dimension vectors or matrices A and B of group elements, $A \odot B$ stands for their entry-wise product (*i.e.* their Hadamard product). For a vector or a matrix $A = (a_{i,j})_{i,j}$ of group elements and $x \in \mathbb{Z}$, we let A^x denote the matrix $(a_{i,j}^x)_{i,j}$. Let $\Gamma = (\gamma_{i,j})_{i,j} \in \mathbb{Z}^{m \times n}$ and $\vec{\mathcal{B}} \in \mathbb{G}^n$. Then $\Gamma \vec{\mathcal{B}} := (\prod_{j=1}^n \mathcal{B}_j^{\gamma_{ij}})_{i=1}^m$. We will use $\langle \cdot, \cdot \rangle$ for bilinear products between vectors of either scalars or group elements. Let $\vec{a}, \vec{b} \in \mathbb{Z}_N^n$ and $\vec{\mathcal{A}}, \vec{\mathcal{B}} \in \mathbb{G}^n$. We define

$$\langle \vec{a}, \vec{b} \rangle := \textstyle\sum_{i=1}^n a_i \cdot b_i \qquad \langle \vec{a}, \vec{\mathcal{B}} \rangle := \prod_{i=1}^n \mathcal{B}_i^{a_i} \qquad \langle \vec{\mathcal{A}}, \vec{\mathcal{B}} \rangle := \prod_{i=1}^n e(\mathcal{A}_i, \mathcal{B}_i)$$

We employ Groth and Sahai's notation of a bilinear product (for $k \in \{2, 3\}$):

$$\bullet\colon \mathbb{G}_1^{n \times k} \times \mathbb{G}_2^{n \times k} \to \mathbb{G}_T^{k \times k}$$

defined as $\vec{\mathbf{c}} \bullet \vec{\mathbf{d}} := (\prod_{\ell=1}^n e(c_{\ell,i}, d_{\ell,j}))_{1 \le i,j \le k}$. For the case $\mathbb{G}_1 = \mathbb{G}_2$ and $k = 3$ we define a symmetric variant[3] $\overset{s}{\bullet}\colon \mathbb{G}^{n \times 3} \times \mathbb{G}^{n \times 3} \to \mathbb{G}_T^{3 \times 3}$ by:

$$\vec{\mathbf{c}} \overset{s}{\bullet} \vec{\mathbf{d}} := \left(\prod_{\ell=1}^n e(c_{\ell,i}, d_{\ell,j})^{\frac{1}{2}} e(c_{\ell,j}, d_{\ell,i})^{\frac{1}{2}} \right)_{1 \le i,j \le 3}$$

3 Groth-Sahai Proof Systems

We sketch the results of Groth and Sahai [GS08] on proofs of satisfiability of sets of equations over a bilinear group $(N, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$. Due to the complexity of their methodology, we merely give what is needed for our results and

[3] Note that in their DLIN instantiation, Groth and Sahai use $\tilde{\bullet}$ for the asymmetric map and $\bullet$ for the symmetric variant.

refer to the full version of [GS08] for any additional details. The three types of equations are the following:

A *pairing-product equation* over variables $\vec{\mathcal{X}} \in \mathbb{G}_1^m$ and $\vec{\mathcal{Y}} \in \mathbb{G}_2^n$ is of the form

$$\langle \vec{\mathcal{A}}, \vec{\mathcal{Y}} \rangle \cdot \langle \vec{\mathcal{X}}, \vec{\mathcal{B}} \rangle \cdot \langle \vec{\mathcal{X}}, \Gamma \vec{\mathcal{Y}} \rangle = t_T \ , \tag{1}$$

defined by constants $\vec{\mathcal{A}} \in \mathbb{G}_1^n$, $\vec{\mathcal{B}} \in \mathbb{G}_2^m$, $\Gamma \in \mathbb{Z}_N^{m \times n}$ and $t_T \in \mathbb{G}_T$.

A *multi-scalar multiplication equation* over variables $\vec{y} \in \mathbb{Z}_N^n$ and $\vec{\mathcal{X}} \in \mathbb{G}_1^m$ is of the form

$$\langle \vec{y}, \vec{\mathcal{A}} \rangle \cdot \langle \vec{b}, \vec{\mathcal{X}} \rangle \cdot \langle \vec{y}, \Gamma \vec{\mathcal{X}} \rangle = T \ , \tag{2}$$

defined by the constants $\vec{\mathcal{A}} \in \mathbb{G}_1^n$, $\vec{b} \in \mathbb{Z}_N^m$, $\Gamma \in \mathbb{Z}_N^{m \times n}$ and $T \in \mathbb{G}_1$.
A multi-scalar multiplication equation in group $\mathbb{G}_2$ is defined analogously.

A *quadratic equation in* $\mathbb{Z}_N$ over variables $\vec{x} \in \mathbb{Z}_N^m$ and $\vec{y} \in \mathbb{Z}_N^n$ is of the form

$$\langle \vec{a}, \vec{y} \rangle + \langle \vec{x}, \vec{b} \rangle + \langle \vec{x}, \Gamma \vec{y} \rangle = t \ , \tag{3}$$

defined by the constants $\vec{a} \in \mathbb{Z}_N^n$, $\vec{b} \in \mathbb{Z}_N^m$, $\Gamma \in \mathbb{Z}_N^{m \times n}$ and $t \in \mathbb{Z}_N$.

The common reference string for the proof system is a key to make commitments to the variables of the different types. A proof of satisfiability is constructed by first committing to the variables of the respective equation and then constructing a "proof" for each equation. The latter asserts that the committed values indeed satisfy the equation. There are three instantiations of the proof system described in [GS08]; we present only those based on the SXDH and the DLIN assumption (the instantiation based on the *subgroup decision* assumption is described in the full version of the paper [BFI+10]).

SXDH. The language is over a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ where p is prime. The commitment key consists of $\mathbf{u}_1 = (u_{1,1}, u_{1,2})\,, \mathbf{u}_2 = (u_{2,1}, u_{2,2})$ in $\mathbb{G}_1^2$ and $\mathbf{v}_1 = (v_{1,1}, v_{1,2})\,, \mathbf{v}_2 = (v_{2,1}, v_{2,2})$ in $\mathbb{G}_2^2$.

We write $\vec{\mathbf{u}} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix}$ and $\vec{\mathbf{v}} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{pmatrix}$.

Let $X \in \mathbb{G}_1$, $Y \in \mathbb{G}_2$ and $x \in \mathbb{Z}_p$. We define $\iota_1(X) := (1, X)$, $\iota_2(Y) := (1, Y)$, $\iota_1'(x) := (u_{2,1}^x, (u_{2,2}g_1)^x)$ and $\iota_2'(x) := (v_{2,1}^x, (v_{2,2}g_2)^x)$. To commit to $X \in \mathbb{G}_1$, one chooses randomness $s_1, s_2 \in \mathbb{Z}_p$ and sets $\mathbf{c}_X := \iota_1(X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2}$, a commitment to $Y \in \mathbb{G}_2$ is defined as $\mathbf{d}_Y := \iota_2(Y) \odot \mathbf{v}_1^{s_1} \odot \mathbf{v}_2^{s_2}$. To make a commitment to $x \in \mathbb{Z}_p$ in $\mathbb{G}_1^2$ one chooses $s \in \mathbb{Z}_p$ and sets $\mathbf{c}_x := \iota_1'(x) \odot \mathbf{u}_1^s$, a commitment in $\mathbb{G}_2^2$ is defined as $\mathbf{d}_x := \iota_2'(x) \odot \mathbf{v}_1^s$.

To show satisfiability of a set of equations of the form (1), (2) or (3), one first makes commitments to a satisfying *witness* (*i.e.* an assignment to the variables of each equation) and then adds a "proof" per equation. Groth and Sahai describe how to construct these; for Type (1), they are in $\mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$, for Type (2) they are in $\mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^2$ and for Type (3) in $\mathbb{G}_2^2 \times \mathbb{G}_1^2$.

The verification relations for the proofs are given in Section 5, where we also discuss how to optimize them. For convenience we define some notations. Let $t \in \mathbb{Z}_p$, $T_1 \in \mathbb{G}_1$, $T_2 \in \mathbb{G}_2$ and $t_T \in \mathbb{G}_T$. Then we let[4]

$$\iota_T(t_T) := \begin{pmatrix} 1 & 1 \\ 1 & t_T \end{pmatrix},\ \hat{\iota}_T(T_1) := \begin{pmatrix} 1 & 1 \\ e(T_1, v_{2,1}) & e(T_1, v_{2,2}g_2) \end{pmatrix},\ \hat{\iota}_T(T_2) := \begin{pmatrix} 1 & e(u_{2,1}, T_2) \\ 1 & e(u_{2,2}g_1, T_2) \end{pmatrix},$$

and $\iota'_T(t) := \big[(u_{2,1}, u_{2,2}g_1) \bullet (v_{2,1}, v_{2,2}g_2)\big]^t = \begin{pmatrix} e(u_{2,1}, v_{2,1})^t & e(u_{2,1}, v_{2,2}g_2)^t \\ e(u_{2,2}g_1, v_{2,1})^t & e(u_{2,2}g_1, v_{2,2}g_2)^t \end{pmatrix}$.

For the sake of consistency with [GS08], for $\mathbf{c} \in \mathbb{G}_1^{1\times 2}$ and $\mathbf{d} \in \mathbb{G}_2^{1\times 2}$ we denote $F(\mathbf{c}, \mathbf{d}) := [\mathbf{c} \bullet \mathbf{d}]$.

DLIN. In this instantiation, the language is over a bilinear (symmetric) group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ with p prime. The commitment key $\vec{\mathbf{u}} \in \mathbb{G}^{3\times 3}$ is of the form $\mathbf{u}_1 = (u_{1,1}, 1, g)$, $\mathbf{u}_2 = (1, u_{2,1}, g)$, $\mathbf{u}_3 = (u_{3,1}, u_{3,2}, u_{3,3})$. Let $X \in \mathbb{G}$ and $x \in \mathbb{Z}_p$. We define $\iota(X) := (1, 1, X)$ and $\iota'(x) := (u_{3,1}^x, u_{3,2}^x, (u_{3,3}g)^x)$. To commit to $X \in \mathbb{G}$, choose randomness $s_1, s_2, s_3 \in \mathbb{Z}_p$ and set $\mathbf{c}_X := \iota(X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} \odot \mathbf{u}_3^{s_3}$. To commit to $x \in \mathbb{Z}_p$, choose $s_1, s_2 \in \mathbb{Z}_p$ and set $\mathbf{c}_x := \iota'(x) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_3^{s_2}$.

Due to the fact that $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ in this setting, the equations (1), (2) and (3) simplify to the following respective equations:

$$\langle \vec{\mathcal{A}}, \vec{\mathcal{Y}} \rangle \cdot \langle \vec{\mathcal{Y}}, \Gamma \vec{\mathcal{Y}} \rangle = t_T \tag{1'}$$

$$\langle \vec{a}, \vec{\mathcal{Y}} \rangle \cdot \langle \vec{x}, \vec{\mathcal{B}} \rangle \cdot \langle \vec{x}, \Gamma \vec{\mathcal{Y}} \rangle = T \tag{2'}$$

$$\langle \vec{x}, \vec{b} \rangle + \langle \vec{x}, \Gamma \vec{x} \rangle = t \tag{3'}$$

Groth and Sahai show how to construct "proofs" for each type of equation, where for Types (1') and (2'), the proof is in $\mathbb{G}^{3\times 3}$, whereas for Type (3') it is in $\mathbb{G}^{2\times 3}$. The verification relations for the proofs are given in Section 6. We define the following notations. Let $t \in \mathbb{Z}_p$, $T \in \mathbb{G}$ and $t_T \in \mathbb{G}_T$. Then we let

$$\iota_T(t_T) := \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & t_T \end{pmatrix} \qquad \hat{\iota}_T(T) := \begin{pmatrix} 1 & 1 & e(u_{3,1}, T)^{\frac{1}{2}} \\ 1 & 1 & e(u_{3,2}, T)^{\frac{1}{2}} \\ e(u_{3,1}, T)^{\frac{1}{2}} & e(u_{3,2}, T)^{\frac{1}{2}} & e(u_{3,3}g, T) \end{pmatrix}$$

and $\iota'_T(t) := \big[(u_{3,1}, u_{3,2}, u_{3,3}g) \overset{s}{\bullet} (u_{3,1}, u_{3,2}, u_{3,3}g)\big]^t$.

4 Batch Verification of Pairing Equations

We address the problem of securely batching the verification of (potentially many) Groth-Sahai proofs. We achieve a trade-off between soundness and efficiency: if the verification algorithm returns valid, the verifier is assured that all proved statements are valid with overwhelming probability. Ferrara, Green,

[4] Here (and in the DLIN instantiation) we use the rectifications of $\hat{\iota}_T$ and ι'_T by [GSW09].

Hohenberger and Pedersen [FGHP09] presented a detailed study on how to securely batch-verify a set of pairing-based equations, which we briefly recall here (see the full version of [FGHP09] for any additional details).

Given a bilinear structure $(N, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, a *pairing-based verification equation* is a Boolean relation of the form: $\prod_{i=1}^{k} e(f_i, h_i)^{c_i} \stackrel{?}{=} A$ for $k \in \mathbb{N}$, $(f_i, h_i, c_i) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{Z}_N$ for $i \in \{1, \ldots, k\}$ and $A \in \mathbb{G}_T$. A *pairing-based verifier* is an algorithm which given a pairing-based verification equation outputs *yes* if the Boolean relation holds, and *no* otherwise (except with negligible probability).

In order to design a pairing-based verifier for m pairing-based verification equations, one has to find a way to combine all equations. The technique proposed in [FGHP09] consists in using the *small exponents test* proposed by Bellare et al. [BGR98], which here amounts to pick small random exponents $\delta_1, \ldots, \delta_m$ and checking whether $\prod_{j=1}^{m} \prod_{i=1}^{k_j} e(f_{i,j}, h_{i,j})^{c_{i,j}\delta_j} = \prod_{j=1}^{m} A_j^{\delta_j}$ holds. In order to further reduce the computational needs, three main techniques may be used:

1. **Move the exponent into the pairing:** Since, in practice, exponentiation in $\mathbb{G}_T$ is more expensive[5] than in $\mathbb{G}_1$ and $\mathbb{G}_2$, this gives a first speed up. As we are working on pairings, we can also do the opposite if it allows another technique to apply: $e(f_i, h_i)^{\delta_i} \to e(f_i^{\delta_i}, h_i)$
2. **Move the product into the pairing:** When two pairings have a common element, they can be combined to reduce the number of pairings:
$$\prod_{j=1}^{m} e(f_j^{\delta_j}, h_i) \to e\left(\prod_{j=1}^{m} f_j^{\delta_j},\ h_i\right)$$
3. **Switch two products:** Sometimes improvements can be made by moving a product from the first to the second component of a pairing (or vice-versa):
$$\prod_{i=1}^{k} e\left(\prod_{j=1}^{m} f_j^{\delta_{i,j}},\ h_i\right) \leftrightarrow \prod_{j=1}^{m} e\left(f_j,\ \prod_{i=1}^{k} h_i^{\delta_{i,j}}\right)$$

The soundness of the pairing-based verifier based on the small exponents test is quantified in the following theorem [FGHP09, Theorem 3.2]:

Theorem 1. *Given m pairing-based verification equations, the small-exponents verifier described above with random exponents $\delta_1, \ldots, \delta_m$ of ℓ bits is a pairing-based batch verifier that accepts an invalid batch with probability at most $2^{-\ell}$.*

Handling Invalid Proofs. In the case of verification of multiple proofs (as in Sections 7 and 8), if there is an invalid proof in the batch, then the verifier will reject the entire batch with high probability. A simple technique for finding invalid proofs in a batch consists in using a recursive *divide-and-conquer* approach [PMPS00]. Recently, more efficient techniques were proposed for pairing-based signatures (see e.g. [Mat09] and references therein) and they apply as well to our setting.

[5] Note that, for Type 2 pairings, exponentiation in $\mathbb{G}_2$ is more expensive than in $\mathbb{G}_T$ (see [GPS08] for details).

5 Instantiation 2: SXDH

5.1 Pairing-Product Equation

A proof $(\vec{\mathbf{c}}, \vec{\mathbf{d}}, \vec{\pi}, \vec{\theta}) \in \mathbb{G}_1^{m\times 2} \times \mathbb{G}_2^{n\times 2} \times \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ of satisfiability of an equation of Type (1) is verified by checking the following equation [GS08]:

$$\left[\iota_1(\vec{\mathcal{A}}) \bullet \vec{\mathbf{d}}\,\right] \odot \left[\vec{\mathbf{c}} \bullet \iota_2(\vec{\mathcal{B}})\right] \odot \left[\vec{\mathbf{c}} \bullet \Gamma \vec{\mathbf{d}}\right] = \iota_T(t_T) \odot \left[\vec{\mathbf{u}} \bullet \vec{\pi}\right] \odot \left[\vec{\theta} \bullet \vec{\mathbf{v}}\right] .$$

Let $\vec{\mathbf{c}} = (c_{i,k})_{\substack{1\le i\le m\\ 1\le k\le 2}} \in \mathbb{G}_1^{m\times 2}$, $\vec{\mathbf{d}} = (d_{j,k})_{\substack{1\le j\le n\\ 1\le k\le 2}} \in \mathbb{G}_2^{n\times 2}$, $\Gamma = (\gamma_{i,j})_{\substack{1\le i\le m\\ 1\le j\le n}} \in \mathbb{Z}_p^{m\times n}$, $\vec{\mathcal{A}} = (\mathcal{A}_j)_{1\le j\le n} \in \mathbb{G}_1^{n\times 1}$ and $\vec{\mathcal{B}} = (\mathcal{B}_i)_{1\le i\le m} \in \mathbb{G}_2^{m\times 1}$.

Plugging in the definitions from Section 3, the left hand side is equal to

$$\begin{pmatrix} \prod_{i=1}^{m} e\left(c_{i,1}, \prod_{j=1}^{n} d_{j,1}^{\gamma_{i,j}}\right) & \prod_{i=1}^{m} e\left(c_{i,1}, \mathcal{B}_i \prod_{j=1}^{n} d_{j,2}^{\gamma_{i,j}}\right) \\ \prod_{j=1}^{n} e\left(\mathcal{A}_j \prod_{i=1}^{m} c_{i,2}^{\gamma_{i,j}}, d_{j,1}\right) & \prod_{j=1}^{n} e(\mathcal{A}_j, d_{j,2}) \prod_{i=1}^{m} e\left(c_{i,2}, \mathcal{B}_i \prod_{j=1}^{n} d_{j,2}^{\gamma_{i,j}}\right) \end{pmatrix} .$$

If we denote $\vec{\pi} = \begin{pmatrix} \pi_{1,1} & \pi_{1,2} \\ \pi_{2,1} & \pi_{2,2} \end{pmatrix}$, $\vec{\theta} = \begin{pmatrix} \theta_{1,1} & \theta_{1,2} \\ \theta_{2,1} & \theta_{2,2} \end{pmatrix}$, the right hand side is equal to

$$\begin{pmatrix} e(u_{1,1}, \pi_{1,1}) e(u_{2,1}, \pi_{2,1}) & e(u_{1,1}, \pi_{1,2}) e(u_{2,1}, \pi_{2,2}) \\ \quad \cdot e(\theta_{1,1}, v_{1,1}) e(\theta_{2,1}, v_{2,1}) & \quad \cdot e(\theta_{1,1}, v_{1,2}) e(\theta_{2,1}, v_{2,2}) \\ e(u_{1,2}, \pi_{1,1}) e(u_{2,2}, \pi_{2,1}) & t_T e(u_{1,2}, \pi_{1,2}) e(u_{2,2}, \pi_{2,2}) \\ \quad \cdot e(\theta_{1,2}, v_{1,1}) e(\theta_{2,2}, v_{2,1}) & \quad \cdot e(\theta_{1,2}, v_{1,2}) e(\theta_{2,2}, v_{2,2}) \end{pmatrix} .$$

By grouping pairings, we reduced the number of pairings on the left-hand side of the equation from $5m + 3n$ to $3m + 2n$, while the right-hand side remains at 16 pairings. Using the techniques explained in Section 4, *i.e.* taking each element $M_{i,j}$ of the equation to a random power $r_{i,j}$, multiplying all the components, and regrouping pairings, we get the following equation:

$$\begin{aligned} &\prod_{k=1}^{2} \prod_{j=1}^{n} e\left(\left(\prod_{i=1}^{m} c_{i,1}^{\gamma_{i,j}}\right)^{r_{1,k}} \left(\mathcal{A}_j \prod_{i=1}^{m} c_{i,2}^{\gamma_{i,j}}\right)^{r_{2,k}}, d_{j,k}\right) \cdot \prod_{i=1}^{m} e\left(c_{i,1}^{r_{1,2}} c_{i,2}^{r_{2,2}}, \mathcal{B}_i\right) \\ &= e(u_{1,1}^{r_{1,1}} u_{1,2}^{r_{2,1}}, \pi_{1,1}) e(u_{2,1}^{r_{1,1}} u_{2,2}^{r_{2,1}}, \pi_{2,1}) e(\theta_{1,1}^{r_{1,1}} \theta_{1,2}^{r_{2,1}}, v_{1,1}) e(\theta_{2,1}^{r_{1,1}} \theta_{2,2}^{r_{2,1}}, v_{2,1}) \\ &\qquad \cdot e(u_{1,1}^{r_{1,2}} u_{1,2}^{r_{2,2}}, \pi_{1,2}) e(u_{2,1}^{r_{1,2}} u_{2,2}^{r_{2,2}}, \pi_{2,2}) e(\theta_{1,1}^{r_{1,2}} \theta_{1,2}^{r_{2,2}}, v_{1,2}) e(\theta_{2,1}^{r_{1,2}} \theta_{2,2}^{r_{2,2}}, v_{2,2}) \cdot t_T^{r_{2,2}} \end{aligned}$$

which requires $m + 2n$ pairings and $2mn + 2m + 4n$ exponentiations in $\mathbb{G}_1$ for the left part and 8 pairing computations and 16 exponentiations in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_T$ for the right side of the equation. The alternative expression

$$\prod_{j=1}^{n} e\left(\mathcal{A}_j, d_{j,1}^{r_{2,1}} d_{j,2}^{r_{2,2}}\right) \cdot \prod_{k=1}^{2} \prod_{i=1}^{m} e\left(c_{i,k}, \left(\prod_{j=1}^{n} d_{j,1}^{\gamma_{i,j}}\right)^{r_{k,1}} \left(\mathcal{B}_i \prod_{j=1}^{n} d_{j,2}^{\gamma_{i,j}}\right)^{r_{k,2}}\right)$$

for the left side of the equation requires $2m + n$ pairings and $2mn + 4m + 2n$ exponentiations in $\mathbb{G}_2$.

5.2 Multi-scalar Multiplication Equation in $\mathbb{G}_1$

Here, we consider equations of Type (2) in $\mathbb{G}_1$ (the case of equations in $\mathbb{G}_2$, which work analogously, is treated in the full version of the paper [BFI+10]). The verification of a proof $(\vec{\mathbf{c}}, \vec{\mathbf{d}}', \vec{\pi}, \theta) \in \mathbb{G}_1^{m\times 2} \times \mathbb{G}_2^{n\times 2} \times \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{1\times 2}$ consists in checking the following:

$$\left[\iota_1(\vec{\mathcal{A}}) \bullet \vec{\mathbf{d}}'\right] \odot \left[\vec{\mathbf{c}} \bullet \iota_2'(\vec{b})\right] \odot \left[\vec{\mathbf{c}} \bullet \varGamma \vec{\mathbf{d}}'\right] = \hat{\iota_T}(\mathcal{T}_1) \odot \left[\vec{\mathbf{u}} \bullet \vec{\pi}\right] \odot F(\theta, \mathbf{v}_1).$$

Let $\vec{\mathbf{c}} = (c_{i,k})_{\substack{1\le i\le m\\ 1\le k\le 2}} \in \mathbb{G}_1^{m\times 2}$, $\vec{\mathbf{d}}' = (d'_{j,k})_{\substack{1\le j\le n\\ 1\le k\le 2}} \in \mathbb{G}_2^{n\times 2}$, $\varGamma = (\gamma_{i,j})_{\substack{1\le i\le m\\ 1\le j\le n}} \in \mathbb{Z}_p^{m\times n}$, $\vec{\mathcal{A}} = (\mathcal{A}_j)_{1\le j\le n} \in \mathbb{G}_1^{n\times 1}$, $\vec{b} = (b_i)_{1\le i\le m} \in \mathbb{Z}_p^{m\times 1}$. The left hand-side is equal to

$$\begin{pmatrix} \prod_{i=1}^{m} e(c_{i,1}, v_{2,1}^{b_i} \prod_{j=1}^{n} {d'_{j,1}}^{\gamma_{i,j}}) & \prod_{i=1}^{m} e(c_{i,1}, (v_{2,2}g_2)^{b_i} \prod_{j=1}^{n} {d'_{j,2}}^{\gamma_{i,j}}) \\ \prod_{i=1}^{m} e(c_{i,2}, v_{2,1}^{b_i} \prod_{j=1}^{n} {d'_{j,1}}^{\gamma_{i,j}}) & \prod_{i=1}^{m} e(c_{i,2}, (v_{2,2}g_2)^{b_i} \prod_{j=1}^{n} {d'_{j,2}}^{\gamma_{i,j}}) \\ \cdot \prod_{j=1}^{n} e(\mathcal{A}_j, d'_{j,1}) & \cdot \prod_{j=1}^{n} e(\mathcal{A}_j, d'_{j,2}) \end{pmatrix}$$

while the right-hand side is equal to

$$\begin{pmatrix} e(\theta_1, v_{1,1})e(u_{1,1}, \pi_{1,1})e(u_{2,1}, \pi_{2,1}) & e(\theta_1, v_{1,2})e(u_{1,1}, \pi_{1,2})e(u_{2,1}, \pi_{2,2}) \\ e(\theta_2, v_{1,1})e(u_{1,2}, \pi_{1,1})e(u_{2,2}, \pi_{2,1}) & e(\theta_2, v_{1,2})e(u_{1,2}, \pi_{1,2})e(u_{2,2}, \pi_{2,2}) \\ \cdot e(\mathcal{T}_1, v_{2,1}) & \cdot e(\mathcal{T}_1, g_2 v_{2,2}) \end{pmatrix}$$

By grouping the pairings, the number of pairings on the left-hand side of the equation has already been reduced from $8m+2n$ to $4m+2n$. Now, by using the batch technique, i.e., multiplying each member by a random value and multiplying all the components, we obtain on the left-hand side

$$\prod_{k=1}^{2}\prod_{j=1}^{n} e\left(\left(\prod_{i=1}^{m} c_{i,1}^{\gamma_{i,j}}\right)^{r_{1,k}} \left(\mathcal{A}_j \prod_{i=1}^{m} c_{i,2}^{\gamma_{i,j}}\right)^{r_{2,k}}, d'_{j,k}\right)$$
$$\cdot\, e\left((\textstyle\prod_{i=1}^{m} c_{i,1}^{b_i})^{r_{1,1}}(\prod_{i=1}^{m} c_{i,2}^{b_i})^{r_{2,1}}, v_{2,1}\right) \cdot e\left((\textstyle\prod_{i=1}^{m} c_{i,1}^{b_i})^{r_{1,2}}(\prod_{i=1}^{m} c_{i,2}^{b_i})^{r_{2,2}}, v_{2,2}g_2\right)$$

which requires $2mn+2m+4n+4$ exponentiations in $\mathbb{G}_1$ and $2n+2$ pairing computations. The alternative expression

$$\prod_{j=1}^{n} e\Big(\mathcal{A}_j, \prod_{k=1}^{2} {d'_{j,k}}^{r_{2,k}}\Big) \prod_{k=1}^{2}\prod_{i=1}^{m} e\Big(c_{i,k}, \Big(v_{2,1}^{b_i} \prod_{j=1}^{n} {d'_{j,1}}^{\gamma_{i,j}}\Big)^{r_{k,1}} \Big((v_{2,2}g_2)^{b_i} \prod_{j=1}^{n} {d'_{j,2}}^{\gamma_{i,j}}\Big)^{r_{k,2}}\Big)$$

for the left side of the equation requires $2mn+6m+2n$ exponentiations in $\mathbb{G}_2$ and $2m+n$ pairing computations. On the right-hand side, the same technique achieves a reduction from 14 to 7 pairings:

$$e(\theta_1^{r_{1,1}}\theta_2^{r_{2,1}}, v_{1,1})e(\theta_1^{r_{2,1}}\theta_2^{r_{2,2}}, v_{1,2})e(u_{1,1}^{r_{1,1}}u_{1,2}^{r_{2,1}}, \pi_{1,1})e(u_{2,1}^{r_{1,1}}u_{2,2}^{r_{2,1}}, \pi_{2,1})$$
$$\cdot\, e(u_{2,1}^{r_{1,2}}u_{2,2}^{r_{2,2}}, \pi_{1,2})e(u_{2,1}^{r_{2,1}}u_{2,2}^{r_{2,2}}, \pi_{2,2})e(\mathcal{T}_1, v_{2,1}^{r_{2,1}}(g_2v_{2,2})^{r_{2,2}})$$

5.3 Quadratic Equation

The verification of $(\vec{\mathbf{c}}', \vec{\mathbf{d}}', \pi, \theta) \in \mathbb{G}_1^{m\times 2} \times \mathbb{G}_2^{n\times 2} \times \mathbb{G}_2^{1\times 2} \times \mathbb{G}_1^{1\times 2}$ for an equation of Type (3) consists in checking

$$\left[\iota_1'(\vec{a}) \bullet \vec{\mathbf{d}'}\right] \odot \left[\vec{\mathbf{c}'} \bullet \iota_2'(\vec{b})\right] \odot \left[\vec{\mathbf{c}'} \bullet \Gamma\vec{\mathbf{d}'}\right] = \iota_T'(t) \odot F(\mathbf{u}_1, \pi) \odot F(\theta, \mathbf{v}_1) \ .$$

Let $\vec{\mathbf{c}'} = (c'_{i,k})_{\substack{1\le i\le m\\1\le k\le 2}} \in \mathbb{G}_1^{n\times 2}$, $\vec{\mathbf{d}'} = (d'_{j,k})_{\substack{1\le j\le n\\1\le k\le 2}} \in \mathbb{G}_2^{n\times 2}$, $\Gamma = (\gamma_{i,j})_{\substack{1\le i\le m\\1\le j\le n}} \in \mathbb{Z}_p^{m\times n}$, $\vec{a} = (a_j)_{1\le j\le n} \in \mathbb{Z}_p^{n\times 1}$, $\vec{b} = (b_i)_{1\le i\le m} \in \mathbb{Z}_p^{m\times 1}$. The left hand side is equal to

$$\begin{pmatrix} \prod_{i=1}^m e(c'_{i,1}, v_{2,1}^{b_i}) \cdot \prod_{j=1}^n e\left(u_{2,1}^{a_j} \prod_{i=1}^m {c'_{i,1}}^{\gamma_{i,j}}, d'_{j,1}\right) & \prod_{i=1}^m e(c'_{i,1}, (v_{2,2}g_2)^{b_i}) \cdot \prod_{j=1}^n e\left(u_{2,1}^{a_j} \prod_{i=1}^m {c'_{i,1}}^{\gamma_{i,j}}, d'_{j,2}\right) \\ \prod_{i=1}^m e(c'_{i,2}, v_{2,1}^{b_i}) \cdot \prod_{j=1}^n e\left((u_{2,2}g_1)^{a_j} \prod_{i=1}^m {c'_{i,2}}^{\gamma_{i,j}}, d'_{j,1}\right) & \prod_{i=1}^m e(c'_{i,2}, (v_{2,2}g_2)^{b_i}) \cdot \prod_{j=1}^n e\left((u_{2,2}g_1)^{a_j} \prod_{i=1}^m {c'_{i,2}}^{\gamma_{i,j}}, d'_{j,2}\right) \end{pmatrix}$$

Denoting $\pi = (\pi_1, \pi_2)$ and $\theta = (\theta_1, \theta_2)$, for the right-hand side we have

$$\begin{pmatrix} e(u_{1,1}, \pi_1)e(\theta_1, v_{1,1})e(u_{2,1}, v_{2,1})^t & e(u_{1,1}, \pi_2)e(\theta_1, v_{1,2})e(u_{2,1}, v_{2,2}g_2)^t \\ e(u_{1,2}, \pi_1)e(\theta_2, v_{1,1})e(u_{2,2}g_1, v_{2,1})^t & e(u_{1,2}, \pi_2)e(\theta_2, v_{1,2})e(u_{2,2}g_1, v_{2,2}g_2)^t \end{pmatrix}$$

By grouping the pairings, the number of pairings on the left-hand side member of the equation has been reduced from $8m + 8n$ to $4m + 4n$. By using the batch technique, i.e., multiplying each member by a random value and multiplying all the members, we obtain on the left-hand side:

$$e\Big(\Big(\prod_{i=1}^m {c'_{i,1}}^{b_i}\Big)^{r_{1,1}} \Big(\prod_{i=1}^m {c'_{i,2}}^{b_i}\Big)^{r_{2,1}}, v_{2,1}\Big) \cdot e\Big(\Big(\prod_{i=1}^m {c'_{i,1}}^{b_i}\Big)^{r_{1,2}} \Big(\prod_{i=1}^m {c'_{i,2}}^{b_i}\Big)^{r_{2,2}}, v_{2,2}g_2\Big)$$
$$\cdot \prod_{k=1}^2 \prod_{j=1}^n e\Big(\left(u_{2,1}^{a_j} \prod_{i=1}^m {c'_{i,1}}^{\gamma_{i,j}}\right)^{r_{1,k}} \left((u_{2,2}g_1)^{a_j} \prod_{i=1}^m {c'_{i,2}}^{\gamma_{i,j}}\right)^{r_{2,k}}, d'_{j,k}\Big)$$

which requires $2mn + 2m + 6n + 4$ exponentiations in $\mathbb{G}_1$ and $2n + 2$ pairing computations. Alternatively, the left-hand side is also equal to

$$e\Big(u_{2,1}, \Big(\prod_{j=1}^n {d'_{j,1}}^{a_j}\Big)^{r_{1,1}} \Big(\prod_{j=1}^n {d'_{j,2}}^{a_j}\Big)^{r_{1,2}}\Big) \cdot e\Big(u_{2,2}g_2, \Big(\prod_{j=1}^n {d'_{j,1}}^{a_j}\Big)^{r_{2,1}} \Big(\prod_{j=1}^n {d'_{j,2}}^{a_j}\Big)^{r_{2,2}}\Big)$$
$$\cdot \prod_{k=1}^2 \prod_{i=1}^m e\Big(c'_{i,k}, \Big(v_{2,1}^{b_i} \prod_{j=1}^n {d'_{j,1}}^{\gamma_{i,j}}\Big)^{r_{k,1}} \Big((v_{2,2}g_2)^{b_i} \prod_{j=1}^n {d'_{j,2}}^{\gamma_{i,j}}\Big)^{r_{k,2}}\Big)$$

which requires $2mn + 6m + 2n + 4$ exponentiations in $\mathbb{G}_2$ and $2m + 2$ pairing computations. On the right-hand side, the same technique achieves a reduction from 12 to 6 pairings:

$$e(u_{1,1}^{r_{1,1}}u_{1,2}^{r_{2,1}},\pi_1)e(u_{1,1}^{r_{1,2}}u_{1,2}^{r_{2,2}},\pi_2)e(\theta_1^{r_{1,1}}\theta_2^{r_{2,1}},v_{1,1})e(\theta_1^{r_{1,2}}\theta_2^{r_{2,2}},v_{1,2})$$
$$\cdot\ e(u_{2,1}^{r_{1,1}t}(u_{2,2}g_1)^{r_{2,1}t},v_{2,1})e(u_{2,1}^{r_{1,2}t}(u_{2,2}g_1)^{r_{2,2}t},v_{2,2}g_2)$$

6 Instantiation 3: DLIN

6.1 Pairing-Product Equation

The verification relation of a proof $(\vec{\mathbf{d}},\phi)\in\mathbb{G}^{n\times 3}\times\mathbb{G}^{3\times 3}$ for equation Type (1′) is the following:

$$\left[\iota(\vec{\mathcal{A}})\overset{s}{\bullet}\vec{\mathbf{d}}\right]\odot\left[\vec{\mathbf{d}}\overset{s}{\bullet}\Gamma\vec{\mathbf{d}}\right]=\iota_T(t_T)\odot\left[\vec{\mathbf{u}}\overset{s}{\bullet}\vec{\phi}\right]$$

For simplicity, we consider the squares of all $\mathbb{G}_T$ elements on both sides of the equation. Writing $\Gamma\vec{\mathbf{d}}$ as $\left(\prod_{k=1}^n d_{k,j}^{\gamma_{i,k}}\right)_{\substack{1\le i\le n\\1\le j\le 3}}$ and replacing the bilinear product $\overset{s}{\bullet}$ by its definition, we get for the left-hand side:

$$\begin{pmatrix}
\prod_{i=1}^n e(d_{i,1},\prod d_{k,1}^{\gamma_{i,k}})^2 & \prod_{i=1}^n e(d_{i,1},\prod d_{k,2}^{\gamma_{i,k}})\cdot e(d_{i,2},\prod d_{k,1}^{\gamma_{i,k}}) & \prod_{i=1}^n e(\mathcal{A}_i,d_{i,1})e(d_{i,1},\prod d_{k,3}^{\gamma_{i,k}})\cdot e(d_{i,3},\prod d_{k,1}^{\gamma_{i,k}}) \\
\prod_{i=1}^n e(d_{i,2},\prod d_{k,1}^{\gamma_{i,k}})\cdot e(d_{i,1},\prod d_{k,2}^{\gamma_{i,k}}) & \prod_{i=1}^n e(d_{i,2},\prod d_{k,2}^{\gamma_{i,k}})^2 & \prod_{i=1}^n e(\mathcal{A}_i,d_{i,2})e(d_{i,2},\prod d_{k,3}^{\gamma_{i,k}})\cdot e(d_{i,3},\prod d_{k,2}^{\gamma_{i,k}}) \\
\prod_{i=1}^n e(\mathcal{A}_i,d_{i,1})\cdot e(d_{i,3},\prod d_{k,1}^{\gamma_{i,k}})\cdot e(d_{i,1},\prod d_{k,3}^{\gamma_{i,k}}) & \prod_{i=1}^n e(\mathcal{A}_i,d_{i,2})\cdot e(d_{i,3},\prod d_{k,2}^{\gamma_{i,k}})\cdot e(d_{i,2},\prod d_{k,3}^{\gamma_{i,k}}) & \prod_{i=1}^n e(\mathcal{A}_i,d_{i,3})^2\cdot e(d_{i,3},\prod d_{k,3}^{\gamma_{i,k}})^2
\end{pmatrix}$$

For the right-hand side, we get:

$$\begin{pmatrix}
\prod_{i=1}^3 e(u_{i1},\phi_{i1})^2 & \prod_{i=1}^3 e(u_{i1},\phi_{i2})e(u_{i2},\phi_{i1}) & \prod_{i=1}^3 e(u_{i1},\phi_{i3})e(u_{i3},\phi_{i1}) \\
\prod_{i=1}^3 e(u_{i2},\phi_{i1})e(u_{i1},\phi_{i2}) & \prod_{i=1}^3 e(u_{i2},\phi_{i2})^2 & \prod_{i=1}^3 e(u_{i2},\phi_{i3})e(u_{i3},\phi_{i2}) \\
\prod_{i=1}^3 e(u_{i3},\phi_{i1})e(u_{i1},\phi_{i3}) & \prod_{i=1}^3 e(u_{i3},\phi_{i2})e(u_{i2},\phi_{i3}) & t_T^2\prod_{i=1}^3 e(u_{i3},\phi_{i3})^2
\end{pmatrix}$$

Taking each component $M_{i,j}$ of the equation to the power of $r_{i,j}$, multiplying all components, and regrouping pairings, we get the following for the left-hand side:

$$\prod_{i=1}^{n} e\big(d_{i,1},\ \mathcal{A}_i^{r_{1,3}+r_{3,1}} \prod d_{k,1}^{\gamma_{i,k} 2\cdot r_{1,1}} d_{k,2}^{\gamma_{i,k}(r_{1,2}+r_{2,1})} d_{k,3}^{\gamma_{i,k}(r_{1,3}+r_{3,1})}\big) \cdot \\ e\big(d_{i,2},\ \mathcal{A}_i^{r_{2,3}+r_{3,2}} \prod d_{k,1}^{\gamma_{i,k}(r_{1,2}+r_{2,1})} d_{k,2}^{\gamma_{i,k} 2\cdot r_{2,2}} d_{k,3}^{\gamma_{i,k}(r_{2,3}+r_{3,2})}\big) \cdot \\ e\big(d_{i,3},\ \mathcal{A}_i^{2\cdot r_{3,3}} \prod d_{k,1}^{\gamma_{i,k}(r_{1,3}+r_{3,1})} d_{k,2}^{\gamma_{i,k}(r_{2,3}+r_{3,2})} d_{k,3}^{\gamma_{i,k} 2 r_{3,3}}\big) \quad (4)$$

and for the right-hand side:

$$\prod_{i=1}^{3} e\big(u_{i,1},\ \phi_{i,1}^{2\cdot r_{1,1}} \phi_{i,2}^{r_{1,2}+r_{2,1}} \phi_{i,3}^{r_{1,3}+r_{3,1}}\big) \cdot e\big(u_{i,2},\ \phi_{i,1}^{r_{1,2}+r_{2,1}} \phi_{i,2}^{2\cdot r_{2,2}} \phi_{i,3}^{r_{2,3}+r_{3,2}}\big) \\ \cdot e\big(u_{i,3},\ \phi_{i,1}^{r_{1,3}+r_{3,1}} \phi_{i,2}^{r_{2,3}+r_{3,2}} \phi_{i,3}^{2\cdot r_{3,3}}\big) \cdot t_T^{2 r_{3,3}}$$

Due to the fact that $u_{1,2} = u_{2,1} = 1$, and $u_{1,3} = u_{2,3}$ (*cf.* Section 3) this simplifies to:

$$e\big(u_{1,1},\ \phi_{1,1}^{2\cdot r_{1,1}} \phi_{1,2}^{r_{1,2}+r_{2,1}} \phi_{1,3}^{r_{1,3}+r_{3,1}}\big) \cdot e\big(u_{2,2},\ \phi_{2,1}^{r_{1,2}+r_{2,1}} \phi_{2,2}^{2\cdot r_{2,2}} \phi_{2,3}^{r_{2,3}+r_{3,2}}\big) \\ \cdot e\big(u_{1,3},\ (\phi_{1,1}\phi_{2,1})^{r_{1,3}+r_{3,1}} (\phi_{1,2}\phi_{2,2})^{r_{2,3}+r_{3,2}} (\phi_{1,3}\phi_{2,3})^{2\cdot r_{3,3}}\big) \\ \cdot e\big(u_{3,1},\ \phi_{3,1}^{2\cdot r_{1,1}} \phi_{3,2}^{r_{1,2}+r_{2,1}} \phi_{3,3}^{r_{1,3}+r_{3,1}}\big) \cdot e\big(u_{3,2},\ \phi_{3,1}^{r_{1,2}+r_{2,1}} \phi_{3,2}^{2\cdot r_{2,2}} \phi_{3,3}^{r_{2,3}+r_{3,2}}\big) \cdot \\ e\big(u_{3,3},\ \phi_{3,1}^{r_{1,3}+r_{3,1}} \phi_{3,2}^{r_{2,3}+r_{3,2}} \phi_{3,3}^{2\cdot r_{3,3}}\big) \cdot t_T^{2 r_{3,3}}$$

We reduce the number of pairings from $12n+27$ to $3n+6$ pairings at the expense of adding $9n^2 + 3n$ exponentiations in $\mathbb{G}$ and one exponentiation in $\mathbb{G}_T$.

6.2 Multi-scalar Multiplication and Quadratic Equations

The description of the batch verification of multi-scalar multiplication equation and quadratic equation is similar to the previous one. Due to space constraints it is given in the full version [BFI+10].

7 Application 1: Groth's Group Signatures

7.1 Description

We demonstrate our techniques by applying them to one of the most practical fully-secure group-signature schemes in the standard model to date: Groth's construction [Gro07]. Groth proposed a methodology of transforming *certified signatures* [BFPW07] that respect a certain structure into group signatures using Groth-Sahai NIWI proofs:

– a member picks a key pair for the certified-signature scheme and asks the issuer to certify her verification key;
– to produce a group signature, the member makes a certified signature, encrypts it and makes a NIWI proof that demonstrates that the ciphertext contains a valid certified signature.

Groth proposed an efficient certified-signature scheme based on the so called q-**U** assumption (see [Gro07] for details). In the CPA-anonymous version[6] of the scheme, the issuer's public key is a triple $(f, h, T) \in \mathbb{G}^2 \times \mathbb{G}_T$ (and its private key is $z \in \mathbb{G}$ such that $e(f, z) = T$) and the certificate for a group member with public key $v = g^x \in \mathbb{G}$ is a pair (a, b) satisfying $e(a, vh)\, e(f, b) = T$. To sign a message $m \in \mathbb{Z}_p$, the group member first computes a weak Boneh-Boyen signature [BB08] $\sigma = g^{1/(x+m)}$ using her private key x; then she forms Groth-Sahai commitments $\mathbf{d}_v$, $\mathbf{d}_b$ and $\mathbf{d}_\sigma$ to the group elements v, b and σ, resp., and makes a proof that they satisfy the following:

$$e(a, vh)\, e(f, b) = T \qquad\qquad e(\sigma, g^m v) = e(g, g) \tag{5}$$

The fact that a is given in the clear is not a problem since the certificate is malleable, so the group member can unlinkably re-randomize it each time she signs a message. A group signature is thus of the form $(a, \mathbf{d}_b, \mathbf{d}_v, \mathbf{d}_\sigma, \psi, \phi)$, where ψ and ϕ denote the Groth-Sahai proofs for the two equations in (5), respectively.

We first instantiate our generic batch construction to verify a single signature more efficiently and then show how to verify multiple signatures at once. The first equation is of a particular form that allows for more efficient proofs and verification. We describe the verification relations and the batch verification in the next section.

7.2 Batching Linear Pairing-Product Equations

We consider a special case of pairing-product equations for which $\Gamma = 0$, called *linear equations*, *i.e.* the equation is of the following form: $\langle \vec{\mathcal{A}}, \vec{\mathcal{Y}} \rangle = t_T$, that is $\prod_{i=1}^n e(\mathcal{A}_i, \mathcal{Y}_i) = t_T$. In this case, the proof simplifies to three group elements and is verified as follows (taking into account that $u_{1,2} = u_{2,1} = 1$, and $u_{1,3} = u_{2,3}$):

$$\begin{aligned}
\textstyle\prod_{i=1}^n e(\mathcal{A}_i, d_{i,1}) &= e(u_{11}, \psi_1)\, e(u_{31}, \psi_3) \\
\textstyle\prod_{i=1}^n e(\mathcal{A}_i, d_{i,2}) &= e(u_{22}, \psi_2)\, e(u_{32}, \psi_3) \\
\textstyle\prod_{i=1}^n e(\mathcal{A}_i, d_{i,3}) &= t_T\, e(u_{13}, \psi_1 \psi_2)\, e(u_{33}, \psi_3)
\end{aligned}$$

which can be batch-verified by checking[7]

$$\begin{aligned}
&\prod_{i=1}^n e(\mathcal{A}_i, d_{i,1}^{s_1} d_{i,2}^{s_2} d_{i,3}^{s_3}) \\
&= t_T^{s_3}\, e(u_{11}, \psi_1^{s_1})\, e(u_{13}, (\psi_1\psi_2)^{s_3})\, e(u_{22}, \psi_2^{s_2})\, e(u_{31}, \psi_3^{s_1})\, e(u_{32}, \psi_3^{s_2})\, e(u_{33}, \psi_3^{s_3})\ .
\end{aligned} \tag{6}$$

[6] Groth also proposes group signatures achieving CCA-anonymity [BSZ05]; for illustrative purposes we restrict ourselves to the basic CPA-anonymous scheme here.

[7] If we considered a single set of equations then it would be more efficient to order the right-hand side by the ψ_i's and save 3 pairings. We order by the u_{ij} though, since this enables us to batch with other equations containing pairings of these constants.

7.3 Batching the Equations for One Group Signature

1st Equation. Instantiating (6) for first equation in (5), we get, after some more optimization (shifting $e(a, h^{-1}))^{s_3}$ to the left-hand side of the equation)

$$\begin{aligned}
&e(d_{v,1}^{s_1} d_{v,2}^{s_2} (d_{v,3}h)^{s_3},\, a)\, e(d_{b,1}^{s_1} d_{b,2}^{s_2} d_{b,3}^{s_3},\, f) \;= \\
&T^{s_3}\, e(u_{11}, \psi_1^{s_1})\, e(u_{13}, (\psi_1\psi_2)^{s_3})\, e(u_{22}, \psi_2^{s_2})\, e(u_{31}, \psi_3^{s_1})\, e(u_{32}, \psi_3^{s_2})\, e(u_{33}, \psi_3^{s_3})
\end{aligned}$$

2nd Equation. Setting $\vec{\mathcal{A}} := \begin{pmatrix} g^m \\ 1 \end{pmatrix}$, $\vec{\mathcal{Y}} := \begin{pmatrix} \sigma \\ v \end{pmatrix}$, $\Gamma := \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, $t_T := e(g, g)$, $\mathbf{d}_1 := \mathbf{d}_\sigma$ and $\mathbf{d}_2 := \mathbf{d}_v$ and substituting in (4), we get

$$\begin{aligned}
&e\big(d_{\sigma 1},\ (g^m d_{v3})^{(r_{13}+r_{31})} d_{v1}^{2\cdot r_{11}} d_{v2}^{(r_{12}+r_{21})}\big)\, e\big(d_{\sigma 2},\ (g^m d_{v3})^{(r_{23}+r_{32})} d_{v1}^{(r_{12}+r_{21})} d_{v2}^{2\cdot r_{22}}\big) \\
&\cdot e\big(d_{\sigma 3},\ (g^m d_{v3})^{2\cdot r_{33}} d_{v1}^{(r_{13}+r_{31})} d_{v2}^{(r_{23}+r_{32})}\big) \;= \\
&e\big(u_{11},\ \phi_{11}^{2\cdot r_{11}} \phi_{12}^{r_{12}+r_{21}} \phi_{13}^{r_{13}+r_{31}}\big)\, e\big(u_{13},\ (\phi_{11}\phi_{21})^{r_{13}+r_{31}} (\phi_{12}\phi_{22})^{r_{23}+r_{32}} (\phi_{13}\phi_{23})^{2\cdot r_{33}}\big) \\
&\qquad \cdot e\big(u_{22},\ \phi_{21}^{r_{12}+r_{21}} \phi_{22}^{2\cdot r_{22}} \phi_{23}^{r_{23}+r_{32}}\big)\, e\big(u_{31},\ \phi_{31}^{2\cdot r_{11}} \phi_{32}^{r_{12}+r_{21}} \phi_{33}^{r_{13}+r_{31}}\big) \\
&\qquad \cdot e\big(u_{32},\ \phi_{31}^{r_{12}+r_{21}} \phi_{32}^{2\cdot r_{22}} \phi_{33}^{r_{23}+r_{32}}\big)\, e\big(u_{33},\ \phi_{31}^{r_{13}+r_{31}} \phi_{32}^{r_{23}+r_{32}} \phi_{33}^{2\cdot r_{33}}\big)\, e(g, g^{2r_{33}})\ .
\end{aligned}$$

Multiplying the two equations we get a single verification relation of the following form:

$$\begin{aligned}
&e(d_{v,1}^{s_1} d_{v,2}^{s_2} (d_{v,3}h)^{s_3}, a)\, e(d_{b,1}^{s_1} d_{b,2}^{s_2} d_{b,3}^{s_3}, f) e\big(d_{\sigma 1},\ (g^m d_{v3})^{(r_{13}+r_{31})} d_{v1}^{2\cdot r_{11}} d_{v2}^{(r_{12}+r_{21})}\big) \\
&\cdot e\big(d_{\sigma 2},\ (g^m d_{v3})^{(r_{23}+r_{32})} d_{v1}^{(r_{12}+r_{21})} d_{v2}^{2\cdot r_{22}}\big)\, e\big(d_{\sigma 3},\ (g^m d_{v3})^{2\cdot r_{33}} d_{v1}^{(r_{13}+r_{31})} d_{v2}^{(r_{23}+r_{32})}\big) \\
&\quad = \big(T^{s_3} e(g, g^{2r_{33}})\big) e\big(u_{13},\ (\phi_{11}\phi_{21})^{r_{13}+r_{31}} (\phi_{12}\phi_{22})^{r_{23}+r_{32}} (\phi_{13}\phi_{23})^{2\cdot r_{33}} (\psi_1\psi_2)^{s_3}\big) \\
&\qquad e\big(u_{11},\ \phi_{11}^{2\cdot r_{11}} \phi_{12}^{r_{12}+r_{21}} \phi_{13}^{r_{13}+r_{31}} \psi_1^{s_1}\big) \cdot e\big(u_{22},\ \phi_{21}^{r_{12}+r_{21}} \phi_{22}^{2\cdot r_{22}} \phi_{23}^{r_{23}+r_{32}} \psi_2^{s_2}\big) \\
&\qquad e\big(u_{31},\ \phi_{31}^{2\cdot r_{11}} \phi_{32}^{r_{12}+r_{21}} \phi_{33}^{r_{13}+r_{31}} \psi_3^{s_1}\big) \cdot e\big(u_{32},\ \phi_{31}^{r_{12}+r_{21}} \phi_{32}^{2\cdot r_{22}} \phi_{33}^{r_{23}+r_{32}} \psi_3^{s_2}\big) \\
&\qquad \cdot e\big(u_{33},\ \phi_{31}^{r_{13}+r_{31}} \phi_{32}^{r_{23}+r_{32}} \phi_{33}^{2\cdot r_{33}} \psi_3^{s_3}\big)
\end{aligned}$$

Analysis. With no use of batching techniques, the verification of a single signature takes for the first equation 13 pairings and for the second 20 pairings for the left-hand side and 35 for its right-hand side. This is an overall of 68 pairing evaluations, compared to 11 for the batched verification.

7.4 Batching Several Group Signatures

Consider the situation where we want to verify multiple group signatures at once. That is given a group public key $(f, h, T, u_{11}, u_{13}, u_{22}, u_{31}, u_{32}, u_{33})$ and n group signatures

$$\Big(a^{(k)}, \mathbf{d}_b^{(k)}, \mathbf{d}_v^{(k)}, \mathbf{d}_\sigma^{(k)}, (\psi_i^{(k)})_{1\le i\le 3}, (\phi_{ij}^{(k)})_{1\le i,j\le 3}\Big)\ .$$

Using the same technique of taking each of the (new) equations to the power of some randomness and multiplying them, we can unify the pairings $e(\cdot, f)$ on the left-hand side and all pairings (which are of the form $e(u_{ij}, \cdot)$) on the right-hand side. Instead of $11n$ pairings needed when checking each equation, the batched version only requires $4n + 7$ pairings.

8 Application 2: P-Signatures

8.1 Description

Belenkiy *et al.* [BCKL08] formalize digital signature schemes with an additional non-interactive proof of signature possession that they called *P-signature schemes*. They proposed two constructions[8]: the first is based on the weak Boneh-Boyen signature scheme [BB08] while the second one is inspired by its full version.

Since Belenkiy *et al.*'s first scheme relies on a rather strong assumption, we consider only their second proposal: a signature σ on a message $m \in \mathbb{Z}_p$ is a triple $\sigma = (C_1, C_2, C_3) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$ such that $e(C_1, vh^m C_2) = e(g, h)$ and $e(f, C_2) = e(C_3, w)$, where f and g are (public) generators of $\mathbb{G}_1$, h is a (public) generator of $\mathbb{G}_2$ and $v, w \in \mathbb{G}_2$ are parts of the signer's public key. To prove possession of such a signature, a prover forms the Groth-Sahai commitments $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{c_3}$ for the group elements $C_1, M_1 = f^m, C_3$ in $\mathbb{G}_1$ and $\mathbf{d_1}$ and $\mathbf{d_2}$ for the group elements $M_2 = h^m$ and C_2 in $\mathbb{G}_2$ (respectively) and provides a proof that they satisfy:

$$e(C_1, vM_2C_2) = e(g, h), \quad e(f, C_2) = e(C_3, w) \quad \text{and} \quad e(f, M_2) = e(M_1, h) \quad (7)$$

8.2 SXDH Instantiation

In [BCKL08], the authors evaluated that the verification of the proof in the SXDH instantiation requires the computation of 68 pairings. In the full version of this paper [BFI+10] we show that it can be reduced to 15.

8.3 DLIN Instantiation

As in Section 7, the last two pairing-product equations from (7) are actually linear pairing-product equations. We denote the Groth-Sahai commitments for the group elements $C_1, C_2, C_3, M_1 = f^m, M_2 = h^m$ in $\mathbb{G}$ by $\mathbf{d_1}$,$\mathbf{d_2}$, $\mathbf{d_3}$, $\mathbf{d_4}$ and $\mathbf{d_5}$ (respectively) and by ϕ, ψ and θ the proofs that they satisfy the first, the second and the third equation (respectively). For the first equation, setting and substituting

$$\vec{\mathcal{A}} = \begin{pmatrix} v \\ 1 \\ 1 \end{pmatrix}, \ \vec{\mathbf{d}} = \begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \mathbf{d}_5 \end{pmatrix}, \ \Gamma = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad t_T = e(g, g)$$

in (4), we get:

[8] An extended version of their scheme was recently proposed [BCKL09] but here we restrict ourselves to the basic scheme from [BCKL08].

$$\begin{aligned}
&e\big(d_{1,1},\ (v\, d_{2,3}\, d_{5,3})^{r_{1,3}+r_{3,1}}(d_{2,1}\, d_{5,1})^{2r_{1,1}}(d_{2,2}\, d_{5,2})^{(r_{1,2}+r_{2,1})}\big)\\
&\qquad\cdot e\big(d_{1,2},\ (v\, d_{2,3}\, d_{5,3})^{r_{2,3}+r_{3,2}}(d_{2,1}\, d_{5,1})^{r_{1,2}+r_{2,1}}(d_{2,2}\, d_{5,2})^{2r_{2,2}}\big)\\
&\qquad\qquad\cdot e\big(d_{1,3},\ (v\, d_{2,3}\, d_{5,3})^{2r_{3,3}}(d_{2,1}\, d_{5,1})^{r_{1,3}+r_{3,1}}(d_{2,2}\, d_{5,2})^{r_{2,3}+r_{3,2}}\big) =\\
&= e\big(u_{1,1},\ \phi_{1,1}^{2\cdot r_{1,1}}\phi_{1,2}^{r_{1,2}+r_{2,1}}\phi_{1,3}^{r_{1,3}+r_{3,1}}\big)\cdot e\big(u_{2,2},\ \phi_{2,1}^{r_{1,2}+r_{2,1}}\phi_{2,2}^{2\cdot r_{2,2}}\phi_{2,3}^{r_{2,3}+r_{3,2}}\big)\\
&\qquad\cdot e\big(u_{1,3},\ (\phi_{1,1}\phi_{2,1})^{r_{1,3}+r_{3,1}}(\phi_{1,2}\phi_{2,2})^{r_{2,3}+r_{3,2}}(\phi_{1,3}\phi_{2,3})^{2\cdot r_{3,3}}\big)\\
&\quad\cdot e\big(u_{3,1},\ \phi_{3,1}^{2\cdot r_{1,1}}\phi_{3,2}^{r_{1,2}+r_{2,1}}\phi_{3,3}^{r_{1,3}+r_{3,1}}\big)\cdot e\big(u_{3,2},\ \phi_{3,1}^{r_{1,2}+r_{2,1}}\phi_{3,2}^{2\cdot r_{2,2}}\phi_{3,3}^{r_{2,3}+r_{3,2}}\big)\\
&\qquad\qquad\cdot e\big(u_{3,3},\ \phi_{3,1}^{r_{1,3}+r_{3,1}}\phi_{3,2}^{r_{2,3}+r_{3,2}}\phi_{3,3}^{2\cdot r_{3,3}}\big)\cdot e(g,g)^{2r_{3,3}}\ .
\end{aligned}$$

Substituting $\vec{\mathcal{A}} = \begin{pmatrix} f \\ w^{-1}\end{pmatrix}$, $\vec{\mathbf{d}} = \begin{pmatrix}\mathbf{d}_2\\ \mathbf{d}_3\end{pmatrix}$, $t_T = 1$, and $\vec{\mathcal{A}} = \begin{pmatrix} f \\ h^{-1}\end{pmatrix}$, $\vec{\mathbf{d}} = \begin{pmatrix}\mathbf{d}_5\\ \mathbf{d}_4\end{pmatrix}$, $t_T = 1$ (respectively) in (6), we obtain the second and third equation. Once the three equations multiplied, we obtain:

$$\begin{aligned}
&e\big(d_{1,1},\ (v\, d_{2,3}\, d_{5,3})^{r_{1,3}+r_{3,1}}(d_{2,1}\, d_{5,1})^{2r_{1,1}}(d_{2,2}\, d_{5,2})^{(r_{1,2}+r_{2,1})}\big)\\
&\qquad e\big(d_{1,2},\ (v\, d_{2,3}\, d_{5,3})^{r_{2,3}+r_{3,2}}(d_{2,1}\, d_{5,1})^{r_{1,2}+r_{2,1}}(d_{2,2}\, d_{5,2})^{2r_{2,2}}\big)\\
&\qquad e\big(d_{1,3},\ (v\, d_{2,3}\, d_{5,3})^{2r_{3,3}}(d_{2,1}\, d_{5,1})^{r_{1,3}+r_{3,1}}(d_{2,2}\, d_{5,2})^{r_{2,3}+r_{3,2}}\big)\\
&\quad e(f, d_{2,1}^{s_1}d_{2,2}^{s_2}d_{2,3}^{s_3}d_{5,1}^{t_1}d_{5,2}^{t_2}d_{5,3}^{t_3})e(w^{-1}, d_{3,1}^{s_1}d_{3,2}^{s_2}d_{3,3}^{s_3})e(h^{-1}, d_{4,1}^{t_1}d_{4,2}^{t_2}d_{4,3}^{t_3})\\
&= e\big(u_{1,3},\ (\phi_{1,1}\phi_{2,1})^{r_{1,3}+r_{3,1}}(\phi_{1,2}\phi_{2,2})^{r_{2,3}+r_{3,2}}(\phi_{1,3}\phi_{2,3})^{2\cdot r_{3,3}}(\psi_1\psi_2)^{s_3}(\theta_1\theta_2)^{t_3}\big)\\
&\cdot e\big(u_{1,1},\ \phi_{1,1}^{2\cdot r_{1,1}}\phi_{1,2}^{r_{1,2}+r_{2,1}}\phi_{1,3}^{r_{1,3}+r_{3,1}}\psi_1^{s_1}\theta_1^{t_1}\big)\cdot e\big(u_{2,2},\ \phi_{2,1}^{r_{1,2}+r_{2,1}}\phi_{2,2}^{2\cdot r_{2,2}}\phi_{2,3}^{r_{2,3}+r_{3,2}}\psi_2^{s_2}\theta_2^{t_2}\big)\\
&\cdot e\big(u_{3,1},\ \phi_{3,1}^{2\cdot r_{1,1}}\phi_{3,2}^{r_{1,2}+r_{2,1}}\phi_{3,3}^{r_{1,3}+r_{3,1}}\psi_3^{s_1}\theta_3^{t_1}\big)\cdot e\big(u_{3,2},\ \phi_{3,1}^{r_{1,2}+r_{2,1}}\phi_{3,2}^{2\cdot r_{2,2}}\phi_{3,3}^{r_{2,3}+r_{3,2}}\psi_3^{s_2}\theta_3^{t_2}\big)\\
&\qquad\qquad e\big(u_{3,3},\ \phi_{3,1}^{r_{1,3}+r_{3,1}}\phi_{3,2}^{r_{2,3}+r_{3,2}}\phi_{3,3}^{2\cdot r_{3,3}}\psi_3^{s_3}\theta_3^{t_3}\big)e(g,g)^{2r_{3,3}}
\end{aligned}$$

In [BCKL08], the authors evaluated that the verification of the proof in the DLIN instantiation requires the computation of 126 pairings. With our result, we prove it can be reduced to 12.

Batching Several P-Signatures. As in the previous section, in case we want to verify multiple P-signatures at once, we can unify the pairings containing f, h and w on the left-hand side and all pairings (which are of the form $e(u_{i,j},\cdot)$) on the right-hand side. Instead of $15n$ (*resp.* $12n$) pairings needed when checking each equation, the batched version only requires $2n + 13$ (*resp.* $3n + 9$) pairings.

9 Conclusion

In this paper, we presented efficiency improvements for the verification of Groth-Sahai non-interactive zero-knowledge and witness-indistinguishable proofs and two privacy-preserving authentication schemes, saving up to 90% of the (dominant) pairing operations. These results can be combined with known methods to compute the product of many pairing evaluations efficiently [GS06]. Our results notably provide the first algorithm to batch-verify a group signature scheme in the standard model (an open problem raised in [FGHP09]) and, surprisingly, demonstrate that thanks to batch verification techniques, the DLIN instantiation of the

Groth-Sahai proof system may be the most efficient implementation for the verification of a single signature.

Acknowledgments

This work was supported by the French ANR-07-TCOM-013-04 PACE Project, by the European Commission through the IST Program under Contract ICT-2007-216646 ECRYPT II and by EADS.

References

[ACHdM05] Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385 (2005)

[BB08] Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BCC$^+$09] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)

[BCKL08] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)

[BCKL09] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Compact E-cash and simulatable vRFs revisited. In: Shacham, H. (ed.) PAIRING 2009. LNCS, vol. 5671, pp. 114–131. Springer, Heidelberg (2009)

[BFI$^+$10] Blazy, O., Fuchsbauer, G., Izabachène, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch Groth-Sahai. Cryptology ePrint Archive, Report 2010/040 (2010)

[BFM90] Blum, M., Feldman, P., Micali, S.: Proving security against chosen cyphertext attacks. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 256–268. Springer, Heidelberg (1990)

[BFPW07] Boldyreva, A., Fischlin, M., Palacio, A., Warinschi, B.: A closer look at PKI: Security and efficiency. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 458–475. Springer, Heidelberg (2007)

[BGN05] Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)

[BGR98] Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)

[BR93] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 93 Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)

[BSZ05] 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)

[BW06] Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)

[BW07] Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)

[CGH98] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th ACM STOC, pp. 209–218. ACM Press, New York (1998)

[CGS07] Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (2007)

[CHP07] Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007)

[FGHP09] Ferrara, A.L., Green, M., Hohenberger, S., Pedersen, M.Ø.: Practical short signature batch verification. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 309–324. Springer, Heidelberg (2009)

[Fia90] Fiat, A.: Batch RSA. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 175–185. Springer, Heidelberg (1990)

[FPV09] Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Transferable constant-size fair e-cash. In: Miyaji, A., Echizen, I., Okamoto, T. (eds.) CANS 2009. LNCS, vol. 5888, pp. 226–247. Springer, Heidelberg (2009)

[GL07] Groth, J., Lu, S.: A non-interactive shuffle with pairing based verifiability. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67. Springer, Heidelberg (2007)

[GPS08] Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)

[Gro07] Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)

[GS06] Granger, R., Smart, N.P.: On Computing Products of Pairings. Cryptology ePrint Archive, Report 2006/172 (2006)

[GS08] 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)

[GSW09] Ghadafi, E., Smart, N.P., Warinschi, B.: Groth–sahai proofs revisited. Cryptology ePrint Archive, Report 2009/599 (2009)

[Mat09] Matt, B.J.: Identification of multiple invalid signatures in pairing-based batched signatures. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 337–356. Springer, Heidelberg (2009)

[NMVR94] Naccache, D., M'Raïhi, D., Vaudenay, S., Raphaeli, D.: Can D.S.A. be improved? complexity trade-offs with the digital signature standard. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 77–85. Springer, Heidelberg (1995)

[PMPS00] Pastuszak, J., Michatek, D., Pieprzyk, J., Seberry, J.: Identification of bad signatures in batches. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 28–45. Springer, Heidelberg (2000)

Efficient and Secure Evaluation of Multivariate Polynomials and Applications

Matthew Franklin[1] and Payman Mohassel[2]

[1] Department of Computer Science, UC Davis
franklin@cs.ucdavis.edu
[2] Department of Computer Science, University of Calgary
pmohasse@cpsc.ucalgary.ca

Abstract. In this work, we design two-party and multiparty protocols for evaluating multivariate polynomials at participants' inputs with security against a *malicious adversary* who may corrupt all but one of the parties. Our protocols are round and communication efficient, and use the underlying cryptographic primitives in a black-box way. Our construction achieves optimal communication complexity for degree 2 and 3 polynomials.

Our constructions can be used to securely and efficiently realize a wide range of functionalities. For instance, we demonstrate how our techniques lead to efficient protocols for secure linear algebra with security against malicious adversaries. Other applications include secure evaluation of DNF/CNF formulas, and conditional secret reconstruction (or conditional oblivious transfer) for a large family of condition functions.

1 Introduction

In a secure multiparty computation (MPC) protocol, several parties each with their own private inputs collectively compute a function of their inputs without revealing additional information. Security is defined with respect to an adversary who corrupts a fraction of the parties in order to undermine the correctness of protocol and/or the privacy of honest participants. In this paper we allow the adversary to corrupt all but one of the participants (dishonest majority). A modest security consideration for MPC is to defend against a *semi-honest* adversary who follows the steps of the protocol but tries to learn more information based on the messages he receives. Semi-honest security, however, is often not sufficient and in many cases does not reflect the scenarios one encounters in real-world. A solution to this problem is to strengthen the definition by requiring security against a *malicious* adversary who can deviate from the protocol, arbitrarily. While many functions of interest are efficiently realizable in the semi-honest model, the same statement is not true in the malicious model. This motivates the following question for any function of practical importance f:

Is it possible to design an MPC protocol for realizing f with security against malicious adversaries that matches, in efficiency, the best existing constructions in the semi-honest model?

We study the above question where the functions of interest are *low-degree multivariate polynomials*, or equivalently, *constant-depth arithmetic circuits.* Many problems of interest in cryptography can be represented as such. Examples include, linear algebra computation, and conditional oblivious transfers or secret reconstruction. The efficiency criteria we consider when trying to answer the above question are:

• *Round, communication, computation.* We require that the round, computation and communication complexity (in terms of the input size) of the protocol in the malicious model matches those of the best existing constructions (for realizing f) in the semi-honest model. Note that this is the best we can hope for since malicious adversaries are strictly stronger than semi-honest ones.

• *Black-box use of the primitives.* We aim for a black-box use of the underlying cryptographic primitives. Non-black-box techniques require parties to prove in zero-knowledge, statements that involve the computation of the underlying primitives. A black box construction, on the other hand, would make the number of invocations of the primitives independent of the complexity of implementing them. Furthermore, black-box constructions can potentially be instantiated based on a variety of computational assumptions.

The existing techniques for defending against malicious adversaries fall short of giving a positive answer to the question we asked. Generic zero-knowledge compilers [10,11] for transforming a protocol that is secure in the semi-honest model into one that is secure in the malicious model, require communication complexity that is polynomial in the computational complexity of the original semi-honest protocol. The only exception in this framework is the compiler of [25] based on sublinear-communication zero-knowledge techniques such as [20], which preserves the communication complexity of the original semi-honest protocol up to a poly-logarithmic factor. This method, however, does not meet our efficiency criteria as it requires a non-black-box use of the primitives, and invokes inefficient procedures such as reductions to the circuit satisfiability problem and applications of the PCP theorem.

A different line of research has focused on our second criteria for efficiency, i.e., a black-box use of the underlying primitives. This includes the variants of Yao's garbled circuit protocol in the malicious model such as [22,13] in the two-party case and works such as [17,18,19] in the multi-party case. Due to their generic nature, however, these constructions have communication complexities that are proportional to the size of the circuit being computed which, for many functions of interest (including multivariate polynomials), is higher than the best existing protocols in the semi-honest model.

1.1 Our Results

Let P be a multivariate polynomial of degree 3 in n variables and let k be a security parameter. Note that P can have as many as n^3 terms. In the semi-honest model, the most efficient protocols for securely evaluating P at parties inputs require the communication of $O(n)$ ciphertexts between the participants.

In this paper, we design efficient two-party and multiparty protocols for the same task, with security against *malicious* adversaries:

Two-party case. In the two-party setting, we design a protocol for this task that runs in a constant number of rounds and requires the communication of $O(kn)$ ciphertexts[1] between the parties. This matches the most efficient constructions in the semi-honest model and is essentially optimal. It also significantly improves on the efficiency of the previous constructions in the malicious model (e.g. variants of Yao's protocol) which require $O(poly(k)n^3)$ communication in worst case. It is also interesting to note that while we focus on security against malicious adversaries, our protocols are automatically secure against *covert adversaries* [1] wit lower communication overhead.

We use a Reed-Solomon (RS) code with properly chosen parameters to encode inputs and use an additively homomorphic encryption scheme to send the encrypted encodings to the other party. Each character in the encoding of an input can also be interpreted as a share in a Shamir's secret sharing of the input. Parties use the homomorphic properties of the encryption scheme in order to evaluate the multivariate polynomial at each share and compute an encrypted RS encoding of the final output. Then, parties engage in a *cut-and-choose* process where a random subset of the characters in the codewords for each input and output are revealed. This allows them to verify honest computation for that subset and ensures that with high probability the number of errors in the codeword for the output is small. Thus, parties can unambiguously decode the final output. We also show how to use *share packing* techniques in order to achieve better amortized efficiency when multiple instances of the protocol are performed.

We prove our protocols secure in the stand-alone model using the *ideal-world/real-world simulation* framework. Nevertheless, many instances of our protocols can be securely run in parallel if the same challenge-verification steps are used for all of them.

On use of homomorphic encryption. As mentioned above we use a semantically secure and additively homomorphic encryption scheme as the main cryptographic primitive in our protocol. However, in addition to being additively homomorphic, we require that the Reed-Solomon encoding and decoding algorithms work properly over the domain of plaintexts defined by the cryptosystem. At first glance this seems to be a rather restrictive assumption. Consider the two widely used additively homomorphic encryption schemes in the literature, i.e. the Goldwasser-Micali (GM) encryption scheme [12] and the Paillier's encryption scheme [27]. The GM encryption works over GF(2) while we need the finite field to have at least k distinct elements for the Reed-Solomon encoding to be meaningful (where k is the security parameter). In case of Paillier's encryption scheme, the domain of plaintexts is not a finite field, and it is no longer obvious whether the Reed-Solomon encoding continues to work.

Nevertheless, we show that both schemes can be adapted to satisfy this additional requirement. First we show a simple way of extending the GM encryption

[1] Sometimes we drop the term ciphertext, but all the asymptotic complexities given in this paper refer to the number of ciphertexts communicated.

to an *additively homomorphic* encryption scheme over the extension field $\mathrm{GF}(2^s)$ for any positive integer s. Second, we prove that while the domain of plaintexts in Paillier's scheme is Z_N where N is the product of two large and secret primes, the Reed-Solomon encoding and decoding algorithms work correctly over Z_N or else the decoding algorithm can be used to factor N. We hope that the observations we make about these schemes can be of further use in other applications that use homomorphic encryption schemes.

Multiparty case. In the multi-party setting, our protocol is based on a recent compiler of Ishai et. al. [18] that combines an *outer* protocol with security against malicious adversaries (only corrupting a constant fraction of parties) with an *inner* protocol with security against semi-honest adversaries. We instantiate the inner and outer protocols with black-box constructions that meet our efficiency criteria. We design a constant round protocol with $O(poly(c, k)n)$ communication where c is the number of parties. The protocol uses the underlying primitives in a black-box way. This improves on the existing generic black-box constructions, most notably the construction of [19] for evaluating arithmetic circuits, which require $O(poly(c, k)n^3)$ communication in worst case. Due to lack of space, we omit the constructions of the multiparty protocol from this extended abstract and refer the reader to the full version for more details.

Higher degrees. Our protocols (for both two-party and multiparty setting) generalize to higher degree polynomials in a natural way. Particularly, for a degree t multivariate polynomial, we achieve black-box constructions with communication of $O(poly(k)n^{\lfloor t/2 \rfloor})$ ciphertexts (linear in k in the two-party case). While this is no longer optimal, it is more efficient that what can be achieved using existing general constructions. We leave it as an open problem to design protocols with security against malicious adversaries for evaluating multivariate polynomials of degree $t > 3$ with the optimal communication complexity.

Applications. We describe how our constructions lead to the design of efficient *secure linear algebra protocols* in the malicious model. Other applications we discuss include *communication-efficient* protocols for secure evaluation of DNF and CNF formulas, and *conditional secret reconstruction* (or conditional oblivious transfer) for a large family of condition functions. We expect that other useful cryptographic protocols can be efficiently instantiated by protocols for secure evaluation of multivariate polynomials.

1.2 Related Work

Cut-and-choose techniques have been used in works such as [23,29,22,13] to boost the security of Yao's garbled circuit protocol [30] from semi-honest adversaries to malicious adversaries. These constructions lead to protocols with communication that is proportional to the circuit size (as opposed to the input size). In contrast, our constructions apply cut-and-choose techniques to algebraic encodings of the inputs and the final output.

Representing functions with low-degree multivariate polynomials has been used by [15,16] to design round-efficient secure MPC in a different setting. The connection between secret-sharing schemes and error correcting codes has been the subject of study in other cryptography research such as [4] where it is shown how to construct linear secret sharing schemes from random error correcting codes. It is interesting to see if their techniques can be used to design more efficient mechanism for defending against malicious adversaries. A similar usage of Reed-Solomon codes and cut-and-choose techniques was recently and independently used in [5] to design efficient special-purpose secure computation protocols.

2 Preliminaries

Security Definitions. The security definitions we use to prove our protocols secure follow the ideal-world/real-world simulation paradigm. Roughly speaking, in this framework, a protocol is secure if anything that an adversary can do in the real protocol can be simulated by a simulator in an ideal world where participants send their inputs to a trusted party who performs the computation on their behalf and sends back their corresponding outputs. We do not include the detailed definitions here. Please see [9] for more detail.

Commitment Schemes. We use two types of commitment schemes in our constructions, perfectly binding (com_b) and perfectly hiding (com_h) commitments. Roughly speaking, in a perfectly binding commitment, the committed party cannot alter his commitment even if he has unbounded computational power. Similarly, a commitment scheme is perfectly hiding if an adversary that does not know the decommitment, cannot learn anything about the committed value even with unbounded computational resources. For a vector $\overrightarrow{V}$ and a set I, we abuse the notation and use $com(\overrightarrow{V})$ and $com(I)$ for element-wise commitments to each entry of the vector $\overrightarrow{V}$ and the set I respectively. We use a similar notation to commit to a permutation which in this paper is simply represented using a vector of the permuted elements.

Reed-Solomon Codes. We briefly introduce Reed-Solomon codes.

Definition 1. *Let $\Sigma = \mathbb{F}_q$ be a finite field and $\alpha_1, \ldots, \alpha_k$ be distinct elements of $\mathbb{F}_q$. Given k, d and $\mathbb{F}_q$ such that $d \leq k < q$, we define the encoding function for Reed-Solomon codes as $C : \Sigma^d \rightarrow \Sigma^k$ which on message $m = (m_0, \ldots, m_{d-1})$ outputs $C(m) = \langle p(\alpha_1), \ldots, p(\alpha_k) \rangle$ where $p(X) = \sum_{i=0}^{d-1} m_i X^i$.*

The next Lemma refers to the Welch-Berlekamp algorithm [28] for decoding Reed-Solomon codes.

Lemma 1. *If number of errors in a code word is less than or equal to $(k-d)/2$, the **Welch-Berlekamp** algorithm can unambiguously decode the message in $O(k^3)$ time.*

3 On Using Homomorphic Encryption Schemes

We use a semantically secure public-key encryption scheme that is also additively homomorphic. In particular, we call an encryption scheme E additively homomorphic if given two encryptions $E(m_1)$ and $E(m_2)$, we can efficiently compute an encryption of $m_1 + m_2$. We denote this by $E(m_1 + m_2) = E(m_1) +_h E(m_2)$. This implies that given an encryption $E(m)$ and a value c, we can efficiently compute a random encryption $E(cm)$; we denote this by $E(cm) = c \times_h E(m)$. For a vector $\overrightarrow{V}$ we denote by $E(\overrightarrow{V})$ an entry-wise encryption of the vector. We can encrypt matrices and polynomials in a similar way. We can then add two encrypted vectors (matrices/polynomials) by adding each encrypted component individually. Finally, we assume that a party who knows the randomness and plaintexts for two ciphertexts $E(m_1)$ and $E(m_2)$, can efficiently compute the randomness for $E(m_1 + m_2) = E(m_1) +_h E(m_2)$.

There are a number of homomorphic encryption schemes each with their own special properties. Our protocols work with any encryption scheme that is additively homomorphic and where the domain of plaintexts is either a *finite field* or a commutative ring where we can assume that with high probability (i) all the values we work with throughout the protocol are invertible and (ii) that the Reed-Solomon encoding/decoding algorithms work in the expected way. Next we show that two of the widely used additively homomorphic encryption schemes in the literature, namely the Goldwasser-Micali (GM) encryption scheme [12] and the Paillier's encryption scheme [27] can both be used to implement our protocols. In case of GM encryption we show a simple way of extending it to work over any extension field of GF(2) while preserving its homomorphic properties. For the Paillier's encryption which operates over a ring Z_N where N is the product of two large primes, we argue that conditioned on the fact that it is hard to factor N, we can safely assume that with high probability any element in Z_N used by our protocols is invertible, and that the Reed-Solomon encoding/decoding algorithms over Z_N work in the expected way. We hope that the observations we make here about these two encryption schemes can be of further use in other applications that use additively homomorphic encryption schemes.

3.1 Using the Goldwasser-Micali Encryption

The GM encryption scheme [12] is a semantically secure scheme based on the quadratic residuosity problem, with the additional property that it is additively homomorphic over GF(2). Unfortunately, we cannot use the GM encryption directly since for the Reed-Solomon encoding to work we need to do our computation over a finite field with at least k distinct elements where k is the security parameter. Nevertheless, we describe a simple way to extend the homomorphic properties of the GM encryption to any extension field of GF(2). We also emphasize that a similar transformation works for any other encryption scheme that is additively homomorphic over a small-size finite field. Particularly, by considering higher residuosity classes, Benaloh [2] constructs homomorphic encryption schemes over Z_p, where p is a polynomially bounded (small) prime number. Our transformation is applicable to his constructions as well.

Let $k = 2^s$, where we want to perform our computation over the extension field GF(2^s). Let $L[x]$ be an irreducible polynomial of degree s with coefficients in GF(2). We define our extension field GF(2^s) to be the set of all polynomials of degree $s-1$ with coefficients in GF(2). *Addition* is defined in the natural way to be the normal polynomial addition. In order to *multiply* two elements in GF(2^s), we first multiply the two polynomials in the normal way and then reduce the resulting polynomial mod $L[x]$.

Adding encrypted data in GF*(2^s).* In order to encrypt an element in GF(2^s) we encrypt each coefficient of the corresponding polynomial using the GM encryption. Then, it is easy to add two encrypted elements by performing the coefficient-wise addition of encrypted values over GF(2). This requires $O(s)$ *homomorphic additions* of encrypted elements in GF(2).

Multiplying encrypted data in GF*(2^s).* It is slightly more subtle to compute $E(ab)$, given a and $E(b)$ for $a, b \in$ GF(2^s). Let $A[x]$ and $B[x]$ be the polynomials of degree $s-1$ over GF(2) representing a and b in the extension field. Multiplying the publicly known polynomial $A[x]$ with the encrypted polynomial $B[x]$ is straightforward and can be performed using the homomorphic properties of GM over GF(2). This results in an encrypted polynomial of degree at most $2s-2$ over GF(2). In order to complete the multiplication operation over GF(2^s), however, we need to reduce the product polynomial modulo the publicly known irreducible polynomial $L[x]$. A simple inspection of the synthetic polynomial division reveals that this step can also be reduced to addition and multiplication operations of encrypted elements over GF(2) which can again be performed using the homomorphic properties of the GM encryption (all the multiplications are between one public and one encrypted value in GF(2)). For completeness, we describe a cleaner way to perform the modular polynomial division next.

Let $C[x] = A[x]B[x] = \sum_{j=0}^{2s-2} c_j x^j$ be the product polynomial. Our goal is to compute $C[x]$ mod $L[x]$ where $C[x]$ is encrypted and $L[x]$ is publicly known. Denote the encrypted coefficients of $C[x]$ by $E(c_{2s-2}), \cdots, E(c_0)$. Let $u_i[x] = x^{s+i}$ mod $L[x]$ for $0 \leq i \leq s-1$. Each $u_i[x]$ is a publicly computable polynomial of degree at most $s-1$ with coefficients in GF(2). Let $\alpha_0[x] = c_{s-1}x^{s-1} + \cdots + c_0$ and $\alpha_i[x] = u_i[x]$ if $c_{s+i} = 1$, and equal to the zero polynomial if $c_{s+i} = 0$, for $1 \leq i \leq s-1$. Then we have that $C[x]$ mod $L[x] = \sum_{j=0}^{s-1} \alpha_j$. First note that since $u_i[x]$'s are publicly known and we know $E(c_{s+i})$ for $1 \leq i \leq s-1$, we can non-interactively compute an encryption of $\alpha_i[x]$ for $0 \leq i \leq s-1$ using the homomorphic properties of the GM encryption. We then compute an encryption of the final result by homomorphically adding the encrypted $\alpha_i[x]$'s. It is not hard to verify that the above procedure for multiplying an encrypted element in GF(2^s) with a publicly known one requires at most $O(s^2)$ *homomorphic additions* over GF(2) using the GM encryption scheme. Also note that polynomials $u_i[x]$ are computed only once and can be reused for all future homomorphic multiplication operations.

3.2 Using the Paillier's Encryption Scheme

Another option for our protocols is to use the Paillier's cryptosystem [27]. Paillier's encryption is a semantically secure and additively homomorphic encryption scheme based on the decisional composite residuosity assumption. One issue we have to deal with when using the scheme is that the domain of Paillier's cryptosystem is the ring Z_N, where N is the product of two large and secret primes. Note that Z_N is not a finite field and it is no longer clear whether the elements that the parties encounter in the protocol are invertible or if the Reed-Solomon encoding and decoding algorithms, which are an essential part of our schemes, function properly over Z_N. Next, we argue that based on the hardness of factoring N, we can safely assume that working over Z_N will not cause any problems.

Inverting elements in Z_N. As part of the computation that takes place in our protocols, we need to invert values modulo Z_N. For example, this is the case when parties perform the Reed-Solomon decoding algorithm. Hence we need to make sure that these steps can be completed successfully. A simple observation we make is that if during the computation, we encounter a value a that is not invertible mod N, we can use a to efficiently learn a non-trivial factor of N. More specifically, $gcd(a, N)$ is a non-trivial factor of N. Hence, we can assume that all the intermediate values we encounter during the computation are invertible modulo N.

Validity of Reed-Solomon decoding. It is a bit more challenging to argue that the Reed-Solomon decoding algorithm works properly and returns the correct answer. The main problem we need to address is that we can no longer assume the crucial property that any polynomial of degree d has at most d roots.

Let $N = pq$ for primes p, q. A non-zero polynomial of degree d with coefficients in Z_N has at most d distinct roots modulo Z_p and d additional roots modulo Z_q. The same polynomial has at most d^2 distinct roots over the ring Z_N, where each root over Z_N is formed by applying the Chinese Remainder Theorem to a root over Z_p and a root over Z_q. The following lemma states that even though a polynomial of degree d over Z_N might have upto d^2 roots in Z_N, knowing more than d of them allows us to factor N.

Lemma 2. *Given a set $S \subset Z_N$ that contains more than d distinct roots of a polynomial of degree d over Z_N, we can factor N in $O(|S|^2 \text{polylog}|N|)$ time, even if the polynomial is unknown.*

Proof. Let $r_1, ..., r_{d+1} \in Z_N$ be distinct roots of a polynomial over Z_N. By the pigeonhole principle, there exists i, j such that r_i and r_j are the same root of the polynomial over Z_p but different roots over Z_q. But then the $gcd(r_i - r_j, N)$ yields a nontrivial factor of N. By computing the gcd of $s_i - s_j$ and N for every $s_i, s_j \in S$, a nontrivial factor of N is found. $\square$

We briefly mentioned the Reed-Solomon encoding procedure in section 2. The corresponding decoding algorithm is supposed to perform the following task. Given $(\alpha_i, y_i) \in Z_N \times Z_N$ for $i \in \{1, \ldots, k\}$ find a polynomial $f(x)$ of degree at most $d-1$ over Z_N such that $|E| < (k-d)/2$, for $E = \{i : y_i \neq f(\alpha_i) \bmod N\}$. We describe an algorithm that will either find such an $f(x)$, or factor N.

Theorem 1. *There exist an efficient algorithm that either returns a valid decoding of the codeword $(y_1, \ldots, y_k)$, or finds a non-trivial factor of N.*

Proof. The following is an adaptation of the standard Berlekamp-Welch decoding algorithm for Reed-Solomon encoding with the notations taken from the recent survey article of Guruswami and Rudra [14].

Set up a system of linear equations over Z_N with unknowns being the coefficients of $D_2(x)$ and $N_2(x)$ and k homogeneous constraints $Q(\alpha_i, y_i) = 0 \bmod N$, where $Q(X,Y) = D_2(X)Y - N_2(X)$ over Z_N, and where $degree(D_2) \leq (k-d)/2$, $degree(N_2) < (k+d)/2$. Here, D_2 plays the role of an error locator polynomial.

Solve the system of linear equations to recover a bivariate polynomial $Q(X,Y)$ over Z_N. Let $R(X) = Q(X, f(X))$ be an (unknown but well defined) univariate polynomial over Z_N. By construction, the degree of $R(X)$ is less than $(d+k)/2$. Moreover, $R(\alpha_i) = 0 \bmod N$ for every $i \notin E$, and there are at least $(k+d)/2$ of these.

We now have two cases. (i) If $R(X)$ is not identically zero over Z_N, then we can factor N by computing $gcd(\alpha_i - \alpha_j, N)$ for all distinct $i, j \in \{1, \ldots, k\}$. (ii) If $R(X)$ is identically zero over Z_N, then $(Y - f(X))$ is a factor of $Q(X,Y)$ over Z_N, and so we can factor $Q(X,Y)$ over Z_N to recover $f(X)$. Factoring the multivariate polynomial Q is much easier than the general task of factoring polynomials and can be performed in near linear time. $\square$

The above algorithm is guaranteed to find a valid decoding polynomial of the original codeword. But, since we are no longer computing over a finite field it is possible that the computed answer is not unique. Particularly, it might be the case that after factoring $Q(X,Y)$, there are multiple factors $(Y - f_1(X)), \ldots, (Y - f_t(X))$, where $f_i(X)$'s are valid decodings. It is not clear how to pick the "right" polynomial that contains the output we need in our protocol. Next we show that as long as no $(\alpha_i - \alpha_j)$ shares a factor with N for $i, j \in \{1, \ldots, k\}$, the valid decoding is unique and hence this will not be an issue. Note that we have control over the choice of α_i's used for the Reed-Solomon codes in our protocols and can make sure that none of the pairwise differences share a factor with N.

Theorem 2. *Let α_i for $1 \leq i \leq k$ be the k values in Z_N used for the Reed-Solomon encoding. If $gcd(\alpha_i - \alpha_j, N) = 1$ for all $i, j \in \{1, \ldots, k\}$, there is only one valid decoding.*

Proof. Lets assume that there are two polynomials of degree at most $d-1$, $f_1(x)$ and $f_2(x)$ that both are correct decodings for a specific codeword of length k, with at most $(k-d)/2$ errors. This means that $p(x) = f_1(x) - f_2(x)$ is a non-zero polynomial of degree at most $d-1$, with at least $k - 2(k-d)/2 = d$ roots in Z_N. In other words, $p(x)$ which is of degree $d-1$ has at least d roots that are among the α_i's. Based on Lemma 2 this means that one of the $(\alpha_i - \alpha_j)$'s has a non-trivial factor in common with N which contradicts our assumption. $\square$

3.3 Homomorphic Encryption Schemes That do Not Work

Elgamal encryption scheme [7], is an example of a multiplicatively homomorphic encryption scheme that cannot be used in our construction. There are more powerful homomorphic encryption schemes such as [3] for evaluating 2DNF formulas, and the recent construction of a fully-homomorphic encryption scheme by Gentry [8]. These constructions lead to communication-efficient protocols against semi-honest adversaries. Particularly, polynomials of arbitrary degree can be evaluated securely with the optimal communication of $O(n)$ ciphertext using the latter scheme. However, our techniques for defending against malicious adversaries do not seem to work for these schemes. For example, as mentioned earlier we rely on working over plaintext domains in which Reed-Solomon encoding is meaningful while neither scheme seems to satisfies this property. Moreover, the scheme of [3] can only efficiently decrypt small plaintexts which makes the decryption of random values in a large field difficult. We leave it as an open problem to extend our techniques to encryption schemes with more powerful homomorphic properties.

4 Secure Evaluation of Degree 3 Multivariate Polynomials

We start with the description of a simple two-party protocol for secure evaluation of a multivariate polynomial of degree at most 3 against *semi-honest* adversaries. In what follows, to simplify the composition we assume that the computation takes place over a finite field $\mathbb{F}$, but as discussed in detail in section 3, the computation actually takes place over the domain of plaintexts defined by the encryption scheme (given that the plaintext domain has the properties we discussed earlier). In case of the Paillier's encryption, for instance, the computation takes place over Z_N where N is the product of two large primes, while in case of extended GM encryption, the computation takes place over GF(2^s).

4.1 A Protocol against Semi-honest Adversaries

Parties encrypt their inputs using homomorphic encryption schemes and exchange the encrypted inputs. In order to achieve the best communication efficiency possible, terms of the polynomial being evaluated are split between the two parties based on who owns the majority of the variables in that term[2]. Each party computes the encryption of sum of all terms assigned to him using the homomorphic properties of the encryption. Parties then combine their results to compute the final output. The detailed description follows:

Protocol SEMI-HONEST SECPOLY3
Alice's Inputs: $x_1, \ldots, x_n \in \mathbb{F}$
Bob's Inputs: $y_1, \ldots, y_m \in \mathbb{F}$

[2] We use this monomial assignment strategy repeatedly in the paper. Similar ideas have been used in the context of communication complexity and private information retrieval (PIR) protocols.

Output: $P(X_1 = x_1, \ldots, X_n = x_n, Y_1 = y_1, \ldots, Y_m = y_m)$, where P is a publicly known multivariate polynomial of degree at most 3.

1. **Key generation step:** Alice chooses a key-pair $(pk_a, sk_a) \leftarrow G(1^n)$ for a homomorphic encryption scheme E_a. Bob does the same, for an encryption scheme E_b.
2. **Alice sends her encrypted inputs to Bob:** For every x_i, Alice sends $E_a(x_i)$ to Bob. She also generates a random input r_a and sends its encryption $E_a(r_a)$ to Bob.
3. **Bob sends his encrypted inputs to Alice:** For every y_i, Bob sends $E_b(y_i)$ to Alice. He also generates a random value r_b and sends its encryption $E_b(r_b)$ to Alice.
4. **Alice performs her portion of computation:** Alice performs this task in two steps:
   (a) For every term of the form $pX_rX_sX_q$ in polynomial P, Alice computes the value $px_rx_sx_q$. She adds all such values to get a_1. Terms of the form pX_rX_s, pX_r and p are also computed as special cases of this step where some of the variables are set to 1.
   (b) For every term of the form $pX_rX_sY_q$ in polynomial P, Alice computes the ciphertext $px_rx_s \times_h E_b(y_q)$. She then adds all the encrypted values using homomorphic properties of E_b to get the encrypted value $E_b(a_2)$. Terms of the form pX_rY_s and pY_s are also computed as special cases of this step.
   (c) Alice sends $E_b(a = a_1 + a_2 + r_a)$ to Bob.
5. **Bob performs his portion of computation:** Bob follows a similar process to compute $E_a(b_1)$ and b_2 corresponding to the terms of the form $pY_rY_sX_q$ and $pY_rY_sY_q$ respectively. He sends $E_a(b = b_1 + b_2 + r_b)$ to Alice. Note that terms of the form pX_rY_s are already computed by Alice and therefore will not be recomputed here again.
6. **Parties combine and output their results:**
   (a) Alice decrypts the ciphertext she receives to get the value b. She then computes $E_b(a + b - r_a)$ using the homomorphic properties of E_b, and sends the result to Bob. Bob decrypts the result and outputs $a + b - r_a - r_b$.
   (b) Bob computes and sends $E_a(a+b-r_b)$ to Alice in a similar way. Alice decrypts the result and outputs $a + b - r_a - r_b$.

We skip the proof of security for this above construction since we shortly give a security proof for the more complicated case of malicious adversaries. The protocol runs in constant number of rounds and requires the communication of $O(n + m)$ ciphertexts between the two parties. The computational cost is dominated by $O(n+m)$ encryptions, and $O(n+m)$ homomorphic multiplications (denoted $\times_h$). When measuring the computation complexity in this paper we ignore the multiplication and addition operations when both operands are public since the cost of these operations is dominated by operations on encrypted data.

4.2 Defending against Malicious Adversaries

We modify and enhance the protocol of previous section in order to defend against malicious adversaries. Particularly the new protocol runs in a constant number of rounds and requires the communication of $O(k(n + m))$ ciphertexts between the two parties where k is a security parameter. Since the protocol is a bit technical and long, we divide its description into four different phases and explain the intuition behind each phase as we go along: (i) Input exchange (ii) Computation (iii) Challenge-verification and (iv) Output retrieval.

Input Exchange Phase. First Alice and Bob setup two encryption schemes E_a and E_b respectively (step 1). For every input value, each party generates a random polynomial of degree $d = O(k)$ with the only restriction that the input value is the constant term of the polynomial. Parties then encrypt the coefficients of these polynomials using their own encryption schemes and send the encrypted results to each other (steps 2-5). Parties repeat a similar process for a random input, which will be used later to blind the intermediate computation results (similar to the semi-honest case). After receiving the encrypted polynomials, each party uses the homomorphic properties of the encryption scheme to compute an encrypted Reed-Solomon encoding of the other parties input values (steps 6, 7). Each character in the encoding of an input can also be interpreted as a share in a secret sharing of the input (Shamir's secret sharing in this case).

1. **Key generation step:** Alice chooses a key-pair $(pk_a, sk_a) \leftarrow G(1^n)$ for a homomorphic encryption scheme E_a. Bob does the same, for an encryption scheme E_b. We assume that it is possible to verify whether the public key is in a valid range or not. If this is not the case a zero-knowledge proof of this fact must be added to the protocol. Note that it is possible to achieve such a proof through a similar cut-and-choose procedure described next and therefore preserve the black-box use of the encryption scheme. To simplify the presentation of the protocol, however, we eliminate this step.
2. **Alice sends encrypted encoding of her inputs to Bob:** For every x_i, Alice generates random values $a_i^1, \ldots, a_i^d \in \mathbb{F}$ and lets $A_i(z) = a_i^d z^d + ... + a_i^1 z + x_i$. She then computes and sends $E_a(x_i)$ and $E_a(a_i^j)$ for $1 \leq j \leq d$ and $1 \leq i \leq n$ to Bob. The choice of parameter d is crucial for security of the protocol. It turns out that $d = k/5$ is enough, where k is a security parameter.
3. **Alice sends encrypted encoding of a random field element to Bob:** Alice generates a random field element r_a and sends an encrypted encoding of it to Bob similar to previous step. Denote the corresponding polynomial by $R_a(z)$.
4. **Bob sends encrypted encoding of his inputs to Alice:** For every y_i, Bob generates random values $b_i^1, \ldots, b_i^d \in \mathbb{F}$ and lets $B_i(z) = b_i^d z^d + ... + b_i^1 z + y_i$. He then computes and sends $E_b(y_i)$ and $E_b(b_i^j)$ for $1 \leq j \leq d$ and $1 \leq i \leq m$ to Alice.
5. **Bob sends encrypted encoding of a random field element to Alice:** Bob generates a random field element r_b and sends an encrypted encoding of it to Alice. Denote the corresponding polynomial by $R_b(z)$.
6. **Alice computes encrypted Reed-Solomon encoding of Bob's inputs:** Alice computes the encryptions $E_b(B_i(1)), \ldots, E_b(B_i(k))$ for $1 \leq i \leq m$, using the homomorphic properties of the encryption scheme:

$$E_b(B_i(j)) = E_b(y_i) +_h E_b(b_i^1) \times_h j +_h \ldots +_h E_b(b_i^d) \times_h j^d$$

   She computes $E_b(R_b(1)), \ldots, E_b(R_b(k))$ in a similar way.
7. **Bob computes encrypted Reed-Solomon encoding of Alice's inputs:** Bob does so similar to Alice.

Computation phase. Similar to the semi-honest protocol, the terms in the polynomial are split into two disjoint sets, each of which is assigned to one party. In other words the polynomial $P = P_a + P_b$ where Alice is assigned the terms in P_a and Bob is assigned the terms in P_b. Parties use the homomorphic properties of the encryption scheme in order to evaluate their assigned polynomial at each share of the inputs (steps 8a,8b for Alice). Each party then blinds the encrypted result with his/her random input and commits to the encrypted encoding of the result (step 8c for Alice). These are the vectors $\overrightarrow{V_a}$ and $\overrightarrow{V_b}$. The blinding is performed in order to keep the intermediate results private, since at a later step the commitments are opened. Intuitively, $\overrightarrow{V_a}$ and $\overrightarrow{V_b}$ are the Reed-Solomon encodings of the blinded output of evaluating P_a and P_b at parties inputs, respectively.

8. **Alice performs her portion of computation:**
   (a) For every term of the form $pX_rX_sX_q$ in P, Alice computes the vector

   $$\langle pA_r(1)A_s(1)A_q(1), \dots, pA_r(k)A_s(k)A_q(k)\rangle$$

   She adds all such vectors to get a final vector $\overrightarrow{V_1}$. Terms of the form pX_rX_s, pX_r and p are computed as special cases of this step where some variables are set to 1.

   (b) For every term of the form $pX_rX_sY_q$ in polynomial P, Alice computes the vector

   $$\langle pA_r(1)A_s(1) \times_h E_b(B_q(1)), \dots, pA_r(k)A_s(k) \times_h E_b(B_q(k))\rangle$$

   She then adds all such vectors using homomorphic properties of E_b to get the encrypted vector $E_b(\overrightarrow{V_2})$. Terms of the form pX_rY_s and pY_s are also computed as special cases of this step.

   (c) Denote the vector $\langle R_a(1), \dots, R_a(k)\rangle$ by $\overrightarrow{V_r^a}$. Alice computes $E_b(\overrightarrow{V_a} = \overrightarrow{V_1} + \overrightarrow{V_2} + \overrightarrow{V_r^a})$ and sends the commitment $com_b(E_b(\overrightarrow{V_a}))$ to Bob.
9. **Bob performs his portion of computation:** Bob follows a similar process to compute $E_a(\overrightarrow{V_3})$ and $\overrightarrow{V_4}$ corresponding to the terms of the form $pY_rY_sX_q$ and $pY_rY_sY_q$ respectively. He computes $E_a(\overrightarrow{V_b} = \overrightarrow{V_3} + \overrightarrow{V_4} + \overrightarrow{V_r^b})$ where $\overrightarrow{V_r^b} = \langle R_b(1), \dots, R_b(k)\rangle$. Bob then sends the commitment $com_b(E_a(\overrightarrow{V_b}))$ to Alice.

Challenge-verification phase. Parties engage in challenge-generation steps (steps 10,12) in order to generate random subsets of size $d/2$ of $K = \{1, \dots, k\}$. The type and order of the commitments (and decomitments) in the challenge-generation steps are important for the simulation proof of security to go through. Also, while it is tempting to use the same challenge subset for both Alice and Bob, it is not clear how to construct the simulator in the proof and prove the resulting protocol secure.

After the challenge subsets are generated, each party reveals the plaintexts and randomness corresponding to shares with indices in the challenge subset, for each input and the intermediate computation results ($\overrightarrow{V_a}$ and $\overrightarrow{V_b}$). Parties verify the validity of the openings and ensure that they are consistent with the encrypted encodings of the inputs they computed in an earlier step of the protocol. Roughly speaking, this allows the parties to verify honest encoding and computation for the fraction of the shares opened, and ensures with high probability that the number of errors in the codewords for inputs and the intermediate results are small. During the next and final phase, this allows the parties to unambiguously recover the final output using a decoding algorithm for Reed-Solomon codes. It is important to note that the number of shares to be revealed should be chosen carefully for two reasons: (i) It should be *small* enough not to leak any information about the actual inputs and (ii) it should be *large* enough to guarantee that with all but negligible probability, malicious behaviors are caught and that the decoding algorithm successfully computes the final output.

10. **Parties generate a challenge set for Alice:**
    (a) Alice generates a random subset of size $d/2$ of $K = \{1, \dots, k\}$, namely I_a, and sends $com_b(I_a)$ to Bob.
    (b) Bob generates a random permutation π_b of $1, \dots, k$ and sends a commitment to it $com_h(\pi_b)$ to Alice.
    (c) Alice decommits to reveal I_a.
    (d) Bob decommits to reveal π_b. Let $I = \pi_b(I_a)$ where $I = \{i_1, \dots, i_{d/2}\}$.
11. **Bob verifies validity of Alice's inputs:** For $1 \leq e \leq n$, and $1 \leq e' \leq d/2$, Alice reveals the plaintexts and randomness for $E_a(A_e(i_{e'}))$ and $E_a(R_a(i_{e'}))$. Bob verifies that all the opened encryptions are valid. He will abort the protocol if this is not the case.
12. **Parties generate a challenge set for Bob.** This is similar to Alice's challenge generation step.
13. **Alice verifies validity of Bob's inputs.** This is similar to Bob's verification of Alice's input.
14. **Bob verifies validity of Alice's computation:** Alice opens the commitment to $E_b(\overrightarrow{V_a})$. Bob verifies that $E_b(\overrightarrow{V_a}[i_{e'}])$ was computed correctly in step 8 for $1 \leq e' \leq d/2$. Note that in step 11 of the protocol, Alice has already revealed the shares of her input corresponding to indices $i_{e'}$ for $1 \leq e' \leq d/2$. Hence, in order to perform the verification, Bob can simply redo Alice's computation using the revealed values and confirm that the answer is the same as what Alice decommitted to. Bob will abort if this is not the case, or if Alice fails to open the commitment.
15. **Alice verifies correctness of Bob's computation.** This is similar to Bob's verification of Alice's computation.

Output retrieval phase. In this phase, parties combine their results by adding their intermediate encrypted results $\overrightarrow{V_a}$ and $\overrightarrow{V_b}$, and subtracting the random values used for blinding. Moreover, in order to ensure that the addition and subtraction are performed honestly, parties engage in a second challenge-verification phase which is very similar to the one performed previously.

16. **Alice receives her output:**
    (a) Bob decrypts the ciphertext he received in step 14 to get $\overrightarrow{V_a}$. For the ciphertexts he is not able to decrypt (in case they are invalid), he replaces those components of $\overrightarrow{V_a}$ with random values in the field. He then computes $E_a(\overrightarrow{V_a} + \overrightarrow{V_b} - \overrightarrow{V_r^b})$ using the homomorphic properties of the encryption scheme and sends the result to Alice.

(b) Parties generate a challenge set J' of size $d/2$ of $K - J$. This is done similarly to the previous challenge-generation steps.

(c) For $1 \leq e \leq m$, and $1 \leq e' \leq d/2$, Bob reveals the plaintexts and randomness for $E_b(B_e(j'_{e'}))$ and $E_b(R_b(j'_{e'}))$. Alice verifies that all the opened encryptions are valid. She aborts if this is not the case.

(d) For every index $j \in J \cup J'$, Bob reveals $\overrightarrow{V_a}[j]$. Note that given the revealed plaintexts and randomnesses by Bob, Alice can compute the $\overrightarrow{V_a}[j]$'s on her own and verify that they match the values revealed by Bob. Bob also makes sure that $E_a(\overrightarrow{V_b}[j] + \overrightarrow{V_a}[j] - \overrightarrow{V_r^b}[j])$ is computed honestly for every $j \in J \cup J'$. He will abort the protocol if this is not the case.

(e) **Alice decodes her output:** Alice decrypts the encrypted vector she received in step 16a to get the vector $\overrightarrow{V}_{final} = \overrightarrow{V_a} + \overrightarrow{V_b} - \overrightarrow{V_r^b}$. Note that, $\overrightarrow{V}_{final}$ is the Reed-Solomon encoding of *output* $+ r_a$ where *output* is the final output of the protocol, i.e., polynomial P evaluated at parties' inputs. Alice uses a Reed-Solomon decoding algorithm (see Lemma 1) to unambiguously recover the final result (Note that the degree of the polynomial corresponding to the Reed-Solomon encoding of the output is $3d$).

17. **Bob receives his output:** This is almost identical to Alice's strategy.

Theorem 3. *Given that E_a and E_b are semantically secure, com_h is perfectly hiding, and com_b is perfectly binding, the above protocol is secure against malicious adversaries. The protocol runs in a constant number of rounds, requires the communication of $O(k(m+n))$ ciphertexts, and uses the underlying primitives in a black-box way, where k is the security parameter.*

A detailed proof of security and a more concrete measurement of efficiency is given in the full version of the paper. Letting $d = k/5$ is sufficient to get the desired level of security. Given that, it is easy to verify the claimed round and communication efficiency. The computational complexity of the protocol is dominated by $O(k(m+n))$ encryptions/decryptions and the same number of homomorphic multiplications (again here we ignore operations where both operands are public).

Parallel runs of the protocol. While we only prove the protocol secure in the stand-alone model, it is worth pointing out that it is possible to run multiple instances of the protocol in parallel. The only consideration for making the simulation proof go through is to make sure that the same challenge generation steps are used for all instances.

Extension to higher degree polynomials. In the full version we show how to naturally extend this protocol to higher degree polynomials. Particularly, for degree t multivariate polynomials in n variables, our protocol requires $O(kn^{\lfloor t/2 \rfloor})$ communication.

Multiparty protocols. It is not clear how to extend our two-party protocol to the multiparty case while preserving the important features such as round and communication efficiency and/or the black-box use of the underlying primitives.

In the full version, we show a different construction based on a recent construction by Ishai et. al. [18] to design secure multiparty protocols for evaluating multivariate polynomials. The protocol runs in constant round and requires $O(\text{poly}(c,k)n^{\lfloor t/2 \rfloor})$ communication for degree t multivariate polynomials where c is the number of parties.

5 Better Amortized Efficiency

Until now, we have focused on the efficiency of a single run of the protocol for evaluating multivariate polynomials. In this section we describe how to securely evaluate a polynomial at multiple inputs with amortized efficiency that is superior to the naive solutions. Consider a multivariate polynomial P of degree t in n variables. We want to design a secure two-party protocol for evaluating P at ℓ different sets of inputs. The straightforward solution is to run the protocol of previous section ℓ times, but this requires the computation and communication of $O(\ell k n^{\lfloor t/2 \rfloor})$ ciphertexts. We show how to improve on this by designing a protocol that requires the computation and communication of $O((k+\ell)n^{\lfloor t/2 \rfloor})$ ciphertexts between the parties. One immediate application of this improvement is *better efficiency for secure linear algebra protocols* as described in the application section.

We take advantage of *share packing* techniques (originally introduced by Franklin and Yung [6]) in order to encode a collection of inputs at the cost of (almost) sharing one input. In a share packing scheme, a polynomial is used to encode many secrets as opposed to only one. For example, as a generalization of Shamir's secret sharing scheme, one can share s secrets $a_1, \ldots a_s$ by generating a random polynomial S of certain degree with the restriction that $S(i-1) = a_i$ for $1 \leq i \leq s$.

Recall that in our two-party malicious protocol, each input to the protocol is encoded using a polynomial of degree d. When evaluating P on ℓ different inputs, we encode every vector of ℓ values (where each component of the vector belongs to one instance of the protocol) using a single polynomial of degree $\ell + d$. Given such an encoding for the inputs, the rest of the protocol proceeds similar to the original protocol, with a few differences in the choice of parameters. At the end of the protocol, the codeword corresponding to the final output contains ℓ useful values, namely the results of evaluating P at each of the ℓ inputs. Parties can retrieve the ℓ outputs by decoding the codeword for the final output. A detailed description of the protocol is given in the full version. The following theorem summarizes our amortized improvement.

Theorem 4. *Given a polynomial P of degree t in n variables, there exists a secure two-party protocol for evaluating P at ℓ different input sets in constant round and with communication of $O((k+\ell)n^{\lfloor t/2 \rfloor})$ ciphertexts. The protocol is secure against malicious adversaries and uses as semantically secure homomorphic encryption scheme in a black-box way.*

6 Applications

Our constructions lead to more efficient protocols for a wide range of functionalities such as linear algebra problems, evaluation of CNF and DNF formulas and conditional oblivious transfers. Due to lack of space, we only discuss the secure linear algebra protocols here and refer the reader to the full version of the paper for more details.

6.1 Secure Linear Algebra

Efficiency and security of linear algebra protocols have been studied in a series of previous works [26,21,24,19]. Mohassel and Weinreb [24] design protocols with almost optimal efficiency and security against covert adversaries [1]. Ishai et al. [19] extend these results by providing security against malicious adversaries. Our techniques lead to protocols for linear algebra that improve on the efficiency all previous constructions while providing security against malicious adversaries.

In [24], the authors design secure two-party protocols for different linear algebra problems by reducing the task to the design of a secure matrix product protocol. Particularly, important linear algebra functionalities are securely computed using a protocol that runs in $O(s)$ rounds and performs $O(sn^{1/s})$ secure matrix product protocols where the matrices are $n \times n$. First, note that each matrix multiplication can be interpreted as a protocol for evaluating a collection of multivariate polynomials of degree at most 2 on n^2 variables. Our construction leads to a constant round protocol for this task with $O(kn^2)$ communication. Second, to perform $O(sn^{1/s})$ secure matrix multiplications, we can use the amortization techniques of section 5. This amounts to a protocol which requires the communication and computation of $O((k+sn^{1/s})n^2 = sn^{2+1/s}+kn^2)$ ciphertexts and results in a reduction in the dependency of the complexity on the security parameter k compared to the most efficient previous works which require the communication and computation of $O(poly(k)sn^{2+1/s})$ ciphertexts.

Theorem 5. *There exist secure two-party protocols against malicious adversaries, for testing singularity, computing the rank and determinant, and solving a linear system of equation for shared $n \times n$ matrices. For any positive constant s, the protocol runs in constant number of rounds and requires the communication of $O(sn^{2+1/s} + kn^2)$ ciphertexts between the parties. The protocol is secure, based on the existence of a semantically secure homomorphic encryption scheme.*

Note that our construction is in general more efficient than the recent construction of [19] since the security parameter k is only multiplied with an n^2 term in the asymptotic complexity and only additively effects the larger term in the complexity, namely $sn^{2+1/s}$. Hence, for many reasonable choices of s, our protocol is more communication-efficient than all previous constructions with security against malicious adversaries.

References

1. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 137–156. Springer, Heidelberg (2007)
2. Benaloh, J.: Verifiable secret-ballot elections. Yale University, New Haven (1987)
3. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
4. Chen, H., Cramer, R., Goldwasser, S., de Haan, R., Vaikuntanathan, V.: Secure computation from random error correcting codes. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 291–310. Springer, Heidelberg (2007)
5. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient Robust Private Set Intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, p. 142. Springer, Heidelberg (2009)
6. Franklin, M.K., Yung, M.: Communication complexity of secure computation. In: Proc. of the 24th ACM Symp. on the Theory of Computing, pp. 699–710 (1992)
7. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
8. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
9. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. II. Cambridge University Press, Cambridge (2004)
10. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proc. of the 19th ACM Symp. on the Theory of Computing, pp. 218–229 (1987)
11. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. JACM, 691–729 (1991)
12. Goldwasser, S., Micali, S.: Probabilistic encryption & how to play mental poker keeping secret all partial information. In: STOC '82, pp. 365–377 (1982)
13. Goyal, V., Mohassel, P., Smith, A.: Efficient two party and multi party computation against covert adversaries. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 289–306. Springer, Heidelberg (2008)
14. Guruswami, V., Rudra, A.: Error Correction Up to the Information-Theoretic Limit. Communications of the ACM 52(3), 87–95 (2009)
15. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: Proc. of the 41st IEEE Symp. on Foundations of Computer Science, pp. 294–304 (2000)
16. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002)
17. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: STOC, pp. 99–108 (2006)
18. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
19. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 294–314. Springer, Heidelberg (2009)

20. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proceedings of STOCS, pp. 723–732 (1992)
21. Kiltz, E., Mohassel, P., Weinreb, E., Franklin, M.: Secure linear algebra using linearly recurrent sequences. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, p. 291. Springer, Heidelberg (2007)
22. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
23. Mohassel, P., Franklin, M.K.: Efficiency tradeoffs for malicious two-party computation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 458–473. Springer, Heidelberg (2006)
24. Mohassel, P., Weinreb, E.: Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 481–496. Springer, Heidelberg (2008)
25. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: Proc. of the 33rd ACM Symp. on the Theory of Computing (2001)
26. Nissim, K., Weinreb, E.: Communication efficient secure linear algebra. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 522–541. Springer, Heidelberg (2006)
27. 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)
28. Welch, L.R., Berlekamp, E.R.: Error correction for algebraic block codes. US Patent 4,633,470, December 30 (1986)
29. Woodruff, D.P.: Revisiting the efficiency of malicious two-party computation. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 79–96. Springer, Heidelberg (2007)
30. Yao, A.C.: How to generate and exchange secrets. In: Proc. of the 27th IEEE Symp. on Foundations of Computer Science, pp. 162–167 (1986)

Efficient Implementation of the Orlandi Protocol

Thomas P. Jakobsen[1], Marc X. Makkes[2], and Janus Dam Nielsen[1]

[1] The Alexandra Institute
Aabogade 34, 8200 Aarhus N
Denmark
{thomas.jakobsen,janus.nielsen}@alexandra.dk
[2] Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
Netherlands
m.x.makkes@student.tue.nl

Abstract. We present an efficient implementation of the Orlandi protocol which is the first implementation of a protocol for multiparty computation on arithmetic circuits, which is secure against up to $n-1$ static, active adversaries. An efficient implementation of an actively secure self-trust protocol enables a number of multiparty computation where one or more of the parties only trust himself. Examples includes auctions, negotiations, and online gaming. The efficiency of the implementation is largely obtained through an efficient implementation of the Paillier cryptosystem, also described in this paper.

Keywords: Secure multiparty computation, MPC, homomorphic encryption, protocols.

1 Introduction

Secure multiparty computation is a cryptographic technique allowing n parties to jointly compute the result of a function $f(x_1, x_2, ..., x_n)$ while ensuring that the input x_i of each party P_i is kept private, even with a number t of the parties acting maliciously. The only information that is allowed to be revealed is the result of the function.

In the 80s it was proved that secure multiparty computation could in fact be applied to any computable function, making it an extremely general and useful technique, at least in theory. This was first done by Yao [34] in the restricted case of two parties, but soon followed similar results for the general case of n parties [4,13]. These results were, however, mostly of theoretical interest due to the complexity of the protocols.

Since then a large number of results have been obtained using different security and adversary models, underlying network assumptions, and improvements of previously known results.

In recent years, the theory has advanced enough to allow practical implementations of secure multiparty computation. Examples of practical systems which support evaluation of general multiparty computation are the FairPlay [23], VIFF [32], ShareMind [7], and SIMAP [9] systems. However, many applications are still infeasible in practice, especially those that rely on quick response times like online auctions. Also, in order to be practical, the aforementioned systems tend to either be restrited to a limited number of parties or to loosen up the security model. Some examples of the latter could be assuming that the corrupted parties do not deviate from the protocol (the passive security model) or that at most a certain threshold t of parties gets corrupted (threshold security model). Especially the active security model have until recently been regarded as too complex for practical implementations. However, recently Lindell, Pinkas, and Smart showed that active security in the two-party case is indeed practical [22], and Damgård, Geisler, Krøigaard, and Nielsen showed that active security can be practical if less than $n/3$ parties out of n are corrupted [14].

In this paper we go one step further and document an implementation of the Orlandi protocol [28] for secure multiparty computation which is both actively secure and tolerates up to $n-1$ corruptions. We further describe our benchmarks of this implementation, compare it to benchmarks of related protocols and argue that the protocol is indeed practical.

The rest of the paper is organized as follows. Section 2 gives an introduction to the Orlandi protocol. Section 3 describes our implementation, and we introduce our benchmarks, their setup, and discuss our results in Section 4. In Section 5 we describe and discuss how we removed the main performance bottlenecks using a high-performance implementation of the Paillier cryptosystem. A description of related work is given in Section 6, and we conclude in Section 7 along with discussing future work.

2 The Orlandi Protocol

The Orlandi protocol [28] is a protocol for secure multiparty computation on arithmetic circuits, which is secure against up to $n-1$ static, active adversaries. We will introduce the protocol in this section by giving a high-level description of the protocol and a detailed account of the parts of the protocol which have been the target of our optimizations. The Orlandi protocol is based on a Verifiable Secret Sharing (VSS) scheme, secure against a dishonest majority, augmented with a protocol for generating random shared multiplicative triples, based on a homomorphic cryptosystem. The Orlandi protocol needs a group $\mathbb{G}$ of some prime order p which is specified by the generator $g \in \mathbb{G}$. The order p and the generator g are part of the public parameters. A secret $x \in \mathbb{Z}_p$ is shared in the Orlandi protocol using additive secret sharing. Every party of the computation holds a share x_i of the secret, two uniformly randomly chosen additively secret shared elements $\rho_{i,1}$ and $\rho_{i,2}$ in $\mathbb{Z}_p$ and a public commitment C. The two random elements $\rho_{i,1}$ and $\rho_{i,2}$ are needed in order to compute the commitment to the secret, and the commitment is used when reconstructing the secret to check that no party contributed a wrong share. The commitment is computed using a double trapdoor Pedersen commitment scheme [31] based on the hardness of the discrete logarithm in the group $\mathbb{G}$. The commitment C is computed as

$C = \mathrm{Com}(x, \rho_{i,1}, \rho_{i,2}) = g^x h_1^{\rho_1} h_2^{\rho_2}$ where $h_i = g^{t_i}$ for $i \in \{1, 2\}$ and t_i is a trapdoor. We denote h_1, h_2 as the public key of the commitment scheme. A share in the Orlandi protocol is a four-tuple $(\mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p \times \mathcal{C})$, consisting of the share of the secret, $x_i \in \mathbb{Z}_p$, two uniformly randomly chosen numbers $\rho_1, \rho_2 \in \mathbb{Z}_p$, and a commitment $C \in \mathcal{C}$ to the secret. We write a share of the secret x as $[x]$. The protocol is secure in the Common Reference String (CRS) Model [12], and a proof of the security is sketched in Orlandi's PhD progress report [28] under the assumption of the hardness of the discrete logarithm problem in $\mathbb{G}$, the availability of a secure broadcast protocol, and the semantic security of the homomorphic cryptosystem. The security of the protocol holds, up to the security level 2^{-s} if λ and d are chosen such that:

$$s < d\ \log_2(M) + (d + 1)\log_2(\ln(1 + \lambda)) + 2 \tag{1}$$

where M is the number of multiplicative triples needed for a given computation. We refer the reader to Orlandi's PhD progress report [28] for the intuition behind the above expression. The parameters λ and d are used in the definition of the commands below.

The protocol can be divided into two parts: a preprocessing part where multiplicative triples are generated and an online part where arithmetic expressions are evaluated. The online part provides the commands one would usually expect from a VSS scheme such as commands for sharing a given value $\texttt{Input}(x)$, reconstructing a secret $\texttt{Open}([x])$, creating a random secret $\texttt{Rand}()$, addition, subtraction, and multiplication ($\texttt{Mul}([x], [y], [a], [b], [c])$) of shared numbers. We will not explain these commands further, except for the multiplication command. We instead refer the reader to Orlandi's PhD progress report [28]. The preprocessing part is divided into a number of building blocks (*leak-tolerant multiplication*, *triple generation*, and *triple test*), which are composed into the final triple generating (*random triple generation*) functionality which produces a list of triples. We will describe online multiplication, triple generation, and random triple generation below.

Basic Multiplication. We define the multiplication of the shares $[x]$ and $[y]$ as $[z] = \texttt{Mul}([x], [y], [a], [b], [c])$ where we assume that the parties are given a random triple $([a], [b], [c])$ s.t. $c = a \cdot b$ from a honest dealer. The multiplication is realized as follows:

1. $d = \texttt{Open}([x] - [a])$ and $e = \texttt{Open}([y] - [b])$
2. $[z] = e[x] + d[y] - de + [c]$

The basic multiplication is used both as a building block in the preprocessing phase and also for performing online multiplications. This is the main reason why multiplicative triples are generated in the preprocessing, so that they can be used in online multiplications. It also indicates that one multiplication requires one triple.

The leak-tolerant multiplication of shares $[x]$ and $[y]$ is defined as $[z] = \texttt{LTMul}([x], [y], \mathcal{M})$ where $\mathcal{M} = \{([a_i], [b_i], [c_i])\}_{i \in \{1,\ldots,2d+1\}}$ is a set of multiplicative triples. Leak-tolerant multiplication is an extension of the basic multiplication with the property that if $d + 1$ triples (a_i, b_i, c_i) are uniformly random in view of the adversary then the protocol leaks no information about x, y, and $x \cdot y$.

TripleGen() generates a triple by having each party first choose random shares $[a]$ and $[b]$ including the needed randomness and the commitments. Second, each party encrypts ($\mathrm{Enc}_{\mathrm{ek}_i}(a_i)$) his share a_i using his public key ek_i and a homomorphic cryptosystem. Then he broadcasts the encrypted share, the corresponding commitment, and the commitment for b_j. The share of the product $[c] = [a] \cdot [b]$ is computed by using the homomorphic property of the received encrypted values to multiply the shares $[a_i]$ and $[b_j]$. The product is then masked with some randomness $d_{i,j}$ and sent. The share c_i is then computed by decrypting $\mathrm{Dec}_{\mathrm{sk}_i}(\gamma_{i,j})$ the product shares, adding them up and subtracting the randomness. The private key of party i is sk_i.

Triple Generation. The triple generation command TripleGen() creates a multiplicative triple which is shared among the parties. The triple generation is realized as follows:

1. Every party P_i chooses $a_i, r_{i,1}, r_{i,2} \in_R \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p$, computes $\alpha_i = \mathrm{Enc}_{\mathrm{ek}_i}(a_i)$, $A_i = \mathrm{Com}(a_i, r_{i,1}, r_{i,2})$, and broadcasts them
2. Every party P_j does:
   (a) choose $b_j, s_{j,1}, s_{j,2} \in_R \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p$, compute $B_j = \mathrm{Com}(b_j, s_{j,1}, s_{j,2})$ and broadcast B_j
   (b) Party P_j does, for every other party P_i: choose $d_{i,j} \in_R \mathbb{Z}_{p^3}$, compute and send $\gamma_{i,j} = \alpha_i^{b_j} \mathrm{Enc}_{\mathrm{ek}_i}(1;1)^{d_{i,j}}$ to P_i
3. Every party P_i does:
   (a) compute $c_i = \sum_j \mathrm{Dec}_{\mathrm{sk}_i}(\gamma_{i,j}) - \sum_j d_{i,j} \bmod p$
   (b) pick $t_{i,1}, t_{i,2} \in_R \mathbb{Z}_p \times \mathbb{Z}_p$, compute and broadcast $C_i = \mathrm{Com}(c_i, t_{i,1}, t_{i,2})$
4. Everyone computes $(A, B, C) = (\prod_i A_i, \prod_i B_i, \prod_i C_i)$
5. Every party P_i outputs:
   $([a]_i, [b]_i, [c]_i) = ((a_i, r_{i,1}, r_{i,2}, A_i), (b_i, s_{i,1}, s_{i,2}, B_i), (c_i, t_{i,1}, t_{i,2}, C_i))$

The computation inside encrypted values gives rise to the requirement that the modulus of the cryptosystem N must be much larger than the modulus of the shares and the commitment scheme p. This is not an issue in practice because the key size of a factorization based cryptosystem is usually much bigger than the order of the group of points on an elliptic curve, if the same level of security is to be obtained.

The triple test command TripleTest() creates one multiplicative triple from two. The first triple is used to check the correctness of the second triple. This removes the risk of overflow in the encrypted computation in TripleGen() with overwhelming probability. The overflow may occur due to the differrence in the modulus of the cryptosystem and the shares and the commitment scheme.

Random triple generation RandomTriple() creates a set of multiplicative triples $\mathcal{M}$ of size M, which we call the *result set*. The result set is created by first generating a larger *distillation set* $\mathcal{D}$ of triples using TripleTest(). The size of the distillation set depends on the security parameter and M. The result set is distilled from the distillation set by first choosing a uniformly random subset called the *test set*

$\mathcal{T} \subset \mathcal{D}$ of size $\lambda(2d+1)M$. The triples in the test set are checked for correctness, and if any inconsistency is detected the protocol is aborted. Second the remaining triples $\mathcal{D} \backslash \mathcal{T}$ are partitioned into M random sets of size $(2d+1)$. The result set is generated using these sets and the $\mathcal{F}_{\text{PP}}(\textbf{rand}, \ldots)$ functionality.

Random Triple Generation. The implementation of the random triple generation command `RandomTriple()` creates a set $\mathcal{M}$ of multiplicative triples of size M which is shared among the parties. The random triple generation is realized as follows:

```
1. $\mathcal{D} = \emptyset$. For $i = 1, \ldots, (1+\lambda)(2d+1)M$ do: $\mathcal{D} = \mathcal{D} \cup$ TripleTest()
2. Coin-flip a subset $\mathcal{T} \subset \mathcal{D}$ of size $\lambda(2d+1)M$
3. For all $i \in \mathcal{T}$ the parties reveal the randomness used for TripleTest()
4. Check that the randomness is consistent with the view. Check that
   $a_i, b_i < p$ and $d_{i,j} < p^3$. Abort otherwise.
5. Partition $\mathcal{D} \backslash \mathcal{T}$ in $M$ random subsets $\mathcal{D}_i$ of size (2d + 1)
6. For $i = 1, \ldots, M$ do:
   (a) $[a] = \mathcal{F}_{\text{PP}}(\textbf{rand}, \ldots)$, $[b] = \mathcal{F}_{\text{PP}}(\textbf{rand}, \ldots)$, $[r] = \mathcal{F}_{\text{PP}}(\textbf{rand}, \ldots)$
   (b) $[c] =$ LTMul($[a], [b], \mathcal{D}_i$) and Open($[c] + [r]$)
   (c) Add $([a], [b], [c])$ to $\mathcal{M}$
```

The $\mathcal{F}_{\text{PP}}(\textbf{rand}, \ldots)$ functionality used in the random triple generation creates a random share using the $\prod_{\text{comm}}$ protocol described in Chapter 4 of Orlandi's PhD progress report [28]. The difference between using the `Rand()` function and the $\mathcal{F}_{\text{PP}}$ functionality is twofold. First, they differ in the security model of the overall protocol. If one uses the `Rand()` function then the protocol provides stand-alone security [11] whereas if one uses the $\mathcal{F}_{\text{PP}}$ functionality then it is secure in the CRS model. Second, they differ in speed. The `Rand()` function is faster than the $\mathcal{F}_{\text{PP}}$ functionality because the latter generates random shares using Universal Composable commitments whereas the first does not. In the implementation we use the `Rand()` function and thus achieve stand-alone security.

In the original protocol it is assumed that the public key for the commitment scheme is provided to the parties by a trusted third party (TTP), so that the key is randomly chosen. However in a real world setting, where the parties don't trust each other, it might not be the case that there is a single TTP that all parties trust. Other ways of generating the public key might include: measuring some physical random quantity, running a coin-flip protocol, or modeling a hash function with a random oracle (e.g. the first party can choose a random string r and publish $(r, H(r))$ and everyone parses $H(r)$ as the public key). The security of the whole protocol will reflect the security of the method used to generate the public key.

3 Implementation of the Orlandi Protocol

In this section we describe how we implemented the Orlandi protocol using VIFF, the Virtual Ideal Functionality Framework [14,32]. VIFF is an open source framework implemented in Python for executing general multiparty computations. It is possible to extend VIFF with new protocols for evaluation of arithmetic circuits. Such protocols are called *runtimes* in VIFF lingo and are materialized by the `Runtime` class, which new runtimes must subclass. A share in VIFF is represented by instances of the `Share` class. A `Share` instance represents a value to be computed in the future, and one can attach callbacks which will be executed once the share gets a concrete value. A share in the Orlandi protocol is represented using the `OrlandiShare` class which extends `Share`. The concrete value held by an `OrlandiShare` forms a tuple as described in Section 2.

The Orlandi protocol is implemented as the `OrlandiRuntime`, a subclass of `Runtime` and as such overloading the usual addition, subtraction and multiplication operators. It also provides some further methods largely corresponding to the commands described in Section 2. The implementation of the various commands follows the protocol closely, except that we combine steps and/or schedule them in parallel whenever possible. An example where we combine steps is step 1, 2.a, and 2.b of `TripleGen`() where we save one broadcast operation. An example of scheduling operations in parallel is the `TripleTest`() command where two `TripleGen`() commands are scheduled in parallel with one `Open`() and a `Rand`() command.

To speed-up the computation it can be observed that in step 2.c of the `TripleGen`() function that $\mathrm{Enc}_{\mathrm{ek}i}(1;1)$ will result in g^{N+1} when encrypting with the Paillier system. Hence, γ_{ij} can be computed by using a simultaneous multi-exponentiation method as described in Section 5, i.e. $\gamma_{ij} = \alpha_i^{b_j}(g^{N+1})^{d_{ij}}$. In addition, when using homomorphic properties of the Paillier cryptosystem, step 3.a can be rewritten to $c_{ij} = \mathrm{Dec}_{\mathrm{sk}}(\prod_j \gamma_{ij} \mod N^2) - \sum_j d_{ij}$, which results in just doing one exponentiation in total instead of one per party.

The security of the Orlandi protocol is based on the assumption of the hardness of the discrete logarithm of the group used and the presence of a broadcast channel. We satisfy the hardness assumption of the discrete logarithm by computing the commitments in a group defined by an elliptic curve over the field $\mathbb{F}_p$ with prime p of 192-bits with the generator g and the public key h_1, h_2 which consists of points on the curve. We have implemented the commitment scheme as a Python C extension using the industry strength PrimeInk ECC library v. 6.4.0 [1]. The main obstacle was the conversion from integers in base 2^{15}, which is used as the internal representation of arbitrary precision integers in Python, to base 2^{32} which is the representation used by PrimeInk ECC. The broadcast channel assumed by the Orlandi protocol is implemented using an instance of the *weak-crusader* broadcast. The weak-crusader broadcast is a variant of the *crusader* broadcast [16] where we allow a malicious adversary to make some honest parties output a message while others abort. The crusader broadcast is not needed in the Orlandi case since the protocol is already vulnerable to denial of service attacks, e.g. an adversary can just refrain from sending messages at all. By relaxing the requirements on the broadcast protocol we also get a more efficient implementation since we do not need a signature scheme. The protocol consists of two rounds. In the first round the *senders* send a value to each of the receivers, who then computes a collision resistant hash of the received value, and sends it to the other receivers in the second round, who check the correctness. We generally use the broadcast protocol in the implementation for broadcasting from a set of parties to all parties, except for the share reconstruction command in the case where only some subset of the parties should learn the output.

Broadcast. $ls = \text{broadcast}(value, senders, receivers)$, where the result ls is a list of received values.

1. Each party $P_j \in senders$ sends *value* to every party $P_i \in receivers$
2. Every party P_i in *receivers* computes a collision resistant hash on the received value and sends the hash to every other party in *receivers*
3. Each party in *receivers* checks that the received hash is equal to the hash computed by the party in the previous step, and returns *value* if true, or aborts if not

4 Benchmarks

In this section we describe how we have benchmarked our implementation with various levels of optimization, and discuss the results. We have chosen to benchmark the three commands `Mult`, `TripleGen`, and `RandomTriple`, because the other commands are not much different than the commands in a standard additive secret sharing scheme. `Mult` and `TripleGen` are straightforward to benchmark since they do not depend on the security parameter. The execution of `RandomTriple` on the other hand depends on the security parameter and the needed number of triples.

The `RandomTriple` command generates a set of triples which is distilled into a smaller set that is the result of the command. The total number of triples generated is $(1+\lambda)(2d+1)M$ where M is the size of the result set, and λ and d have to satisfy Equation 1. The overhead of distilling M triples is $(1+\lambda)(2d+1)-1$, and it is clear from Equation 1 that the overhead increases as the security parameter goes up, but also that it decreases as the number of needed triples M increases. This gives two interesting dimensions along which to investigate the execution time.

It is infeasible to benchmark every possible combination of security parameter and number of triples so we chose the security parameter values 1 (covert security [2]), 16, and 21, and 5, 10, and 30 triples, because they are representative and feasible for the interval of interesting security parameters $[1, 32]$. They are feasible in the sense that they can be computed in a reasonable amount of time. We have chosen λ and d such that the overhead is minimal.

The benchmarks are created using the VIFFBench Framework [33], which automates benchmarking of VIFF protocols. The benchmarks are defined as a small program parametrized with the number of parties and the VIFF repository revision. The results are automatically stored in a relational database. We have chosen to hardwire two numbers t_1, t_2 into the implementation, in order to avoid unnecessary complexity. The numbers are used to compute the public keys (as g^{t_1}, g^{t_2}) for the commitment scheme. This breaks the security of the implementation if one is to use the implementation for practical applications.

It does, however, not influence the efficiency of the commands, because the key can be computed in a setup phase, before the preprocessing phase. We have performed three benchmarks which were executed for each of the VIFF revisions containing significant improvements to the commands. Except for random triple generation which is only benchmarked for revision 1435. The online multiplication benchmark consist of 100 multiplications run in parallel. If we only executed one multiplication we would get too close to the resolution of the system clock that it would affect the precision of our measurements. The triple generation and the random triple generation benchmarks, on the other hand, only execute one invocation of the corresponding commands, because the execution time is much longer. For each revision we have repeated each benchmark 50, 50, and 1 times for online multiplication, triple generation, and random triple generation, respectively, in order to eliminate random noise. Executing the random triple generation 50 times for each revision would be prohibitively time consuming. All the benchmarks are performed using 1024-bits key size for the Paillier cryptosystem. We do not investigate how the implementation behaves as the latency on the network changes. The benchmarks were performed by using up to 10 identical computers equipped with 1 GHz dual-core AMD Opteron 2216 processors with 2x1 Mb level 2 cache and 2 Gb RAM each. The hosts are running Red Hat Enterprise Linux 5.2 on a 64-bit x86 architecture and were connected using gigabit Ethernet with a round-trip latency of 0.104 ms. One of the machines was chosen as the coordinator, whose responsibility it was to distribute and execute the benchmarks on the needed subset of the nine other machines. VIFFBench chooses the subset randomly.

4.1 Benchmark Results

The results of the basic multiplication benchmarks are shown in Table 1 where the average execution time for one multiplication is presented for two to nine parties along with the standard deviation. We clearly see that the implementation is efficient and achieves 15.9 ms per multiplication for three parties. The figure also shows that the average execution time increases linearly as the number of parties increases which is as expected due to the broadcast. A multiplication basically consists of two `Open`() operations with execution time linear in the number of parties. The execution time for two parties is not as expected. Based on the protocol we would expect it to be faster than for three parties, but the measurements shows that it is slower than for three, four, five, six, and even seven parties. We contemplate that the cause of this anomaly is that for two parties the implementation is CPU bound and not network bound as we have observed for three parties. The standard deviations are large compared to the measurements, and indicates some variation in our measurements. However the timings are meaningful and the basic multiplication is useful in practice even if we take the standard deviation into account.

Table 2 shows the average execution time of triple generation for two, three, and nine parties and the data is visualized as a graph in Figure 1. We only show a subset of our measurements, please see the extended version of this

Table 1. The average *execution time* in ms. of selected Basic Multiplication benchmarks as function of the number of *parties*

parties	2	3	4	5	6	7	8	9
time (ms)	27.4	15.9	19.7	22.8	25.6	26.7	28.2	35.9
stdvar (ms)	0.1	3.5	4.7	6.7	7.4	6.8	8.1	8.3

paper [18] for the full set of measurements. We have benchmarked different revisions of our implementation corresponding to the various optimizations we have performed. Revision 1231 is the initial unoptimized implementation which uses the implementation of Paillier in VIFF, revision 1355 in-lined step 1, 2.a, and 2.b of `TripleGen`(), 1370 uses our efficient implementation of the Paillier cryptosystem, 1393 moves step 2.c into C, 1399 moves step 3.a into C, and 1400 is a minor technical optimization.

The performance of the final revision is below 200 ms for all but two and four parties. This is encouraging for practical uses of the protocol. Based on the definition of `TripleGen`() we would expect to see the execution time increase linearly ($\mathcal{O}(n)$) in the number of players n. This is also the case until revision 1393. One explanation is that random noise is more dominant when the measured time is small. We again see that two parties are slower than even nine parties, but it seems like the anomaly is introduced in revision 1370, where we use a more efficient implementation of Paillier. This is consistent with our earlier observation that the two party case is CPU bound and the other are network bound. It is clear from Figure 1 that the use of an efficient implementation of Paillier gives a substantial improvement of the execution time and is the main contributor to the efficiency of the Orlandi implementation. The Figure shows that rewriting step 2.c in C gives a larger performance increase than rewriting step 3.a does, which is as we would expect. Step 2.c is more computational intensive. The improvements we have done in the Python code in revision 1355 and 1400 are dwarfed by the other improvements.

Table 3 shows the average execution time of random triple generation defined in revision 1435 for two, three, and nine parties. The full set of measurements can be found in the extended version of this paper [18]. One would expect the benchmarks to show that the execution time per triple increases as the security

Table 2. The average *execution time* in ms. of triple generation as a function of *number of parties*

parties		1231	1355	1370	1393	1399	1400
2	time	3519.6	3519.6	894.6	243.8	226.5	224.2
2	stdvar	1.0	0.8	3.2	0.9	0.7	0.7
3	time	3972.7	4012.1	376.3	155.0	168.3	170.9
3	stdvar	94.8	157.4	72.1	59.2	35.9	38.2
9	time	8937.4	8849.7	846.9	237.0	188.9	188.4
9	stdvar	460.2	281.2	27.0	36.5	20.7	29.0

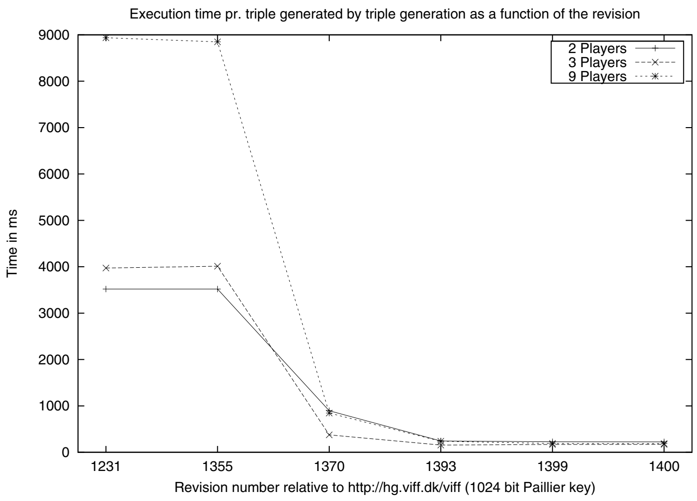


Fig. 1. The average *execution time* in ms. of selected Triple Generation benchmarks as a function of the changes to the implementation

parameter increases ($\mathcal{O}(d \log \ln(\lambda))$) and decreases as the number of triples increases. The measurements shows that the execution time increases as the security parameter increases. And, in most cases the execution time also decreases as the number of triples increases. Random noise may explain the cases where we do not see the a decrease in execution time. We have only run the benchmarks once for each combination of security parameter and number of triples due to time considerations.

4.2 Performance Comparison

We are aware of two other implementations of secure multiparty computation protocols with active security: A protocol by Damgård, Geisler, Krøigaard, and Nielsen (DGKN) which has been implemented in VIFF [14], and a protocol by Lindell, Pinkas, and Smart (LPS) [22]. The performance of these implementations cannot be directly compared to the performance of the Orlandi protocol since they rely on different security models and the benchmarks have been executed

Table 3. The average execution time in seconds of random triple generation as a function of *parties* (2, 5, and 9), *security parameter*, and *number of triples*

s	1	1	1	16	16	16	21	21	21
t	5	10	30	5	10	30	5	10	30
2	1.872	1.511	1.370	15.879	15.157	11.641	20.959	16.560	16.453
3	1.598	0.952	1.059	11.796	11.883	10.944	16.931	15.981	15.269
9	2.238	1.799	1.794	25.931	24.444	25.638	31.901	32.572	37.545

on different hardware. However, in this section we will try to elaborate on the difference between the performance of the systems.

The DGKN protocol provides evaluation of arithmetic circuits and is secure against an adaptive active adversary up to a threshold of $n/3$ corrupted parties. An adversary may halt the computation up to a synchronization point, not after - in which case termination is guaranteed. The LPS protocol is a 2-party protocol for evaluation of boolean circuits. The Orlandi protocol is a full-threshold multiparty protocol. Both the Orlandi and the LPS protocol are secure against a static active adversary, the security is based on cryptographic assumptions, and they are "unfair" in the sense that a corrupt party can prevent honest parties from getting any result while the corrupt party get results himself.

It is difficult to make a direct comparison between our results and those reported for the LPS protocol, since they do not benchmark multiplications, but rather comparisons of 16-bit integers.

Results have been reported for the DGKN protocol for 4, 7, 10, 13, 16, 19, 22, and 25 parties. If we compare the numbers for 4 and 7 parties, which is the setups we have numbers for, then DGKN takes 4 and 6 ms which is only a factor 5 (roughly) better than our results for the online case. Whereas in the preprocessing case the DGKN uses 5 ms and 22 ms in the best case, which is a factor 80-240 better then our implementation even with the smallest security parameter. Based on these numbers our implementation may seem inferior, but remember that the Orlandi protocol provides full threshold. And the benchmarks show that it is possible to use the Orlandi protocol in practice.

5 High-Performance Paillier

Various non-deterministic cryptosystems have been proposed based on randomized encryption schemes which encrypt a message m by raising a base g to the power m and suitably randomizing this result [5,15,17,26,27,29]. The security of these systems is based on the intractability of various "residuosity" problems. As an important consequence of this encryption technique, those schemes have homomorphic properties. These homomorphic properties enable computation on ciphertexts without knowing the context. This allows for a wide spread of applications such as secure multiparty computations, like Orlandi's protocol. In this section we discuss the Paillier cryptographic system and its implementation and optimization.

5.1 Description of the Paillier Schemes

Paillier has presented multiple closely related cryptosystems [29,30]. We will focus on the main- and subgroup variants of these cryptosystems shown in Figure 2 and 3, respectively. The subgroup variant is slightly different as it computes residues in a subgroup of order $\lambda(N)$.

Key Generation Let N be a RSA modulus $N = pq$, where p and q are large prime integers. Let $g \in \mathbb{Z}^*_{N^2}$ be chosen such that its order is a multiple of N. Let $\lambda(N) = \mathrm{lcm}(p-1, q-1)$. The public key is (g, N), and the private key is $\lambda(N)$

Encryption To encrypt a message $m \in \mathbb{Z}_N$, randomly chose $r \in \mathbb{Z}^*_N$ and compute the ciphertext $c = g^m r^N \mod N^2$.

Decryption The decryption of c is defined by $\frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod N^2)} \mod N$. Where the $L(\mu)$ function is defined as $\frac{\mu-1}{N}$ and takes inputs of $S_N = \{u < N^2 \mid u = 1 \mod N\}$.

Fig. 2. The main variant

5.2 Paillier Performance Evaluation

A common task in implementations of many public-key cryptosystems is multi-exponentiation in commutative groups. This is also the case for the Paillier cryptosystem, namely computing $g^m r^N \mod N^2$ for the main variant and $g^m (g^N)^r \mod N^2$ for the subgroup. Many algorithms have been proposed to speed-up the computation a single exponentiation [10,24,25,35,21]. These algorithms can be modified to compute a product over of multiple exponentiations in such a way that it is faster than a product of single exponentiations. In the following subsections we show how to reduce this overhead with different simultaneous multi-exponentiation algorithms.

The simultaneous 2^k-ary method was first introduced by Brauer [10], the idea behind the method is slicing the binary representation of an exponent into pieces using a windows of length k and processing the exponent in a larger basis. For each evaluation of the exponent the intermediate results get raised by power of 2^k and multiplied by its base raised to the power of the evaluated bits in the exponent. The powers $\{0, 1, 2, \ldots, 2^k - 1\}$ of base g are precomputed in an auxiliary table.

In order to make the 2^k-ary method evaluate two powers at the same time (i.e. $g_1^{e_1} g_2^{e_2} \mod n$). Two separate auxiliary tables with there powers of g_1 and g_2 are required. Each time both exponents get evaluated at the same time. First the intermediate result is raised to 2^k and is multiplied by each separate base raised to the power of the evaluated bits from the corresponding exponent. This saves a squaring for every bit that is evaluated.

Key Generation Let N be a RSA modulus $N = pq$, where p and q are large prime integers. Let $\lambda(N) = \mathrm{lcm}(p-1, q-1)$ and choose α such that it divides $\lambda(N)$. Let $h \in \mathbb{Z}^*_{N^2}$ such that is has maximal order of $n\lambda(N)$, and $g = h^{\lambda/\alpha} \mod N^2$. The public key is (g, N), and the private key is α

Encryption To encrypt a message $m \in \mathbb{Z}_N$, randomly chose $r \in \mathbb{Z}^*_N$ and compute the ciphertext $c = g^{m+r\cdot N} \mod N^2$.

Decryption The decryption of c is defined by $m = \frac{L(c^\alpha \mod N^2)}{L(g^\alpha \mod N^2)} \mod N$. Where the $L(\mu)$ function is defined as $\frac{\mu-1}{N}$ and takes inputs of $S_N = \{u < N^2 \mid u = 1 \mod N\}$.

Fig. 3. The subgroup variant

Algorithm 1. 2^k-ary Method

```
Require: $aux_a, aux_b, b = 2^k - 1, e_1, e_2$
Ensure: $g_1^{e_1} \cdot g_2^{e_2}$
  $A \leftarrow 1$
  for $j = \lfloor (b-1)/w \rfloor w$ down to 0 do
  $A \leftarrow A^{2^k}$
    if $(e_1[j + w - 1 \ldots j])$
      $A \leftarrow A \cdot aux_{g_1}[e_{1,j+1}, e_{1,j+2}, \ldots e_{1,j+w-1}]$
    if $(e_2[j + w - 1 \ldots j])$
      $A \leftarrow A \cdot aux_{g_2}[e_{2,j+1}, e_{2,j+2}, \ldots e_{2,j+w-1}]$
```

The simultaneous 2^k-ary matrix method is a slight modification of the simultaneous 2^k-ary method. The main difference is the computation of the auxiliary table which consists of $k \times k$ table entries which holds for $0 < i, j < 2^k - 1$ the product of $g_1^i g_2^j$ in entry $aux[i][j]$. Building such a table requires more pre-computation, but gives one less multiplication per evaluated window of length k.

The simultaneous sliding window exponentiation method of Yen, Laih, and Lenstra [35] is an improvement of the 2^k-ary method. Just like the 2^k-ary method the sliding window method consists of slicing the binary representation of e_i into pieces using a window of length ω and processing the part one by one. The addition of letting the window slide allows us to skip consecutive zeros in e_i while squaring the intermediate result. As a result, evaluation of two even exponents are avoided, and computation of the entries of these entries in the auxiliary table can be avoided. This results in a generally faster algorithm for evaluating exponents.

To evaluate two exponents simultaneously we apply the same trick as for simultaneous 2^k-ary method. But with the change that we check if both evaluated bits are zero. Additional bookkeeping is needed to keep track of the bits.

When comparing both decryption version of the Paillier scheme, the basic computation consists of one fixed exponentiation and a multiplication in $\mathbb{Z}_{N^2}^*$ and a multiplication in $\mathbb{Z}_N^*$. The subgroup variant requires the same operation, except the size of the exponent α is smaller, which makes the subgroup variant faster. Paillier has suggested an alternative decryption method by means of the Chinese Remainder Theorem (CRT). By defining $L_p = \frac{\mu - 1}{p}$ and $L_q = \frac{\mu - 1}{q}$ we can decrypt by separately computing the message modulo p and q and combining m_p and m_q with CRT. First compute $h_p = L_p(g^{p-1} \mod p^2)$ and $h_q = L_q(g^{q-1} \mod q^2)$ then $m_p = L_p(c^{p-1} \mod p^2) h_p \mod p$ and $m_q = L_q(c^{q-1} \mod q^2) h_q \mod q$ and finally recombine using CRT. Additional speed-up can be found by computing $L(\mu)$ with $\mu \cdot n^{-1} \mod 2^{|N|}$. This is just a multiplication and a logical `AND`. Another way to make computations more efficient is a careful choice of parameters. For instance if one chooses $g = 1 + n$ then the exponentiation g^m can be executed using only one multiplication, namely $g^m = (1+n)^m \equiv (1+mn) \mod n^2$. This only works for the encryption in the main variant of the Paillier scheme. Such optimizations can provide substantial speed-ups as we show in the next subsection.

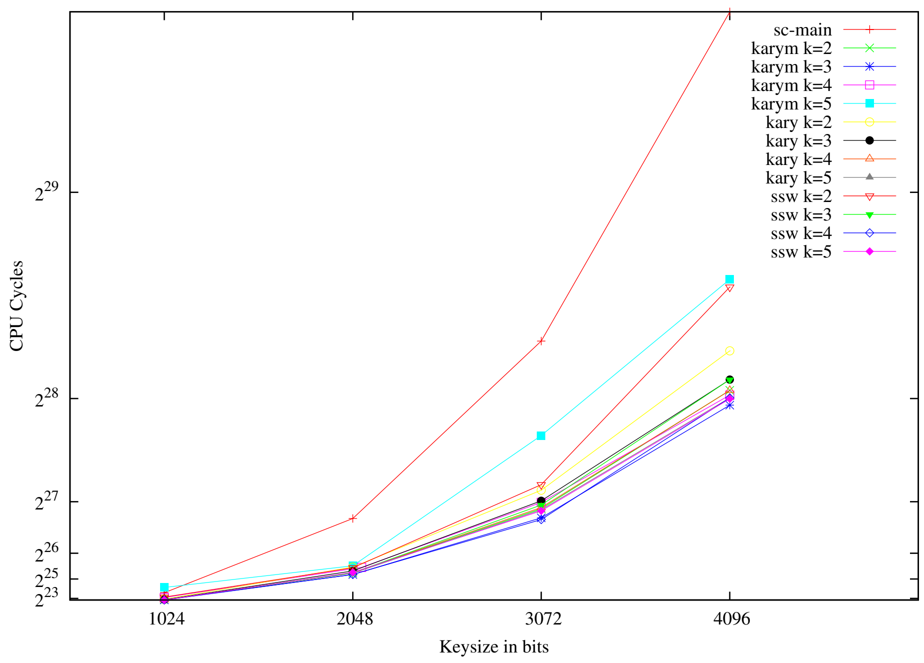


Fig. 4. The *execution time* in CPU cycles with different keys sizes with *simultaneous sliding window method* (ssw), *simultaneous* 2^k*-ary method* (kary) and *simultaneous* 2^k*-ary method* (karym) Parameter k is the size of the window in bits and the *main variant* (sc-main) with $g = n + 1$

5.3 Results

In this section we describe how we benchmarked our implementation with various optimizations and discuss the results. We have chosen to benchmark the three above simultaneous multi-exponentiation algorithms and their parameters for key sizes N ranging from 1024-bit to 4096-bit with increments of 1024-bits.

To benchmark the speed of encryption of the Paillier cryptosystem we ran the main variant with $g = N + 1$ and the subgroup variant with the 3 different simultaneous multi-exponentiation algorithms, with window size k as parameter. For windows size k we select $1 < k \leq 5$, as choosing k higher than 5 will result in longer pre-computation for these bit sizes.

The benchmarks can be found in Figure 4. The speed of the algorithms becomes clear as the key-sizes increase. As it is infeasible to test all combinations of random element r and message m, we have chosen m and r to have the same size as the sub-group length in bits this is approximately $1/4$ of N. This is due to the requirement of the Paillier cryptosystem. The benchmark includes the generation of auxiliary tables.

It is clear that the main variant of the Paillier with optimized parameters is slowest, this is mostly due to the computation of r^N with N having $\{1024, \ldots 4096\}$-bits. The simultaneous 2^k-ary normal and matrix variant as well as simultaneous sliding window method perform better with a greater k if the key length gets bigger. Also, we can see that 2^k-ary matrix performs better than 2^k-ary method as the key size grows. The 2^k-ary matrix has larger pre-computation but has one multiplication less for evaluating one bit of the exponent. The simultaneous sliding window method is in most cases the best performing algorithm, this is due to the fact that it skips consequent zeros.

The benchmarks were performed by using a 2 GHz Intel Pentium E2180 dual core with 1024KB cache per core and 2 GB Ram. The system is running Fedora release 8 with kernel 2.6.26.8-57.fc8 in 64-bit mode. For benchmarking we used *cpucycles* with is part of eBACS [6] to measure the amount of CPU cycles used by the execution.

6 Related Work

Several practical systems for general multiparty computation have been implemented during the recent years. FairPlay [23] is the earliest implementation that the authors are aware of. In the system one can specify computations in a high-level, procedural programming language. Using the FairPlay compiler, the high-level programs are then compiled to low-level representations of one-pass boolean circuits. These circuits are then used for secure computation as described by Yao [34]. The timings reported on FairPlay show that FairPlay is efficient, but it should be noted that it only supports two-party computation in the passive security model. FairPlay has later been supplemented by FairPlayMP [3] which is capable of handling the case with more than two parties in the passive security model assuming less than $n/2$ corrupted parties. A two-party protocol for secure computation which is secure against a static active adversary has recently been implemented [22]. Like the protocol used in FairPlay, this protocol is also based on boolean circuits.

Another practical system for general multiparty computation was created by Bogetoft et al. [9] in the SIMAP project. The system was used for the first known large-scale commercial application of secure multiparty computation [8]. It supports general multiparty computation in a passive threshold security model assuming less than $n/2$ corrupted parties. Like FairPlay it lets users express programs in a high level language, but contrary to FairPlay, it evaluates the programs as arithmetic rather than boolean circuits. The downside of this strategy is that comparison of integers becomes more complex and time consuming. The protocol used in the SIMAP system has also been implemented in the VIFF framework [32]. In addition, VIFF contains a passively secure two-party protocol based on the Paillier cryptosystem as well as an implementation of a multiparty protocol described in Section 4. The ShareMind system [7] represents yet another efficient approach to practical multiparty computation based on arithmetic circuits and additive sharing. It only supports three parties in the passive model and assumes that at most one party gets corrupted. None of the above implementations, though, support the combination of active security and self-trust, that is available with the implementation of the Orlandi protocol described in this paper.

7 Conclusion and Future Work

In this paper we presented an implementation of the Orlandi protocol, which is the first implementation of a MPC protocol based on arithmetic circuits, which is secure against up to $n-1$ static, active adversaries. We showed that the protocol can be implemented efficiently in the presence of an efficient implementation of a double trapdoor Petersen commitment scheme and a homomorphic cryptosystem. We also described an efficient implementation of the Paillier cryptosystem.

Practical uses is an interesting direction of future work e.g. auctions, benchmarks, and online games. Also it would be interesting to implement and benchmark the $\mathcal{F}_{\text{PP}}(\mathbf{rand}, \ldots)$ functionality and a suitable setup phase.

A further direction of future work would be to implement and benchmark the Lim/Lee [21] and the fractional window exponentiation [24] algorithms, which we expect would provide further speed up.

Acknowledgements

The authors would like to sincerely thank Ivan Damgård, Claudio Orlandi, and Jesper Buus Nielsen for answering our questions about the Orlandi protocol. In addition many thanks to Tanja Lange, Daniel J. Bernstein, and Peter Schwabe for comments and suggestions on the Paillier cryptosystem. We thank the VIFF and VIFFBench Development Teams for creating VIFF and VIFFBench, respectively. Thanks to our partners in the CACE project for making this collaboration possible. Also thanks to Cryptomatic A/S for letting us use the PrimeInk ECC library, and to the anonymous reviewers for suggestions on improving the paper.

References

1. Cryptomatic A/S. PrimeInk ECC library v. 6.4.0, `http://www.cryptomatic.com`
2. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 137–156. Springer, Heidelberg (2007)
3. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: a system for secure multi-party computation. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM Conference on Computer and Communications Security, pp. 257–266. ACM, New York (2008)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Simon, J. (ed.) [19], pp. 1–10
5. Benaloh, J.D.C.: Verifiable Secret-Ballot Elections. PhD thesis, Yale University (1978)
6. Bernstein, D.J., Lange, T.: eBACS: ECRYPT benchmarking of cryptographic systems, `http://bench.cr.yp.to`
7. 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)

8. 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.I., Toft, T.: Secure multiparty computation goes live. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009)
9. Bogetoft, P., Damgård, I., Jakobsen, T.P., 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)
10. Brauer, A.: On addition chains. Bulletin of the American Mathematical Society 45(10), 736–739 (1939)
11. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (2000)
12. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
13. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Simon, J. (ed.) [19], pp. 11–19
14. 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)
15. Damgård, I., Geisler, M., Krøigard, M.: Homomorphic encryption and secure comparison. International Journal of Applied Cryptography 1(1), 22–31 (2008)
16. Dolev, D.: The byzantine generals strike again. Technical report, Stanford University, Stanford, CA, USA (1981)
17. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
18. Jakobsen, T.P., Makkes, M.X., Nielsen, J.D.: Efficient Implementation of the Orlandi Protocol Extended Version. Cryptology ePrint Archive, Report 2010/224 (2010), http://eprint.iacr.org/
19. In: STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing, May 1988. ACM, New York (1988)
20. Lee, P.J., Lim, C.H. (eds.): ICISC 2002. LNCS, vol. 2587. Springer, Heidelberg (2003)
21. Lim, C.H., Lee, P.J.: More flexible exponentiation with precomputation. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 95–107. Springer, Heidelberg (1994)
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. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - Secure Two-Party Computation System. In: USENIX Security Symposium, pp. 287–302. USENIX (2004)
24. Möller, B.: Improved techniques for fast exponentiation. In: Lee, Lim (eds.) [20], pp. 298–312
25. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of computation 44(170), 519–521 (1985)
26. Naccache, D., Stern, J.: A new public-key cryptosystem. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 27–36. Springer, Heidelberg (1997)
27. Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)

28. Orlandi, C.: LEGO and Other Cryptographic Constructions - PhD Progress Report (March 2009), `http://www.cs.au.dk/~orlandi/`
29. 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)
30. Paillier, P., Pointcheval, D.: Efficient public-key cryptosystems provably secure against active adversaries. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 165–179. Springer, Heidelberg (1999)
31. 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)
32. VIFF - The Virtual Ideal Functionality Framework, `http://viff.dk`
33. VIFFBench Framework, `http://bitbucket.org/tpj/viffbench`
34. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: Foundations of Computer Science, pp. 162–167. IEEE, Los Alamitos (1986)
35. Yen, S.M., Laih, C.S., Lenstra, A.K.: Multi-exponentiation. Computers and Digital Techniques 141(6), 325–326 (1994)

Improving the Round Complexity of Traitor Tracing Schemes

Aggelos Kiayias[1,⋆] and Serdar Pehlivanoglu[2,⋆⋆]

[1] Computer Science and Engineering,
University of Connecticut Storrs, CT, USA
aggelos@cse.uconn.edu
[2] Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
pserdar@ntu.edu.sg

Abstract. A traitor tracing scheme is a multiuser encryption that has a built-in key leakage deterrence mechanism : the sender is capable of utilizing a tracing process that can interact with any adversarial decoder and reveal the identities of the users whose keys are employed by the decoder. A number of desired design goals have been put forth for traitor tracing schemes, notably the minimization of the length of the ciphertexts, the length of the encryption key and the storage for private keys. An important efficiency parameter that is not as widely investigated is the round complexity of the tracing process, i.e., the number of rounds of interaction that is required for the tracing process to terminate. In this work we provide (1) a general formalization of this important design consideration, (2) a novel tracing procedure that exhibits an asymptotic improvement over the previously known approaches. Our first result is achieved by casting the tracing process as a game between the tracer and the adversary where the objective of the tracer is to reveal the identity of the corrupted users while the adversary wishes to prevent that while still meeting a minimum functionality requirement. The second result involves a novel application of fingerprinting codes.

1 Introduction

The distribution of content to a set of subscribers is not served adequately by the mere employment of an encryption scheme. Indeed, in this setting — where many users are supposed to share the same decryption capability — it is plausible to expect that an adversary may take hold of some subscriber keys and spread them to other entities. In order to prevent this type of key-leakage some deterrence mechanisms can be built into the encryption scheme. In the multiuser encryption setting, the ability of an authority to interact with a rogue decoder and recover

⋆ Research partly supported by NSF Awards 0447808, 0831304, 0831306.

⋆⋆ Research is supported by The Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

at least one of the identities of the users whose keys have been leaked and utilized in the decoder gives rise to traitor tracing schemes.

These constructions were introduced in [6] and subsequently a great number of subsequent works improved various aspects of them [1,2,3,4,7,8,10,13,14,15, 17,18,19,20,21,22,23,24,26,25,27,28,29,30,31]. In this long sequence of works a number of important efficiency characteristics of traitor tracing schemes have been identified and have been stepwise refined : (i) the length of ciphertexts, (ii) the length of the encryption key, (iii) the size of storage required in the subscriber side. These quantities are typically expressed as functions of the number of users n, the failure probability of tracing ϵ and an upper bound w on the number of corrupted users.

An important aspect of traitor tracing schemes is the number of rounds of interaction that are required between the tracing authority (or simply the tracer) and a rogue device in order for the tracer to establish the desired identities. A possible reason for not pursuing this measure of optimization is that much fewer essentially different techniques to perform tracing exist compared to the large number of different constructions.

In fact the majority of schemes share the same tracing strategy that can be summarized in the following fashion : divide the users in N subsets of a certain size and perform a "walking procedure" of N steps that utilizes partially corrupted ciphertexts and identifies which one of the N subsets is overlapping with the corrupted users. Then repeat the procedure with a different set of subsets. In the end combine all the results to infer a corrupted user identity. In its most basic form, the subsets can be singletons, i.e., the users themselves and it is only needed to perform a single such walking procedure (cf. [22]).

We refer to this ubiquitous tracing strategy as "linear tracing"; it was utilized in the majority of the works cited above. Interestingly none of the previous works explicitly focused on the round complexity as an efficiency measure and on its proper formalization. In fact a proper formalization of the measure not only involves the number of steps/users n, the failure probability ϵ but also the minimum threshold σ that is required by a rogue decoder to meet when decrypting a valid ciphertext (as obviously if σ drops it becomes harder and harder to trace).

The only two known improvements of the linear tracing strategy were put forth in [26] called "binary search tracing" and "noisy binary search tracing" (the latter being an improvement inspired by [12]). These were the first asymptotic improvements of the round complexity of the linear tracing strategy as we highlight below.

The present work has two major contributions:

1. We present the first formalization of the round complexity of traitor tracing schemes. This is achieved by casting the tracing process as a protocol game between the tracer and the adversary (the rogue device). The goal of the tracer is perform identification of one of the traitor users while the goal of the adversary is to prevent this from happening while still maintaining the required functionality threshold. Of independent interest is the fact that

this formalization of tracing is the flip side of a class of privacy related interactions considered in [11] : there the adversary is in the tracer side and wishes to violate the privacy of the users.

2. We present a new tracing strategy that has an improved round complexity compared to the previous known approaches for small values of σ. Further, our strategy is the first one that can utilize an upper bound on the number of corrupted users and use that bound to curb the round complexity. Our strategy relies on a novel application of fingerprinting codes [3,33] overimposed on the tracing process. An illustration of our results is shown in figure 1.

Tracing Technique In Use	Corruption Bound	Failure Probability	Adversary Threshold σ	Round Complexity
Linear [22,6]	n	ε	$> 4n\varepsilon_p$	$O(n^3\sigma^{-2}\log\frac{1}{\varepsilon)})$
Bin. Search [26]	n	ε	$> 4n\varepsilon_p$	$O(n^2\sigma^{-2}\log\frac{\log n}{\varepsilon}\log^3 n)$
Noisy Bin. Search [26,12]	n	ε	$> \frac{2}{3}$	$O(n^2\log\frac{1}{\varepsilon}\log n)$
Our Result(1)	n	ε	$> 4n\varepsilon_p + 2\delta$	$O(n^2\delta^{-2}\log\frac{n^2\log n/\varepsilon}{\varepsilon})$
Our Result(2)	w	ε	$> 4n\varepsilon_p + 2\delta$	$O(w^2\delta^{-2}\log\frac{w^2\log n/\varepsilon}{\varepsilon})$

Fig. 1. A comparison of tracing techniques where σ is the required minimum success threshold for the adversary in decoding valid transmissions. The value ε_p corresponds to essentially the probability of building a “keyless” decoder. The value n corresponds to the number of users.

As it is evident by the table our first result matches the best previous tracing strategy of noisy binary searching for constant adversarial thresholds but are also capable of addressing smaller adversarial thresholds (e.g., $\sigma = O(\frac{1}{\log n})$ etc.) with comparable asymptotic behavior. In addition our second result improves the round complexity bound much further if one assumes that $w << n$. We note that the general approach we put forth here is fundamentally different than the one of all these previous works and it is not apparent how to utilize a bound in the number of corrupted users w in the previous strategies.

As final note, beyond [26], another previous work that identified the importance of round complexity was [16]; while the results of that paper exhibit a linear dependency on w they employ much stronger assumptions that go beyond the transmission capabilities of the majority of previous works: in particular, they assume that the content can be watermarked with an alphabet that is linear in w. This is not applicable in many settings where traitor tracing is being used (e.g., the distribution of cryptographic keys); further even in settings where watermarking is possible the alphabet size is preferably binary or a small constant independent of w. For this reason we do not include the results of [16] in the comparison above.

Paper Organization. In section 2 we present our basic primitive, the multiuser encryption scheme. In section 2.1 we put forth the simple linear length multiuser encryption scheme as our basic work paradigm. For simplicity we present our results over this scheme but they can be easily generalized to other settings where we have sequences of subsets of users etc. In section 3 we present our formalization of a tracing game that makes explicit the notion of tracing round complexity. In section 4 we present our novel tracing algorithm. Given that our formalization of the notion of tracing round complexity is new, we present first a complete analysis of the standard linear tracing strategy in our model that is then followed by our new tracing strategy based on fingerprinting codes. In the appendix we present the proofs of all our claims.

2 Multiuser Encryption Schemes

Any traitor tracing scheme is based on an underlying encryption mechanism called a multi-user encryption scheme ($\mathtt{ME}$).

A multi-user encryption scheme $\mathtt{ME}$ is a triple (**KeyDist**, **Transmit**, **Receive**). The parameter of the scheme is n, the number of receivers and is associated with three sets $\mathsf{K}, \mathsf{M}, \mathsf{C}$ corresponding to the sets of keys, plaintexts and ciphertexts respectively. We describe the I/O of these procedures below:

– **KeyDist.** It is a probabilistic algorithm that on input 1^n, it produces $(tk, ek, sk_1, \ldots, sk_n)$. The decryption key sk_i is to be assigned to the i-th user while ek is the encryption key. The tracing key tk is some auxiliary information to be used for tracing that may be empty.
– **Transmit.** It is a probabilistic algorithm that given a message $m \in \mathsf{M}$, it prepares an element $c \in \mathsf{C}$. We will write the following to denote the distribution of the output: $c \leftarrow \mathbf{Transmit}(ek, m)$
– **Receive.** It is a deterministic algorithm that on input c sampled from $\mathbf{Transmit}(ek, \langle m \rangle)$ and a user-key sk_i for some $i \in [n]$ where $(tk, ek, sk_1, \ldots, sk_n) \leftarrow \mathbf{KeyDist}(1^n)$, it either outputs m or fails. Note that **Receive** can also be generalized to be a probabilistic algorithm but we will not take advantage of this here.

We can consider a variant of the multi-user encryption scheme that is stateful where the algorithm **Transmit** is parameterized by a set of states denoted by States. In a stateful multi-user encryption scheme, the **Transmit** algorithm prepares an element $c \in \mathsf{C}$ as a function of the current state and updates the state after each transmission. In this chapter, unless stated otherwise, we will be discussing stateless multiuser encryption schemes.

The above determine the syntax of the algorithms that define a multiuser encryption scheme $\mathtt{ME}$. We expect from such a scheme to satisfy correctness in the usual sense. In particular we require that:

Definition 1. ***Correctness.*** *We say a multiuser encryption scheme* $\mathtt{ME}$ *is correct if for any* $n \in \mathbb{N}$*, for any message* $m \in \mathsf{M}$ *and for any* $u \in [n]$*, it holds that*

$$\mathbf{Prob}[\mathbf{Receive}(\mathbf{Transmit}(ek, M), sk_u) \in \{m\}] = 1$$

where $(tk, ek, sk_1, \ldots, sk_n)$ *is distributed according to* $\mathbf{KeyDist}(1^n)$.

Security. In a setting where the hybrid encryption approach is employed, the content transmission operates at two levels: first, a one-time content key k is selected and encrypted with the multiuser encryption scheme. Second, the actual message will be encrypted with the key k and will be transmitted alongside the encrypted key. It follows that a minimum requirement would be that the scheme ME should be sufficiently secure to carry a cryptographic key k. As an encryption mechanism this is known in the context of public key cryptography as a "Key Encapsulation Mechanism" [9]. The security model we present in this section will take this formalization approach, i.e., it will focus on the type of security that needs to be satisfied by a multiuser encryption scheme in order to be used as a key encapsulation mechanism. The figure 2 is the standard security game that captures the key encapsulation we require from the multi-user encryption scheme.

```
TransmitOracle($m$)
    retrieve $ek$;
    $c \leftarrow \textbf{Transmit}(ek, m)$;
    return $c$;

ReceiveOracle($c, u$)
    retrieve $sk_u$;
    return $\textbf{Receive}(c, sk_u)$;

Experiment $\textbf{Exp}_{\mathcal{A}}^{ME}(1^n)$
    $(tk, ek, sk_1, \ldots, sk_n) \leftarrow \textbf{KeyDist}(1^n)$
    $aux \leftarrow \mathcal{A}^{\mathsf{TransmitOracle}(\cdot),\mathsf{ReceiveOracle}(\cdot)}(1^n)$
    $m_0, m_1 \overset{R}{\leftarrow} \mathsf{M}$
    $b \overset{R}{\leftarrow} \{0,1\}$; $c \leftarrow \textbf{Transmit}(ek, m_1)$
    $b' \leftarrow \mathcal{A}^{\mathsf{TransmitOracle}(\cdot)}(aux, m_b, c)$
    return 1 if and only if $b = b'$
```

Fig. 2. The CCA-1 security game for multi user encryption scheme between the adversary and the challenger

Definition 2. *We say a multiuser encryption scheme* ME *is CCA-1 ε-insecure if for any probabilistic polynomial time algorithm $\mathcal{A}$, it holds that*

$$\textbf{Adv}_{\mathcal{A}}^{kem}(1^n) = \textbf{Prob}[\textbf{Exp}_{\mathcal{A}}^{kem}(1^n) = 1] - \frac{1}{2}] \leq \varepsilon$$

where the experiment $\textbf{Exp}_{\mathcal{A}}^{kem}$ is defined as in figure 2.

We note that ε in general is not supposed to be a function of n, i.e. the security property should hold independently of the number of users present in the recipient list. It is possible to define CPA version of the above security definition/game by not letting the attacker to access the ReceiveOracle on the second line of the security game. In such case we say the multiuser encryption scheme is CPA ε-insecure, if the above condition given in the definition holds for a CPA adversary.

Note that typically key encapsulation mechanisms are defined without any input beyond the encryption key (i.e., there is no plaintext part). For convenience we take a different approach where we provide the input. In effect, we state above that the encryption mechanism of the multiuser encryption matches the syntax of regular encryption and is supposed to satisfy the security requirements of a key encapsulation mechanism.

2.1 Linear Length Multiuser Encryption Scheme

We will now, present a straightforward multiuser encryption scheme that produces a ciphertext of length linear in number of receivers. We will name this scheme by $\mathtt{ME}_L$ which will be transmitting a single encrypted message to each of the receivers. It is parameterized by an encryption scheme $(\mathtt{E}, \mathtt{D})$.

– **KeyDist**$_L$: Given 1^n it produces a set of keys $\{k_1, \ldots, k_n\} \subseteq \mathsf{K}$, sk_i is set to be the key k_i for $i = 1, \ldots, n$, and sets $ek = \langle k_1, \ldots, k_n \rangle$. The tracing key tk is empty.
– **Transmit**$_L$: On input $m \in \mathsf{M}$ and the encryption key $ek = \langle k_1, \ldots, k_n \rangle$, it transmits the encryption of the message m with ek by employing the encryption scheme $(\mathtt{E}, \mathtt{D})$ as follows:

$$\langle \mathtt{E}_{k_1}(m), \ldots, \mathtt{E}_{k_n}(m) \rangle$$

– **Receive**$_L$: Given the key $sk_i = k_i$ for some $i \in [n]$ and a transmission of the form $\langle c_1, \ldots, c_n \rangle$, it returns $\mathtt{D}_{k_i}(c_i)$.

We require the broadcast encryption scheme to be capable of transmitting a cryptographic key. We will ask that this same requirement should also be satisfied by the underlying cryptographic primitive $(\mathtt{E}, \mathtt{D})$, i.e., a cryptographic key should be encapsulated safely by the underlying encryption primitive.

We formalize the security requirement as the following game: for a random choice of the key k, the adversary $\mathcal{A}$ can adaptively choose plaintexts and see how $\mathtt{E}_k$ encrypts them; similarly, is capable of observing the output of decryption procedure $\mathtt{D}_k$. The adversary is challenged with a pair (c, m) for which it holds that either $c \leftarrow \mathtt{E}_k(m)$ or $c \leftarrow \mathtt{E}_k(m')$ where m, m' are selected randomly from the message space. The goal of the adversary is to distinguish between the two cases. This models a CCA1 type of encryption security, or what is known as a security against lunch-time attacks.

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{kem}$
    Select $k$ at random.
    $aux \leftarrow \mathcal{A}^{\mathtt{E}_k(),\mathtt{D}_k()}()$
    $m_0, m_1 \overset{R}{\leftarrow} \mathsf{M}$; $b \overset{R}{\leftarrow} \{0,1\}$; $c = \mathtt{E}_k(m_1)$
    $b' \leftarrow \mathcal{A}^{\mathtt{E}_k()}(aux, c, m_b)$
    return 1 if and only if $b = b'$;
```

Fig. 3. The security game of CCA1 secure key encapsulation for an encryption scheme

Definition 3. *We say the symmetric encryption scheme* $(\mathtt{E},\mathtt{D})$ *is* ε*-insecure if it holds that for any probabilistic polynomial depth circuit* $\mathcal{A}$

$$\mathbf{Adv}^{kem}_{\mathcal{A}} = |\mathbf{Prob}[\mathbf{Exp}^{kem}_{\mathcal{A}} = 1] - \frac{1}{2}| \leq \varepsilon$$

Observe that the above requirement is weaker that one would typically expect from an encryption scheme that may be desired to protect the plaintext even if it is arbitrarily distributed. We note though that the key encapsulation security requirement will still force the encryption function to be probabilistic: indeed, in the deterministic case, the adversary can easily break security by encrypting m_b and testing the resulting ciphertext for equality to c. Further, since we are only interested in key encapsulation we can require the encryption oracle to only return encryptions of random plaintexts (as opposed to have them adaptively selected by the adversary).

Theorem 1. *A linear length multiuser encryption scheme* $\mathtt{ME}_L$ *satisfies correctness (cf. Definition 1) i.e., we assume that for all* $k, m \in \mathsf{K}, \mathsf{M}$ *it holds* $\mathtt{D}_k(\mathtt{E}_k(m)) = m$. *It is, further, CCA-1* ε*-insecure in the sense of Definition 2 with* $\varepsilon \leq 2n \cdot \varepsilon_p$ *where the underlying encryption scheme* $(\mathtt{E},\mathtt{D})$ *is* ε_p*-insecure in the sense of Definition 3.*

The obvious shortcoming of the above scheme is that the ciphertext length is linear in n. A nontrivial approach to reduce the transmission overhead involves the usage of fingerprinting codes, see [5,1,6,7,33] for a non-inclusive citation list. As the scope of this paper is not improving the transmission overhead we will not discuss such constructions. We find the description of linear length traitor tracing scheme sufficient to discuss the new formalization and to introduce the new technique in identifying a traitor. This new technique is readily applicable to all traitor tracing mechanisms involving some walking argument, i.e. querying the adversary with special transmissions that randomizes the ciphertexts one by one.

3 Tracing Game: Definitions

We define the property that is the characteristic of a traitor tracing scheme. We first introduce the concept of the tracing game.

Definition 4. *A* tracing game *is specified by any triple* $\langle \mathbf{KeyDist}, \mathcal{Q}, \mathcal{R}\rangle$*, where* **KeyDist** *is a probabilistic algorithm that on input* 1^n*, it produces a tuple* $(tk, ek, sk_1, \ldots, sk_n)$*,* $\mathcal{R}$ *is a predicate and* $\mathcal{Q}$ *is a set of random variables.*

We now explain the meaning of the tracing game. A tracing game is an interaction between two parties: the adversary and the tracer. The tracer has at its disposal the encryption and the tracing key (resp. tk, ek) while the adversary has a set of user keys, that is a subsequence of $sk_1, \ldots, sk_n$. The ultimate objective of the tracer is to identify one of the keys that the adversary controls.

Next we set the general rules of engagement between the tracer and the adversary that are determined by $\mathcal{Q}, \mathcal{R}$. The essence on the constraint that we will place on the interaction is the following: as long as the tracer follows a certain query distribution as defined in $\mathcal{Q}$ the adversary is obliged to formulate its responses in a way that they will satisfy the predicate $\mathcal{R}$ with sufficient probability.

More specifically the pair $\langle \mathcal{A}, \mathcal{T} \rangle$ will be said to be σ-*admissible* according to a tracing game $\langle \mathbf{KeyDist}, \mathcal{Q}, \mathcal{R} \rangle$ with a parameter t provided that $\mathcal{A}, \mathcal{T}$ follow the proper rules of engagement. More specifically, σ-admissible would be a pair of interacting algorithms that when $\mathcal{T}$ sends some message to $\mathcal{A}$ that follows a random variable of $\mathcal{Q}$ then $\mathcal{A}$ as to provide a response that satisfies the predicate $\mathcal{R}$. Formally we have the following definition.

Definition 5. *Let $\langle \mathcal{A}, \mathcal{T} \rangle$ be a pair of interacting algorithms (the adversary and the tracer) and let $\langle \mathbf{KeyDist}, \mathcal{Q}, \mathcal{R} \rangle$ be a tracing game. For $n \in \mathbb{N}$ and any $\mathsf{C} \subseteq [n]$ assume the following regarding the interaction of $\langle \mathcal{A}, \mathcal{T} \rangle$:*

– *Initialization. The tuple $(tk, ek, sk_1, \ldots, sk_n) \leftarrow \mathbf{KeyDist}(1^n)$ is sampled and the adversary $\mathcal{A}$ is given input $\{sk_j\}_{j \in \mathsf{C}}$ while the tracer $\mathcal{T}$ is given tk, ek.*
– *Round actions. $\mathcal{A}$ and $\mathcal{T}$ exchange messages in rounds until the tracer $\mathcal{T}$ terminates. In the i-th round $\mathcal{T}$ goes first transmitting a value q_i and then party $\mathcal{A}$ responds by a value a_i. In case $\mathcal{A}$ produces no output at a certain round i the value a_i is defined to be $\perp$.*

We say the pair $\langle \mathcal{A}, \mathcal{T} \rangle$ is σ-admissible *for the tracing game $\langle \mathbf{KeyDist}, \mathcal{Q}, \mathcal{R} \rangle$ for t-coalitions, where $t \in \mathbb{N}$ if for any $n \in \mathbb{N}, \mathsf{C} \subseteq [n]$, in case, q_i is distributed according to some member of $\mathcal{Q}$ it holds that for any $\mathsf{C} \subseteq [n]$ with $|\mathsf{C}| \leq t$,*

$$\mathbf{Prob}[\mathcal{R}(\mathsf{C}, tk, ek, sk_1, \ldots, sk_n, q_i, a_i) = 1] \geq \sigma$$

where a_i is the response of $\mathcal{A}$ to the query q_i on input $\{sk_j\}_{j \in \mathsf{C}}$. We denote the random variable that is the output of the tracer $\mathcal{T}$ after interacting with $\mathcal{A}$ by $\langle \mathcal{A}, \mathcal{T} \rangle (tk, ek, sk_1, \ldots, sk_n, \mathsf{C})$. We denote the maximum number of rounds that take place before $\mathcal{T}$ terminates the protocol by $r_{\langle \mathcal{A}, \mathcal{T} \rangle}$.

The definition of σ-admissibility in plain words it says that as long as the tracer $\mathcal{T}$ follows some of the specified valid moves in $\mathcal{Q}$ the party $\mathcal{A}$ has to oblige and satisfy with its response the predicate $\mathcal{R}$ with probability σ. We observe that the predicate $\mathcal{R}$ takes into account the total information that is available to both players and thus it is not something that the tracer $\mathcal{T}$ is necessarily capable of computing by itself. In the coming section we will formulate various tracing games that are resulting from interactions based on multiuser encryption schemes.

We will also condider the following variations of the definition:

1. *Stateful Tracing.* Consider the set $\{\mathcal{Q}_h\}_{h \in \{0,1\}^*}$, a collection of sets of random variables. In this specification, for the adversary to oblige and satisfy the $\mathcal{R}$ predicate we require the tracing queries of $\mathcal{T}$ to be consistent with

the history of previous queries. More specifically, we define, for any $i > 0$, $h = \langle q_1, \ldots, q_{i-1} \rangle$ to be the history of the queries posed by $\mathcal{T}$ (it is empty if $i = 1$), the next query q_i should be distributed according to some member of $\mathcal{Q}_h$ in order to impose the σ lower bound in the satisfaction of the predicate $\mathcal{R}$ for $\mathcal{A}$. Note that the predicate $\mathcal{R}$ will also take the history of the queries into account while producing a result. Stateful tracing is thus places a further possible restriction on the tracer's side as it drops any compliance requirements on the part of $\mathcal{A}$ when the tracer becomes inconsistent with the query history.

2. *Alfresco.* When tracing alfresco the tracer needs to form every query he makes to be statistically indistinguishable from members of $\mathcal{Q}$ (or from $\mathcal{Q}_h$ in case of stateful tracing). More specifically, when the tracer has submitted $h = \langle q_1, \ldots, q_{i-1} \rangle$ queries, in the i-th round it must choose a query that is statistically indistinguishable from a member of $\mathcal{Q}$ (from a member of $\mathcal{Q}_h$ in case of stateful tracing). Note that this is a different type of restriction on the tracer $\mathcal{T}$. Without this restriction the tracer has the flexibility to provide queries to $\mathcal{T}$ that are outside of $\mathcal{Q}$ but they may perhaps be computationally indistinguishable to some random variables of $\mathcal{Q}$ for $\mathcal{A}$; depending on the case this may carry substantial advantages for the tracer that are stripped in the case of alfresco tracing.
3. *Tracing with Resetting.* The adversary $\mathcal{A}$ is not allowed to maintain state from one round to the next, i.e., in each round the tracer can "reset" the adversary. This model weakens the adversary $\mathcal{A}$ as it is prohibited from keeping knowledge of previous queries. This can be taken advantage of by the tracer $\mathcal{T}$. Alternatively, when the adversary maintains state across rounds (i.e., the default in the definition above) we say we deal with *history-recording* adversaries.
4. *Abrupt adversaries.* This is a strengthening of the adversarial model that enables $\mathcal{A}$ to finish the game at a moment of its choosing. This means that $\mathcal{A}$ may produce a special symbol as a response upon receiving which the tracer $\mathcal{T}$ is not allowed to submit any further queries. We note that this is not in violation of the basic tenet of being admissible : $\mathcal{A}$ when it forms a σ-admissible pair with the adversary is still supposed to satisfy the $\mathcal{R}$ with probability at least σ. If $\mathcal{A}$ is abrupt though it may decide to stop the tracing game with probability as high as $1 - \sigma$ if given a query from $\mathcal{Q}$ (and in fact with even higher probability when given queries from outside of this set).

Now that we have defined the rules of engagement between the two players of the tracing we will specify when game is winnable by the tracer.

Definition 6. *We say that the tracing game* $\mathtt{TG} = \langle \mathbf{KeyDist}, \mathcal{Q}, \mathcal{R} \rangle$ *is winnable with probability* α *for* σ*-threshold and* t*-coalitions if there exists a tracer* $\mathcal{T}$ *such that for all* $\mathcal{A}$ *for which the pair* $\langle \mathcal{A}, \mathcal{T} \rangle$ *is* σ*-admissible it holds that for all* $n \in \mathbb{N}$, $\mathsf{C} \subseteq [n]$, $|\mathsf{C}| \leq t$,

$$\mathbf{Prob}[\emptyset \neq \langle \mathcal{A}, \mathcal{T} \rangle (tk, ek, sk_1, \ldots, sk_n, \mathsf{C}) \subseteq \mathsf{C}] \geq \alpha$$

where $(tk, ek, sk_1, \ldots, sk_n)$ *is distributed according to* $\mathbf{KeyDist}(1^n)$.

Provided that $\mathtt{TG}$ *is winnable with a tracer* $\mathcal{T}$*, we define the tracing round complexity* $\mathsf{roundc}[\mathtt{TG}, \mathcal{T}]$ *of the tracing game* $\mathtt{TG}$ *as the supremum of all* $r_{\langle \mathcal{A}, \mathcal{T} \rangle}$*; note that* $\mathsf{roundc}[\mathtt{TG}, \mathcal{T}]$ *is a function of* t, σ *and possibly of other parameters as well.*

It is interesting to note that the tracing game can be thought of as a privacy game if we flip the semantics of what side is adversarial. In this alternative interpretation, the good side is sitting on the adversary side and attempts to output some data that meet some "usefulness criterion" determined by $\mathcal{R}$ and are based on the private information of the users. On the other hand, the adversary is sitting on the tracer side and attempts to violate the privacy of the users. This perspective is in line with a recent work by Dwork et.al. [11] for which we do not pursue this parallel further here.

In this work, we are interested in tracing games whose **KeyDist** algorithm and the query distribution $\mathcal{Q}$ are related to multiuser encryption schemes. For an s-ary multiuser encryption scheme $\mathtt{ME} = (\mathbf{KeyDist}, \mathbf{Transmit}, \mathbf{Receive})$, the set of random variables $\mathcal{Q}$ is defined as the collection of all encryptions of plaintexts that can be transmitted, i.e. denoting it by $\mathcal{Q}^{\mathbf{Transmit}}$, it contains the random variables $\mathbf{Transmit}(ek, m)$ for any $m \in \mathsf{M}$ where $(tk, ek, sk_1, \ldots, sk_n) \leftarrow \mathbf{KeyDist}(1^n)$. In case the scheme $\mathtt{ME}$ is stateful over as a set of States, $\mathcal{Q}$ is parameterized with States as well. We note that depending on the actual use of a multiuser encryption scheme one may define the tracing game considering only specific distributions of plaintext.

With respect to the predicate $\mathcal{R}$ there are more than one ways to define it. Recall that intuitively we use this predicate to measure the success of the adversary in responding to the transmissions in a normal mode of operation as we define the query space $\mathcal{Q}$ to be sampled from **Transmit**. Indeed a σ-admissible tracer-adversary pair is one for which $\mathbf{Prob}[\mathcal{R}(\mathsf{C}, tk, ek, sk_1, \ldots, sk_n, q, a) = 1] \geq \sigma$ holds if a is the response of the adversary to the query q that is sampled as an element of $\mathcal{Q}$.

The exact choice of the way predicate $\mathcal{R}$ works will provide a classification of types for tracing games. Jumping forward we give a glimpse to possible definitions: for example, the adversary may return a key from the key space (not necessarily among the $\{sk_i\}_{i \in [n]}$) that successfully decrypts the query/transmission $q \leftarrow \mathbf{Transmit}(ek, m)$ with σ probability for any choice of $m \in \mathsf{M}$. Alternatively the predicate $\mathcal{R}$ checks to see if the response a is the plaintext transmitted in the encrypted form q. We note that $\mathcal{R}$ is capable of performing these checks in polynomial-time as it has access to the full information of keys and the adversarial input.

The exact choice of the predicate $\mathcal{R}$ and the restrictions on the queries posed by the tracer $\mathcal{T}$ yield different types of tracing. Formalizing three types of tracing games very roughly, our interest in this work is black-box traitor tracing.

Black-Box Tracing Game. In this setting, the tracer has merely black-box access to the pirate decoder. Black-box traitor tracing may in some cases allow tracing to be performed remotely without the physical availability of the pirate decoder.

The major challenge in the black-box traitor setting is to extract information regarding the original keys utilised in the construction of the pirate decoder. The tracer will communicate with the pirate decoder using a set of specially crafted queries. These queries will not be necessarily normal transmissions as the tracing center is allowed to communicate with the decoder in an arbitrary way. The response of the decoder may be equal to the decrypted plaintext, or be simply of binary form, essentially "yes", in case of returning the content in the cleartext form, or "no", in case of responding arbitrarily or jamming.

In our exposition, we will use the threshold σ to impose the adversarial constraint related to the success probability of the pirate decoder in decrypting regular transmissions. This is of particular importance, since tracing would be impossible against a pirate decoder that is not required to operate correctly at least some of the time.

Definition 7. *A multi-user encryption scheme* $\mathtt{ME} = (\mathbf{KeyDist}, \mathbf{Transmit}, \mathbf{Receive})$ *is a* black box traitor tracing scheme for t-coalitions with success probability α against σ-pirates *if the black-box tracing game* $\mathtt{TG} = \langle \mathbf{KeyDist}, \mathcal{Q}^{\mathbf{Transmit}}, \mathcal{R}^{\mathtt{BB}} \rangle$ *against t-coalitions is winnable with probability* α *against* σ*-adversaries.*

The $\mathcal{R}^{\mathtt{BB}}$ *is defined as follows:* $\mathcal{R}^{\mathtt{BB}}(\mathsf{C}, tk, ek, sk_1, \ldots, sk_n, q, a)$ *is equal to 1 if and only if* $a = m$ *whenever* $q = \mathbf{Transmit}(ek, m)$*.*

We define the tracing round complexity $\mathsf{roundc}[\mathtt{ME}]$ *of the multiuser encryption scheme as the infimum of all* $\mathsf{roundc}[\mathtt{TG}, \mathcal{T}]$ *for which* $\mathtt{ME}$ *is winnable with the tracer* $\mathcal{T}$*.*

One may also consider a more general view of the black box tracing model, that is related to the case that the pirate decoder is a tamper resistant box, such as a music player and the response of the decoder is not the exact decryption of the transmission but rather the actual rendering of the cleartext transmission on a display device. In such case, the tracer can still extract useful information by observing whether the given ciphertext results in music being played or not. It is possible to address such definitions in our framework by introducing a *filter* algorithm that restricts some of the information contained in a_i and having the tracer have access to a_i only through the filter.

Among the different variations of the tracing game described in Definition 4, tracing with resetting is relevant to black-box tracing as it is defining the capabilities of the pirate decoder the tracer has access to. We would like to motivate this case briefly in the following paragraph for the context of the present section.

A pirate decoder is said to be resettable if the tracer has the capability to reset the pirate decoder to its initial state and the decoder is available for a new query. This gives the tracer the advantage of asking queries that will be handled independently during the tracing process, i.e., effectively preventing the decoder from using previous querying information submitted by the tracer in order to decide its present action. In contrast, a *history recording* pirate decoder "remembers" the previous queries made by the tracer and because the tracing procedure is public, the history recording capability can be used by the decoder to evade tracing.

Other Tracing Games In the setting for non-black box tracing game, the adversary is considered to be a pirate decoder that is capable of receiving the transmission through some key material that is made out of traitor keys. The non-black box tracing game refers to the case where the response of the adversary is defined to be the key material that makes the decoder succesful. This response is not necessarily a real reaction of the adversary but rather an abstract notion that in reality may include physical tampering on behalf of the tracer. In many settings by "reverse-engineering" a decoder, it might be possible to retrieve the key employed within. We note that a decryption key from the key-space K should be available to the tracer because of the unlikelihood of performing decryption without a key.

The scenario for pirate rebroadcasting aims to capture a different setting compared to black-box tracing : when the tracer not only does not have access to the pirate decoder physically but in fact it is incapable of performing tracing outside of normal broadcast to the users. In this scenario, the adversary first decrypts the content by using its traitor key material and then once it is in clear text form, it rebroadcasts the content. Clearly a traitor tracing scheme with merely black-box tracing capability is useless against a pirate rebroadcast attack. The tracer is entirely powerless in handling such an attacker as the output of the rebroadcast itself will potentially provide no information about the traitor keys. It is easy to see that the restrictions imposed to the tracer in the pirate rebroadcasting setting are captured by the notion of tracing alfresco as described in the different variations of the tracing game in Definition 4. This is the case as the tracer needs to perform its task by making queries that are statistically indistinguishable from members of $\mathcal{Q}_h$ where h is the previous queries of the tracer.

4 Traceability in Multiuser Encryption Schemes

We will now show that the linear length multiuser encryption scheme $\mathtt{ME}_L$ is a black box traitor tracing scheme against resettable pirate decoders. Resettable pirate decoders allow the tracer to 'reset' the adversary and to receive 'fresh' responses from the pirate that forgets the history of the interaction. This is the key fact in our choice of tracing queries; i.e. we will deviate from the normal set of random variables $\mathcal{Q}^{\mathbf{Transmit}}$ and query the decoder with some special tracing ciphertexts that will yield the identification of a traitor involved in the piracy.

We will first give a high level description of the old technique in Section 4.1 and present our new technique which decreases the number of rounds. The new technique is more like a replacement of the old technique based on walking argument. Particularly, the improvement we made here is readily applicable to the subset cover framework of [24] and any other traitor tracing schemes based on fingerprinting codes (cf. [1,6]).

4.1 Formal Analysis of Linear Tracing Strategy

We will now show that the linear length multiuser encryption scheme $\mathtt{ME}_L$ is a black box traitor tracing scheme against resettable pirate decoders. Recall that resettable pirate decoders allow the tracer to reset the adversary and to receive fresh responses forgetful of the history of the interaction. This is the key fact in our choice of tracing queries; in particular we will deviate from the normal set of random variables $\mathcal{Q}^{\mathbf{Transmit}}$ and query the decoder with some special tracing ciphertexts that will yield the identification of a traitor involved in piracy.

Recall that the linear length multiuser encryption scheme $\mathtt{ME}_L$ transmits a vector of ciphertext $\langle \mathtt{E}_{k_1}(m), \ldots, \mathtt{E}_{k_n}(m)\rangle$ where $(\mathtt{E}, \mathtt{D})$ is the underlying symmetric encryption scheme of $\mathtt{ME}_L$ and the key k_i is available to the i-th receiver. The tracing queries consist of the special transmission $\mathbf{Transmit}^s_L(ek, m)$ for $s = 0, 1, \ldots, n$ by substituting the first s ciphertexts with random strings.

$$\mathbf{Transmit}^s_L(ek, m) = \langle \mathtt{E}_{k_1}(R_1), \mathtt{E}_{k_2}(R_2), \ldots \mathtt{E}_{k_s}(R_s), \mathtt{E}_{k_{s+1}}(m), \ldots \mathtt{E}_{k_n}(m)\rangle \quad (1)$$

where R_i, for $i = 1, \ldots, s$, is a random string of the same length as the message m. Given that the adversary-tracer pair is σ-admissible the adversary will be required to respond the queries of type $\mathbf{Transmit}^0_L(ek, m)$ such that the predicate $\mathcal{R}^{BB}$ is satisfied with probability at least σ. On the other hand note that the predicate necessarily fails with overwhelming probability for queries of type $\mathbf{Transmit}^n_L(ek, m)$. This suggests that the tracer can progressively randomize the pattern of the ciphertext until a position is identified that the pirate-box fails to decrypt successfully whenever it queries the tracing ciphertext.

The soundness of the above argumentation is supported by the following lemma:

Lemma 1. *Assuming that $s \notin \mathsf{C}$, any probabilistic polynomial time adversary $\mathcal{A}$, on input $\{k_i\}_{i\in\mathsf{C}}$, distinguishes the distributions* $\mathbf{Transmit}^{s-1}_L(ek, m)$ *and* $\mathbf{Transmit}^s_L(ek, m)$ *with probability at most $2\varepsilon_p$ where $(tk, ek, sk_1, \ldots, sk_n) \leftarrow$* $\mathbf{KeyDist}(1^n)$ *assuming that the underlying encryption scheme $(\mathtt{E}, \mathtt{D})$ is ε_p-insecure in the sense of Definition 3.*

Let us define p_s as the probability that the box decodes the special tracing ciphertext $\mathbf{Transmit}^s_L(ek, m)$. Suppose, now, that the key k_s is not available to the adversary, it holds that the pirate decoder can distinguish between the probability spaces $\mathbf{Transmit}^s_L(ek, m)$ and $\mathbf{Transmit}^{s-1}_L(ek, m)$ only with small probability that is related to the insecurity of the underlying encryption scheme $\mathtt{E}(\cdot)$. As a result, it holds that $|p_{s-1} - p_s|$ is sufficiently small with respect to the advantage ε_p in being succesful in security game of the underlying primitive.

On the other hand, for a succesful pirate decoder it holds that $p_0 \geq \sigma$ due to the $\mathcal{A}$-constraint described in the tracing game, while $p_n < \frac{1}{|\mathsf{M}|}$ (M denotes the plaintext space.). Hence there must be some $0 < s \leq n$ for which $|p_{s-1} - p_s| \geq \frac{\sigma - 1/|\mathsf{M}|}{n}$ by the triangular inequality. If it turns out that $\frac{\sigma - 1/|\mathsf{M}|}{n} > 2\varepsilon_p$, this leads the accusation of the user possessing k_s due to the above lemma 1.

Having discussed roughly the interaction of a tracer exploiting the walking argument, we want to note that there are two different ways to locate the probability drop between successive tracing queries. One is due to [22] which results the number of rounds to be $O(n^3\sigma^{-2}\log 1/\varepsilon)$ where ε is the security parameter. The other interaction methodology is given by Naor, Naor and Lotspiech in [24,26] which resembles like a binary search of the unit gap where the probability drops sufficiently enough to make the accusation. This methodology results an interaction between the tracer and the adversary which has $O(n^2\sigma^{-2}\log\frac{1}{\varepsilon}\log^3 n)$ number of rounds.

For the sake of reference, we state(leaving the proof for full-version of the work) that the $\mathtt{ME}_L$ is a black-box traitor tracing scheme for which the corresponding tracing game for n-coalitions is winnable by a tracer exploits the walking argument of [22] against any probabilistic polynomial time adversarial algorithm $\mathcal{A}$. The tracer that we will construct queries the special transmission of the form $\mathbf{Transmit}_L^s(ek, m)$ for $s = 0, 1, \ldots, n$.

Theorem 2. *Consider a multiuser encryption scheme* $\mathtt{ME}_L$ *that employs a symmetric encryption scheme that is* ε_p*-insecure in the sense of Definition* 3. $\mathtt{ME}_L$, *on input* (1^n), *is a black box traitor tracing scheme for* n*-coalitions with probability* $1-\varepsilon$ *against resettable* σ*-pirates with* $\sigma > 4n\varepsilon_p + \frac{1}{|\mathsf{M}|}$. *It further holds that* $\mathsf{roundc}[\mathtt{ME}_L, \mathcal{T}_S] \leq \frac{48n^3\cdot\ln(8/\varepsilon)}{(\sigma-1/|\mathsf{M}|)^2}$.

4.2 Our New Tracing Technique Based on Fingerprinting Codes

Preliminaries on Fingerprinting Codes A fingerprinting code is a pair of algorithms ($\mathbf{CodeGen}, \mathbf{Identify}$) that is defined as follows: **CodeGen** is an algorithm that is given input 1^n, it samples a pair $(\mathcal{C}, tk) \leftarrow \mathbf{CodeGen}(1^n)$ where $\mathcal{C}$ is an (ℓ, n, q)-code defined over an alphabet Q with ℓ as a function of n and q, and the identifying key tk is some auxiliary information to be used for identifying that may be empty. We may use ℓ as a superscript, i.e. denoting by $\mathbf{CodeGen}^\ell$, to emphasize the fact that ℓ might be a function of n, q and some other parameters.

Identify is a deterministic algorithm that on input pair $(\mathcal{C}, tk)$, sampled by $\mathbf{CodeGen}(1^n)$, and a codeword $\mathsf{c} \in Q^\ell$, it outputs a codeword-index $t \in [n]$ or fails. Informally speaking, we say a fingerprinting code is (α, w)-identifier if c is constructed by a traitor coalition size of at most w by combining their codewords, the **Identify** algorithm outputs a traitor with a failure probability of at most α.

Tracing Queries. The tracing queries of the new technique consist of the special transmission $\mathbf{Transmit}_L^{\mathsf{S}}(ek, m)$ for any $\mathsf{S} \in [n]$ by substituting the ciphertexts $\mathtt{E}_{k_i}(m)$ with encryption of random strings if $i \in \mathsf{S}$.

$$\mathbf{Transmit}_L^{\mathsf{S}}(ek, m) = \langle \mathtt{E}_{k_1}(e_1), \mathtt{E}_{k_2}(e_2), \ldots, \mathtt{E}_{k_n}(e_n)\rangle \quad \text{and} \quad e_i = \begin{cases} R_i, \ i \in \mathsf{S} \\ m, \ i \notin \mathsf{S} \end{cases} \tag{2}$$

where R_i, for $i = 1, \ldots, n$, is a random string of the same length as the message m. Given that the adversary-tracer pair is σ-admissible the adversary will be required to respond the queries of type $\mathbf{Transmit}_L^{\emptyset}(ek, m) = \mathbf{Transmit}_L(ek, m)$ such that the predicate $\mathcal{R}^{BB}$ is satisfied with probability at least σ. The predicate is satisfied on responses of the queries of type $\mathbf{Transmit}_L^{\mathsf{S}}(ek, m)$ either with probability at least $\sigma/2$ or less than. Both of these cases will result in existence of a traitor in either S or $[n] \setminus \mathsf{S}$ if the probability σ is sufficiently large, i.e. $\sigma \geq 4n\varepsilon_p$.

The soundness of the above argumentation is based on the following lemma and we will elaborate more on after the lemma:

Lemma 2. *Let* $\mathsf{S} \subseteq [n]$ *satisfies* $\mathsf{C} \cap \mathsf{S} = \emptyset$ *for some set* $\mathsf{C} \subseteq [n]$. *Any probabilistic polynomial time adversary* $\mathcal{A}$, *given* $\{k_i\}_{i \in \mathsf{C}}$, *distinguishes the distributions* $\mathbf{Transmit}_L^{\emptyset}(ek, m)$ *and* $\mathbf{Transmit}_L^{\mathsf{S}}(ek, m)$ *with probability at most* $2n\varepsilon_p$ *where* $(tk, ek, sk_1, \ldots, sk_n) \leftarrow \mathbf{KeyDist}(1^n)$ *assuming that the underlying encryption scheme* $(\mathtt{E}, \mathtt{D})$ *is* ε_p*-insecure in the sense of Definition 3.*

The proof of the above lemma can be structured as a general case of the security theorem 1 of the linear length multiuser encryption scheme. In the statement above we considered the worst case where $\mathsf{S} = [n]$ for which the probability difference is bounded by $2n\varepsilon_p$ as the Theorem 2 suggests.

Let us define p_{S} as the probability that the box decodes the special tracing ciphertext $\mathbf{Transmit}_L^{\mathsf{S}}(ek, m)$. Suppose, now, that no key k_i for $i \in \mathsf{S}$ is available to the adversary, it holds that the pirate decoder can distinguish between the distributions $\mathbf{Transmit}_L^{\emptyset}(ek, m)$ and $\mathbf{Transmit}_L^{\mathsf{S}}(ek, m)$ only with probability at most $2n\varepsilon_p$ due to the Lemma 2, i.e. it holds that $|p_{\emptyset} - p_{\mathsf{S}}| \leq 2n\varepsilon_p$ where ε_p is the insecurity of the underlying encryption scheme $\mathtt{E}(\cdot)$.

On the other hand, if any key k_i available to the adversary satisfies for $i \in \mathsf{S}$, then it holds that the pirate decoder can decrypt the tracing ciphertexts of the form $\mathbf{Transmit}_L^{\mathsf{S}}(ek, m)$ with probability at most $2n\varepsilon_p$ due to the security theorem put forth by the Theorem 1, i.e. $p_{\mathsf{S}} \leq 2n\varepsilon_p$.

Suppose now that we have $\sigma = p_{\emptyset} > 4n\varepsilon_p$. We query the adversary with the tracing ciphertexts of the form $\mathbf{Transmit}_L^{\mathsf{S}}(ek, m)$ to approximate the success probability p_{S}. If $p_{\mathsf{S}} > 2n\varepsilon_p$ then we obtain the fact that there exists a traitor in the set $[n] \setminus \mathsf{S}$. Otherwise it holds that $\sigma - p_{\mathsf{S}} > 2n\varepsilon_p$ for which case we observe the existence of a traitor in the set S due to the Lemma 2. The following theorem 3 suggests that the failure in approximation of p_{S} will be bounded by ε' for $O(\frac{\log 1/\varepsilon'}{2\delta^2})$ many queries with $\sigma > 4n\varepsilon_p + 2\delta$.

The new tracing technique is parameterized by a binary fingerprinting code $\mathtt{F} = \langle \mathbf{CodeGen}, \mathbf{Identify} \rangle$ that generates an $(\ell, n, 2)$-code $\mathcal{C} = \{\mathsf{c}^1, \ldots, \mathsf{c}^n\}$. We then define the set $\mathsf{S}_j = \{i : \mathsf{c}_j^i = 0\}$. Observe that $[n] \setminus \mathsf{S}_j = \{i : \mathsf{c}_j^i = 1\}$ hold.

We submit the adversary with the tracing queries of type $\mathbf{Transmit}_L^{\mathsf{S}_j}(ek, m)$ for $j = 1, \ldots, \ell$. If it happens that $\sigma > 4n\varepsilon_p$ then for each $j = 1, \ldots, \ell$ a traitor exists in the set S_j or in the set $[n] \setminus \mathsf{S}_j$. We let $\mathsf{p}_j = 0$ if the S_j happens to contain a traitor and $\mathsf{p}_j = 1$ holds otherwise. We finally obtain a pirate codeword p that is produced by a coalition of traitors. Running the identification algorithm over the pirate codeword p outputs a receiver index that is found to be traitor. The overall failure probability is bounded by $\varepsilon_f + \ell\varepsilon'$ (for the failure in identification and failure in approximation respectively).

Theorem 3. *Consider a multiuser encryption scheme* $\mathtt{ME}_L$ *that employs a symmetric encryption scheme that is* ε_p*-insecure in the sense of Definition 3.* $\mathtt{ME}_L$*, on input* (1^n)*, is a black box traitor tracing scheme for* w*-coalitions with probability* $1 - \varepsilon_f - \ell\varepsilon'$ *against resettable* σ*-pirates with* $\sigma > 4n\varepsilon_p + 2\delta$*. It further holds that* $\mathsf{roundc}[\mathtt{ME}_L, \mathcal{T}^{\mathtt{F}}_{KP}] \leq \frac{3\ell\cdot\ln(2/\varepsilon')}{\delta^2}$ *where* ℓ *is the length of the binary fingerprinting code* $\mathtt{F} = (\mathbf{CodeGen}^{\ell}, \mathbf{Identify})$ *which is a* (ε_f, w)*-identifier.*

Note that the tracing round complexity is a function of ℓ, the length of the underlying fingerprinting code, the parameter δ required for the adversary's admissibility to approximate the success in decryption, and finally ε', the security parameter of the tracing scheme.

Instantiation 1. In our first instantiation of the tracer, we consider the optimal codes of [33]. This provides a tracing against any size of traitor coalition for $O(n^2\delta^{-2}\log\frac{1}{\varepsilon'}\log\frac{n}{\varepsilon_f})$ number of queries submitted by the tracer where n is the number of receivers and ε_f is the security parameter of the Tardos code. The bound follows from Theorem 3 and the fact that the length of Tardos' codes is $O(n^2\log(n/\varepsilon_f))$ where n is the number of codewords (that in our setting matches the number of receivers). Note that this scheme tolerates an *unlimited* number of traitors and revocations.

Instantiation 2. Our second instantiation employs again Tardos' codes but assuming an upper bound on the number of traitors w. This provides for code length of $O(w^2\log\frac{n}{\varepsilon_f})$ and given that in our setting we have that n is the number of codewords that should be equal to the number of receivers using theorem 3 we obtain a tracing round complexity of $O(w^2\delta^{-2}\log\frac{1}{\varepsilon'}\log\frac{n}{\varepsilon_f})$, i.e., with only logarithmic dependency on the number of receivers in the system.

In the above, we can set $\varepsilon_f = \varepsilon/2$ and $\varepsilon = \frac{\varepsilon'}{2\ell}$, thus obtaining the round complexity of $\frac{3\ell\cdot\ln(4\ell/\varepsilon)}{\delta^2}$. Such selection will yield the results of the table 1(by neglecting $\log n/\varepsilon_f$ as the asymptotic complexity already includes $\frac{n^2\log n/\varepsilon_f}{\varepsilon_f}$).

References

1. Boneh, D., Naor, M.: Traitor Tracing with Constant Size Ciphertext. In: ACM Conference on Computer and Communications Security, pp. 501–510. ACM, New York (2008)
2. Berkman, O., Parnas, M., Sgall, J.: Efficient dynamic traitor tracing. In: SODA 2000, pp. 586–595 (2000)
3. Boneh, D., Franklin, M.: An Efficient Public-Key Traitor Tracing Scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 338–353. Springer, Heidelberg (1999)

4. Boneh, D., Sahai, A., Waters, B.: Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
5. Boneh, D., Shaw, J.: Collusion-Secure Fingerprinting for Digital Data. IEEE Transactions on Information Theory 44(5), 1897–1905 (1998)
6. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
7. Chor, B., Fiat, A., Naor, M., Pinkas, B.: Tracing Traitors. IEEE Transactions on Information Theory 46(3), 893–910 (2000)
8. Chabanne, H., Hieu Phan, D., Pointcheval, D.: Public Traceability in Traitor Tracing Schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 542–558. Springer, Heidelberg (2005)
9. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
10. Dodis, Y., Fazio, N., Kiayias, A., Yung, M.: Scalable public-key tracing and revoking. In: PODC 2003, Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing (PODC 2003), Boston, Massachusetts, July 13-16 (2003)
11. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.P.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: STOC 2009, pp. 381–390 (2009)
12. Feige, U., Raghavan, P., Peleg, D., Upfal, E.: Computing with Noisy Information. SIAM J. Comput. 23(5), 1001–1018 (1994)
13. Fiat, A., Tassa, T.: Dynamic Traitor Tracing. Journal of Cryptology 4(3), 211–223 (2001)
14. Gafni, E., Staddon, J., Lisa Yin, Y.: Efficient Methods for Integrating Traceability and Broadcast Encryption. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 372–387. Springer, Heidelberg (1999)
15. Jin, H., Lotspiech, J.: Renewable Traitor Tracing: A Trace-Revoke-Trace System For Anonymous Attack. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 563–577. Springer, Heidelberg (2007)
16. Jin, H., Lotspiech, J.B.: Unifying Broadcast Encryption and Traitor Tracing for Content Protection. In: ACSAC 2009, pp. 139–148 (2009)
17. Jin, H., Lotspiech, J., Nusser, S.: Traitor tracing for prerecorded and recordable media. In: Digital Rights Management Workshop 2004, pp. 83–90 (2004)
18. Kiayias, A., Pehlivanoglu, S.: Pirate Evolution: How to Make the Most of Your Traitor Keys. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 448–465. Springer, Heidelberg (2007)
19. Kiayias, A., Pehlivanoglu, S.: Tracing and Revoking Pirate Rebroadcasts. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 253–271. Springer, Heidelberg (2009)
20. Kiayias, A., Yung, M.: Self Protecting Pirates and Black-Box Traitor Tracing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 63–79. Springer, Heidelberg (2001)
21. Kiayias, A., Yung, M.: On Crafty Pirates and Foxy Tracers. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, pp. 22–39. Springer, Heidelberg (2002)
22. Kiayias, A., Yung, M.: Traitor Tracing with Constant Transmission Rate. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 450–465. Springer, Heidelberg (2002)

23. Kurosawa, K., Desmedt, Y.: Optimum Traitor Tracing and Asymmetric Schemes. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 145–157. Springer, Heidelberg (1998)
24. Naor, D., Naor, M., Lotspiech, J.B.: Revocation and Tracing Schemes for Stateless Receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
25. Naor, M., Pinkas, B.: Threshold Traitor Tracing. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 502–517. Springer, Heidelberg (1998)
26. Naor, D., Naor, M., Lotspiech, J.B.: Revocation and Tracing Schemes for Stateless Receivers. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 43 (2002)
27. Naor, M., Pinkas, B.: Efficient Trace and Revoke Schemes. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 1–20. Springer, Heidelberg (2001)
28. Hieu Phan, D., Safavi-Naini, R., Tonien, D.: Generic Construction of Hybrid Public Key Traitor Tracing with Full- Public-Traceability, pp. 264–275.
29. Safavi-Naini, R., Wang, Y.: Sequential Traitor Tracing. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 316–332. Springer, Heidelberg (2000)
30. Safavi-Naini, R., Wang, Y.: Traitor Tracing for Shortened and Corrupted Fingerprints. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 81–100. Springer, Heidelberg (2003)
31. Silverberg, A., Staddon, J., Walker, J.L.: Efficient Traitor Tracing Algorithms Using List Decoding. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 175–192. Springer, Heidelberg (2001)
32. Staddon, J.N., Stinson, D.R., Wei, R.: Combinatorial Properties of Frameproof and Traceability Codes. IEEE Transactions on Information Theory 47(3), 1042–1049 (2001)
33. Tardos, G.: Optimal probabilistic fingerprint codes. In: ACM 2003, pp. 116–125 (2003)

Password Based Key Exchange Protocols on Elliptic Curves Which Conceal the Public Parameters

Julien Bringer[1], Hervé Chabanne[1,2], and Thomas Icart[3]

[1] Sagem Sécurité
[2] Télécom ParisTech
[3] This work has been done while this author was affiliated with Sagem Sécurité and the University of Luxembourg

Abstract. We here describe a new Password-based Authenticated Key Exchange (PAKE) protocol based on elliptic curve cryptography. We prove it secure in the Bellare-Pointcheval-Rogaway (BPR) model. A significant novelty in our work is that the elliptic curve public parameters remain private. This is important in the context of ID contactless devices as, in this case, there will exist most probably a way to link these parameters with the nationality of the ID document owners.

Keywords: Password-based Authenticated Key Exchange, Elliptic Curves, Privacy.

1 Introduction

To enable secure communication over insecure channels, two parties encrypt and authenticate their messages using a shared secret key, usually obtained through key exchange. A key exchange protocol enables the two parties to establish a common secret in an authenticated way (Authenticated Key Exchange, AKE). The goal is for the session key to be known only by the parties involved in the protocol; the session key should be indistinguishable from a random data. Password-based key exchange protocols are a convenient way to achieve this. The two parties rely on a shared low-entropy secret (e.g. a four-digit PIN) to derive a common high-entropy session key.

Password-Based Authenticated Key Exchange (PAKE) protocols are today considered in the context of identity documents to ensure the security of the communications between the chip and a reader [18]. With machine readable travel documents (MRTD, cf. International Civil Aviation Organization specifications [15]), the data stored on the machine readable zone (MRZ) are seen as low-entropy shared information between the reader and the chip to establish a secure link. For efficiency constraints, the protocols usually rely on elliptic curves. Moreover, when these protocols will be used in real ID documents, the parameters will almost surely depend on the nationality of the document owner (each country keeps usually its freedom of choice in matter of cryptographic parameters with respect to its own evaluation of the associated level of security).

However, if one eavesdrops the communication resulting from executions of an AKE protocol based on elliptic curves, then he would learn some elliptic curve points and would be able to obtain the elliptic curve parameters [2]. He therefore retrieves the owner's nationality as a side information, thus leading to a privacy leakage and security issues. This is our motivation for finding PAKE protocols based on elliptic curves while hiding the elliptic curve parameters. To the best of our knowledge, our work is the first to provide a secure solution to this problem.

1.1 Related Works

AKE protocols. Password-based authenticated key exchange was considered first by Bellovin and Merritt [5]. The goal is to authenticate a key exchange between two parties based on simple passwords possibly from a small dictionary that an adversary may know. The basic idea behind the different schemes described in [5] was to encrypt some messages of the key exchange (hence the acronym EKE for Encrypted Key Exchange). This work was followed by many variants and later on several security analysis in different models, e.g. [5, 6, 3, 4, 9, 10, 8]. The main issue is the resistance against offline dictionary attacks.

One of the most well-known variant of EKE is Diffie-Hellman EKE (DH-EKE) which is merely a DH key exchange where at least one message exchanged under DH is encrypted via the password. Bellare *et al.* introduced in [3] a formal security model to grasp the specificity of password-based key exchange. This model was used later in [9, 10] to establish the security of DH-EKE under ideal assumptions (namely the ideal cipher model and the Random Oracle Model, ROM). However ideal cipher model is not easily applicable to elliptic curves and thus in its classical version, DH-EKE is not well adapted to elliptic curves. A naive application of the encryption step of DH-EKE to elliptic curves points would lead to an insecure scheme. Indeed, due to the redundancy in a point representation, partition attacks [16] are made possible to distinguish possible passwords from impossible ones and the security against offline dictionary attacks does not hold (the main idea is that a decryption with a bad password of an encrypted point would most probably not be a point over the elliptic curve). In [8], a modification of the scheme is suggested by using either a point of the curve or a point over its twist. Although, this enables to withstand the security issue, the scheme is not proved in the model of [3] and becomes much less simple than DH-EKE.

Another well-known and widely used PAKE is SPEKE (Simple Password Exponential Key Exchange) [14]. It is part of the IEEE P1363.2 standard. Likewise, it is based on Diffie-Hellman key exchange but the password is here used to select the generator of the DH key exchange, which is then operated without encrypted messages. To do so, a hash of the password is used to generate a group element, and the security is based on the ROM. Here again security against offline dictionary attacks does not hold if you do not have a way to hash into elliptic curves (cf. also Remark 5).

Finally, in both cases, DH-EKE and SPEKE do not hide the curve parameters as they can easily be deduced from 2 eavesdropped points.

The BPR (Bellare-Pointcheval-Rogaway) model. Several security models have been suggested to analyze the security of password-based AKE. The BPR model [3] is now considered as a standard model for PAKE protocols. It captures well the security requirements that a PAKE should satisfy. In particular even if protocols remain subject to online guessing attacks, they should thwart offline dictionary attacks. Different protocols have been shown secure in this model. The model is based on the Find-Then-Guess principle where an adversary – mounting an active attack against several protocol instances running concurrently – should not be able to determine whether a session key is the actual one, i.e. that the key should be indistinguishable from a random string. This also ensures an implicit authentication between the two parties involved in the protocol. The model is refined in [1] by requiring that all the concurrent session keys look random (Real-Or-Random principle).

Admissible Encodings. The notion of admissible encoding has been introduced by Boneh-Franklin in order to hash into elliptic curves since it is required for their Identity-Based Encryption scheme [7]. Later, Coron and Icart [12] have introduced a more general notion of admissible encodings. These encodings are built out of deterministic functions that map bit strings into elliptic curves such the ones of [13,17].

An admissible encoding from $\{0,1\}^*$ into a group is a function that enables to transform any bit string into a group element. Moreover, to comply with the definition, there must exist a polynomial-time inversion algorithm that can compute a bit string from a group element. The existence of this algorithm ensures that a random group element is mapped into a random bit string. Thanks to these encodings, [12] describes a way to create a random oracle into an elliptic curve. From [12], we here only use the fact that a random point can be represented by a random bit string.

This property can be used within the DH-EKE protocol, as it is much safer to encrypt random bit strings, especially when the key is directly derived from a low entropy password.

1.2 Our Contribution

We provide the first PAKE protocol which ensures that the elliptic curve public parameters remain hidden. This holds even against dictionary attacks. This is made possible thanks to the existence of admissible encodings. Since each elliptic curve point can be seen as a random bit string, it enables to encrypt a point by processing a bit string through a block cipher. The password can here be seen as a seed for computing the secret key used in the block cipher. This has a direct application within DH-EKE, as an eavesdropper cannot verify which elliptic curve parameters are used. Since eavesdroppers only see “random” bit strings, whatever are the elliptic curve public parameters, they cannot tell which ones are used.

This first construction is a direct application of admissible encodings on elliptic curves. Moreover, we provide a second PAKE protocol where we exploit the fact that the knowledge of the elliptic curve parameters can be interpreted as a shared secret. This is our main result as this enables to drop out the encryption of EKE while the protocol can still be proved secure. We introduce complexity assumptions based on the discrete logarithm problem ensuring that finding one bit string which represents 2 points with known discrete logarithm from 2 different curves is hard. These assumptions enable us to prove the security of our Diffie-Hellman AKE protocol where the password directly relies on the elliptic curve parameters. The hardness of these new complexity assumptions holds in the generic group model (as it is proved in the extended version of the paper [11]).

2 Definitions

2.1 Admissible Encoding and Admissible Representation

We here recall the definition of an admissible encoding.

Definition 1. *Given 2 random variables X and Y over a set S, we say that the distribution of X and Y are (ϵ)-statistically indistinguishable if:*

$$\sum_{s \in S} |\Pr(X = s) - \Pr(Y = s)| < \epsilon.$$

Moreover, given a security parameter, two distributions are statistically indistinguishable if they are (ϵ)-statistically indistinguishable for an ϵ negligible in the security parameter.

Definition 2 (Admissible Encoding, [12]). *A function $F : S \to R$ is said to be an ε-admissible encoding if:*

1. *F is computable in deterministic polynomial time;*
2. *There exists such a probabilistic polynomial time algorithm $\mathcal{I}_F$: given $r \in R$ as input, $\mathcal{I}_F$ outputs s such that either $F(s) = r$ or $s = \perp$, and the distribution of s is ε-statistically indistinguishable from the uniform distribution in S when r is uniformly distributed in R.*

When an admissible encoding from $\{0,1\}^L$ into a curve exists, its inversion algorithm $\mathcal{I}_F$ enables to transform uniformly distributed elliptic curve points into uniformly distributed bit strings. Encrypting an elliptic curve point thus becomes easier when it is represented as a bit string. In particular, no trivial partition attack is possible.

Icart's Mapping in Characteristic 2. The equation which defines an elliptic curve $E_{a,b}$ in characteristic 2 is of general form:

$$(E_{a,b}) \qquad Y^2 + XY = X^3 + aX^2 + b$$

where a and b are elements of $\mathbb{F}_{2^n}$. For an odd n, the map $x \mapsto x^3$ is a bijection. Let

$$\begin{aligned} f_{a,b} : \mathbb{F}_{2^n} &\mapsto (\mathbb{F}_{2^n})^2 \\ u &\mapsto (x, ux + v^2) \end{aligned}$$

where $v = a + u + u^2$ and $x = (v^4 + v^3 + b)^{1/3} + v$. It is clear that, whenever computing a cube root is an exponentiation, computing $f_{a,b}$ is a deterministic polynomial time algorithm.

Lemma 1 (Hashing into $E_{a,b}$, [13]). *Let $\mathbb{F}_{2^n}$ be a field with n odd. For any $u \in \mathbb{F}_{2^n}$, $f_{a,b}(u)$ is a point of $E_{a,b}$.*

We here focus on characteristic 2 version of [13] for two reasons: the computation is simpler than in the general case, and the inverting algorithm is deterministic and easy to implement [11]. Note that this encoding is not the only known general encoding for elliptic curves. In [17], Shallue and van de Woestijne proposed another encoding. We choose to introduce only the encoding from [13] as the encoding from [17] is not as simple to describe. But it is possible to adapt our work to any existing admissible encoding.

Let E be an elliptic curve over a field $\mathbb{F}_{2^n}$. From such a point encoding f and a generator G of the group of points, an admissible encoding F from $\{0,1\}^{2n}$ to E can be constructed. Let l be a $2n$-bit long string. This string is split in 2 substrings $u||\lambda$ where λ is seen as a n-bit integer. We then introduce what we call in the sequel the Coron-Icart admissible encoding:

$$F(l) = f(u) + \lambda \cdot G \tag{1}$$

The resulting function is proved to be an admissible encoding in [12] with a negligible ϵ. The main condition on f is to be an encoding that satisfies a condition weaker than in Definition 2, where the inversion algorithm needs to work only on a polynomial fraction of the inputs and where the statistically indistinguishability is measured only with respect to this fraction. Such encoding is denoted in [12] a weak encoding.

Admissible Representation. We introduce below the notion of admissible representations. These representations are the outputs of $\mathcal{I}_F$, when it is applied to an elliptic curve point.

Definition 3 (Admissible Representation). *Assume that F is an admissible encoding from S to R. For any $r \in R$, we define as an admissible representation of r any output of $\mathcal{I}_F(r)$.*

An element $r \in R$ may have many different admissible representations. Furthermore, an uniformly random $r \in R$ has an uniformly random admissible representation $s \in S$. For instance, following Eq. (1) a random point P of an elliptic curve admits an admissible representation of the form (u, λ) where $u||\lambda$ is a random bit string of size $2n$.

2.2 BPR Security Model

This model defines the notions of partnership, session key freshness and security against dictionary attacks. The model considers a set of honest players who do not deviate from the protocol. The adversary controls all the communications network. This is an active adversary formalized through queries. Users can have many protocol instances running concurrently. The adversary can create, modify, or forward messages and has oracle access to the user instances.

Let A and B be two users which can be part of the key exchange protocol P. Several concurrent instances may run in different executions of P: they are denoted by A_i and B_j. The server and the user share a low-entropy secret *pw* uniformly drawn from a dictionary of size N.

Oracles. The protocol P consists of the execution of a key exchange algorithm. It is an interactive protocol between A_i and B_j that provides the instances of A and B with a session key *sk*. The adversary $\mathcal{A}$ has access to the following oracles for controlling the interactions.

- Execute(A_i, B_j) simulates a passive attack where $\mathcal{A}$ eavesdrops the communication. It causes an honest execution of P between fresh instances A_i and B_j.
- Send(U_i, m) models $\mathcal{A}$ sending a message m to instance U_i ($U = A$ or B). The output is the message generated by U in processing the message m according to the protocol and the state of the instance. It simulates an active attack.
- Reveal(U_i) returns the session key of the input instance. This query models the misuse of the session key by instance U_i. The query is only available to the adversary if the targeted instance actually holds a session key (i.e. if the protocol has correctly terminated).

Security Notions. Two instances are defined as **partnered** if both instances have terminated correctly with the same session key. The **freshness** notion captures the fact that a session key has not been directly leaked. An instance is said to be fresh in the current protocol execution if the instance has terminated and neither a Reveal query has been called on the instance nor on a partnered instance.

The Test(U_i) query models the semantic security of the session key. It is available to the adversary only if the aimed instance is fresh. When called, the oracle tosses a coin b and returns the session key *sk* if $b = 0$ or a random value (from the domain of keys) if $b = 1$.

The **AKE security** is then defined as follows. By controlling executions of the protocol P, the adversary $\mathcal{A}$ tries to learn information on the session keys. The game is initialized by drawing a password *pw* from the dictionary and by letting $\mathcal{A}$ ask a polynomial number of queries. At the end of the game, $\mathcal{A}$ outputs its guess b' for the bit output by the Test oracle.

The AKE advantage of the adversary $\mathcal{A}$ for the key exchange protocol P is denoted

$$\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = |\Pr[b' = b] - \frac{1}{2}|$$

Definition 4. *The protocol P is said to be* **AKE-secure** *if the adversary's advantage is negligible in the security parameter, for any polynomially bounded adversary.*

A strategy of proof consists generally in the simulation of all the oracles to show that there is no leakage.

Remark 1. An oracle Corrupt is also available in this model to analyze the **forward secrecy**. When called with respect to one player, the adversary will obtain the player's password. For AKE with forward secrecy, the Test query should not be related to a player corrupted before the Test query. Nevertheless, corruption after the query is allowed.

2.3 Classical Assumption

We recall the classical Computational Diffie-Hellman (CDH) assumption.

Definition 5 (CDH Assumption). *Let E be an elliptic curve and G be a generator of a subgroup of points of prime order. Let $\mathcal{A}$ be an algorithm that:*

- *inputs two random points $P = a \cdot G$ and $Q = b \cdot G$;*
- *and outputs $R = ab \cdot G$.*

The CDH assumption ensures that the best polynomial time adversary has a negligible probability of success, when the probability is taken over P and Q.

Throughout this work, we define $\mathsf{CDH}_G(P, Q)$ to be the correct value of R. We introduce in the next section assumptions related to this problem when the elliptic curve parameters are unknown. In particular, given two elliptic curves E_1, E_2, we rely on the hardness of finding two admissible representations l and l' such that the points $\mathsf{CDH}(F_1(l), F_1(l')) \in E_1$ and $\mathsf{CDH}(F_2(l), F_2(l')) \in E_2$ are known (with F_i an admissible encoding into E_i, $i \in \{1, 2\}$).

3 A New Family of Complexity Assumptions

In order to prove the strength of our protocol, we introduce complexity assumptions that arise from the fact that we are using in the same scheme different elliptic curve parameters. As already mentioned, these assumptions are new due to our specific setting. However, we strongly justify the difficulty of these assumptions in the sequel.

Throughout this section, we use the following definitions. Let k be a security parameter. Let S be a set of $N = poly(k)$ sets of elliptic curve parameters: $\{a_i, b_i, q_i, G_i\}_{i \in [1,N]}$ over a field $\mathbb{F}_{2^n}$ (i.e. elliptic curves $E_i := E_{a_i,b_i}$ over $\mathbb{F}_{2^n}$ with a point G_i, generator of a subgroup of points of order q_i) such that:

- for each i, an admissible encoding (cf. Definition 2) exists over E_{a_i,b_i};
- q_i is a prime integer and its cofactor is 2 (more generally we need the same cofactor for all curves);
- for all $i \neq j$, we have $q_i \neq q_j$.

The last point ensures that there does not exist an isomorphism between the different curves. It is important since it ensures that the discrete logarithm of a point over E_i is not related to a discrete logarithm over another E_j.

Let F_i be the admissible encoding associated to E_i. In the sequel, we mainly focus on the Coron-Icart admissible encoding obtained via Equation. (1) (Section 2.1) with the point encoding from [13]. It ensures that an admissible representation of size $2n$ exists for almost all points.

3.1 Hard Problems Around the Discrete Logarithm of the Points P_i

One question arises from this setting: *Given a bit string l, is the discrete logarithm of each $P_i = F_i(l)$ in basis G_i still hard to compute?*

Since an admissible encoding has an inversion algorithm, over each curve E_i, given a point with an unknown discrete logarithm, we can almost always (except with a negligible probability) compute one of its admissible representations and thus we have:

Lemma 2. *Assume that F_i is an admissible encoding. Computing the discrete logarithm of any $P_i = F_i(l)$ with the knowledge of l is as hard as solving the discrete logarithm problem over the curve E_i.*

When an adversary computes an admissible representation l of a point P_i over E_i, we also want that for each admissible representation, his advantage on the discrete logarithm of $P_j = F_j(l)$ in basis G_j over E_j, for some $j \neq i$, remains negligible.

Definition 6 (Admissible Encoding Twin Discrete Logarithm Assumption). *Let $\mathcal{A}$ be an algorithm that:*

- *inputs S;*
- *outputs l and a pair $(r_i, r_j) \in \left(\mathbb{Z}/q_i\mathbb{Z}^\times \times \mathbb{Z}/q_j\mathbb{Z}^\times\right)$ such that $P_i = F_i(l) = r_i \cdot G_i$ and $P_j = F_j(l) = r_j \cdot G_j$.*

The AET-DL assumption holds if any polynomial algorithm succeeds with a negligible probability, when the probability is taken over S.

Remark 2. This assumption can be expressed differently for the Coron-Icart admissible encoding of Eq. (1). Indeed, for this encoding, l is a pair of values (u, λ) such that $F_i(l) = f_i(u) + \lambda \cdot G_i$. Clearly, finding u and a pair (r_i, r_j) such that $f_i(u) = r_i \cdot G_i$ and $f_j(u) = r_j \cdot G_j$ is equivalent to solving the AET-DL problem. The problem is thus to find r_1, r_2 such that $f_1^{-1}(r_1 \cdot G_1) \cap f_2^{-1}(r_2 \cdot G_2)$ is a non-empty set. For a random pair $(G, G') \in E_i \times E_j$ the probability to have a u such that $f_i(u) = G$ and $f_j(u) = G'$ is at most $4 \times \frac{4}{2^n} = 2^{-(n-4)}$ for the Icart mapping, because any point has at most 4 preimages (see [13]) through this mapping. Since the scalar multiplication is a one-way map in each E_i, it is computationally hard to find such pairs.

In addition to the above justification of the difficulty of the AET-DL problem, we formally prove in the generic group model the hardness of this AET-DL problem in [11].

Definition 7 (Admissible Encoding Twin Computational Diffie-Hellman Assumption). *Let $\mathcal{A}$ be an algorithm that:*

– *inputs S and l;*
– *outputs l' and a pair of points (R_i, R_j) (both points different from the neutral element) such that*
  $\mathsf{CDH}_{G_i}(F_i(l), F_i(l')) = R_i$ *and* $\mathsf{CDH}_{G_j}(F_j(l), F_j(l')) = R_j$.

The AET-CDH assumption holds if any polynomial time adversary has a negligible probability of success, when the probability is taken over S and l.

This assumption is stronger than the AET-DL assumption because the AET-CDH problem can be solved using the l, r_i, r_j of the AET-DL assumption.

Remark 3. Due to the special form of admissible encodings defined by Eq. (1), F_i can be replaced by f_i in this assumption. For this reason, the AET-CDH assumption ensures that an adversary, who receives a bit string u, cannot compute u' such that over E_i and E_j he knows both $\mathsf{CDH}_{G_i}(f_i(u'), f_i(u))$ and $\mathsf{CDH}_{G_j}(f_j(u'), f_j(u))$. It is easily seen that $\mathsf{CDH}_{G_i}(f_i(u'), f_i(u)) = R_i$ implies $\mathsf{CDH}_{f_i(u)}(G_i, R_i) = f_i(u')$. The AET-CDH problem is thus to find $R_i \in E_i$ and $R_j \in E_j$ such that

$$f_i^{-1}(\mathsf{CDH}_{f_i(u)}(G_i, R_i)) \cap f_j^{-1}(\mathsf{CDH}_{f_j(u)}(G_j, R_j)) \neq \emptyset$$

As above, the probability for a random pair $(G, G') \in E_i \times E_j$ to have a u' such that $f_i(u') = G$ and $f_j(u') = G'$ is at most $4 \times \frac{4}{2^n} = 2^{-(n-4)}$ for the Icart mapping. Thanks to AET-DL, choosing u such that the adversary knows both logarithms of G_i in basis $f_i(u)$ and G_j in basis $f_j(u)$ is hard. Thus either the map $R_i \mapsto \mathsf{CDH}_{f_i(u)}(G_i, R_i)$ or the map $R_j \mapsto \mathsf{CDH}_{f_j(u)}(G_j, R_j)$ is one way. Consequently, it is computationally hard to find a pair (R_i, R_j).

As for the AET-DL assumption, the proof of the validity of the AET-CDH assumption in the generic group model is given in [11].

Remark 4. The AET-DL assumption is stronger than the DL assumption. Likewise, AET-CDH is a stronger assumption than the CDH assumption over any elliptic curve E_i. Indeed, AET-CDH is trivial if, for one curve E_j, CDH is an easy problem. The following algorithm illustrates this.

1. Randomly select r_i, compute $P_i = r_i \cdot G_i, R_i = r_i \cdot F_i(l)$.
2. Compute $l' = \mathcal{I}_{F_i}(P_i)$.
3. Compute $R_j = \mathsf{CDH}_{G_j}(F_j(l), F_j(l'))$ and return l', R_i, R_j.

We finally introduce a last assumption, which is the password based variant of the AET-DL assumption.

Definition 8 (n-Password Based Admissible Encoding Twin Computational Diffie-Hellman Assumption). *Let P_π be a point over E_{a_π,b_π}. Let l be an admissible representation of P_π ($P_\pi = F_\pi(l)$). Let $\mathcal{A}$ be a polynomial algorithm that:*

– *inputs S and l;*
– *outputs l', $K_1, \ldots, K_n$, where each K_i is a point of one of the curves in S.*

The n-PAET-CDH assumption holds if any polynomial adversary $\mathcal{A}$ has a probability $1/N + \varepsilon$ to have returned one K_i such that $\mathsf{CDH}_{G_\pi}(F_\pi(l), F_\pi(l')) = K_i$, where ε is negligible and N corresponds to the dictionary size (i.e. the number of possible curves).

In this assumption, ε is the advantage of the algorithm over the value of π. Indeed, a trivial way to solve the n-PAET-CDH problem is, from S and l, to randomly choose a $j \in [1, N]$ and to assume that $j = \pi$. This has a probability at least $1/N$ to succeed. Further, this assumption implies the AET-CDH assumption. Indeed an algorithm $\mathcal{A}^{\mathsf{aet-cdh}}$, which solves the AET-CDH problem, can be transformed into an adversary which solves the n-PAET-CDH problem with $\varepsilon = 1/N$. The following lemma proves that the converse is also true.

Lemma 3. *The AET-CDH assumption implies the n-PAET-CDH assumption, for any n.*

Proof. Let Succ^x be the probability of success of the best adversary against the problem x. Let Event_i be the event that an algorithm outputs $i \leq n$ points $K_{j_1}, \ldots, K_{j_i}$ such that

$$\mathsf{CDH}_{G_{j_i}}(F_{j_i}(l), F_{j_i}(l')) = K_{j_i}$$

We have:

$$\Pr\left[\mathsf{Succ}^{paet-cdh}\right] \leq \sum_{i=1}^{\min(N,n)} \Pr\left[\mathsf{Event}_i\right] \frac{i}{N}$$

It is easily seen that for all $i \geq 2$, $\Pr\left[\mathsf{Event}_i\right] \leq \Pr\left[\mathsf{Succ}^{aet-cdh}\right]$. This leads to:

$$\begin{aligned}\Pr\left[\mathsf{Succ}^{paet-cdh}\right] &\leq \Pr\left[\mathsf{Event}_1\right]\frac{1}{N} + \Pr\left[\mathsf{Succ}^{aet-cdh}\right]\frac{N-1}{2} \\ &\leq \frac{1}{N} + \Pr\left[\mathsf{Succ}^{aet-cdh}\right]\frac{N-1}{2}\end{aligned}$$

If $\Pr\left[\mathsf{Succ}^{aet-cdh}\right]$ is negligible, since N is polynomial, then: $\Pr\left[\mathsf{Succ}^{paet-cdh}\right] = \frac{1}{N} + \varepsilon$ $\square$

4 The EC-DH-EKE Protocol with an Admissible Encoding

The Diffie-Hellman Encrypted Key Exchange (DH-EKE) protocol is roughly a DH key exchange where each data sent is encrypted by a block cipher with a key derived from a shared secret. This protocol has been introduced in [5], extended in [3], and proved in the ideal cipher model and random oracle model under the CDH assumption in [9]. Its basic flows are presented in Figure 1 (a complete execution, with the final authentication checks, is given in Figure 2 in our elliptic curve instantiation).

Device	parameters : E, N, G	*Reader*
password π		password π
Compute $K_{pw} = H(\pi)$		Compute $K_{pw} = H(\pi)$
Pick α		Pick β
Compute $G_1 = \alpha \cdot G$	$\xrightarrow{z_1=\mathcal{E}_{K_{pw}}(G_1)}$	Compute $G_2 = \beta \cdot G$
	$\xleftarrow{z_2=\mathcal{E}_{K_{pw}}(G_2)}$	
Compute $K = \alpha \cdot \mathcal{D}_{K_{pw}}(z_2)$		Compute $K = \beta \cdot \mathcal{D}_{K_{pw}}(z_1)$

Fig. 1. Basic flows of the DH-EKE scheme

Note however that it is assumed that the ideal cipher inputs group elements. Consequently, a naive implementation of the DH-EKE over elliptic curves could be insecure. Indeed, the encryption of a point $P = (x, y)$ with a key $K_{pw} = H(\pi)$ leads to a ciphertext $z = \mathcal{E}_{K_{pw}}(x||y)$. However, for any password $\pi' \neq \pi$, the decryption of z is not a point over the elliptic curve with an overwhelming probability. This leads to an offline dictionary attack (see for instance [8]). More generally, since there exists a redundancy in the representation of $P = (x, y)$, it is difficult to encrypt P without having a dictionary attack. The encryption over the elliptic curves points should in fact be a permutation. One possibility to address this problem is to represent P thanks to an admissible representation. Hence applying a classical cipher would become possible.

4.1 Parameters

Let k be a security parameter. Let $\mathcal{H}$ be a hash function with $\{0,1\}^l$ as range. Let N be the size of $\mathcal{D}$, the dictionary of the different passwords. Let $E_{a,b}$ be an elliptic curve over $\mathbb{F}_{2^{2k+1}}$ with an admissible encoding F (and a related inversion algorithm $\mathcal{I}_F$) and G be a generator of its prime order subgroup of order q, with a cofactor 2.

We assume that the protocol takes place between different devices D and a reader R. Each device possesses a password $\pi \in \mathcal{D}$.

4.2 EC-DH-EKE

The DH-EKE scheme has been proved secure in [9] in the ideal cipher model and the random oracle model under the CDH assumption. However, the ideal cipher requires to manage group elements as inputs.

Thanks to the admissible encoding, a group element can be seen as a bit-string. For this reason, a real implementation of the protocol is much more realistic because the ideal cipher can be instantiated by a cipher such as AES-128 in ECB mode, while an ideal cipher over elliptic curve points is still to be found. The resulting protocol is described by Figure 2.

Device		*Reader*
	parameters : $E_{a,b}, N, G$	
password π		password π
Compute $K_{pw} = H(\pi)$		Compute $K_{pw} = H(\pi)$
Pick α		Pick β
Compute $G_1 = \alpha \cdot G$		Compute $G_2 = \beta \cdot G$
Compute $l_1 = \mathcal{I}_F(G_1)$		Compute $l_2 = \mathcal{I}_F(G_2)$
	$\xrightarrow{z_1 = \mathcal{E}_{K_{pw}}(l_1)}$	
	$\xleftarrow{z_2 = \mathcal{E}_{K_{pw}}(l_2)}$	
Compute $l_2 = \mathcal{D}_{K_{pw}}(z_2)$		Compute $l_1 = \mathcal{D}_{K_{pw}}(z_1)$
Compute $K = \alpha \cdot F(l_2)$		Compute $K = \beta \cdot F(l_1)$
Compute $\mathcal{K} = \mathcal{H}(K, z_1, z_2)$		Compute $\mathcal{K} = \mathcal{H}(K, z_1, z_2)$
Compute $K_{\mathsf{enc}} = \mathcal{H}(\mathcal{K}, 1)$		Compute $K_{\mathsf{enc}} = \mathcal{H}(\mathcal{K}, 1)$
Compute $K_{\mathsf{mac}} = \mathcal{H}(\mathcal{K}, 2)$		Compute $K_{\mathsf{mac}} = \mathcal{H}(\mathcal{K}, 2)$
Compute $T_D = \mathcal{H}(K_{\mathsf{mac}}, z_2)$		Compute $T_R = \mathcal{H}(K_{\mathsf{mac}}, z_1)$
	$\xrightarrow{T_D}$	
	$\xleftarrow{T_R}$	
Abort if T_R invalid		Abort if T_D invalid

Fig. 2. The EC-DH-EKE scheme with an Admissible Encoding

This finally leads to an efficient and secure protocol. Additionally, the elliptic curves parameters remain hidden from an eavesdropper, since it only sees some encryption of statistically indistinguishable bit string.

Remark 5. In the masked DH-EKE variant, which is proved in [10] in the ROM only, the encryption primitive is a mask generation function instead of an ideal cipher; the Diffie-Hellman values sent are masked by addition with a full-range hash of the password. Here a similar problem arises: the hash needs to be a ROM into elliptic curves. It is possible to use the [12] ROM construction, which is based on Admissible Encoding, to hash the password. But in that case the elliptic curves parameters will not be kept hidden as the resulting ciphertexts are points on the curve.

Remark 6 (Eavesdroppers without the Elliptic Curve Parameters). The family of DH-EKE protocol is secure against offline dictionary attacks under the CDH assumption: an adversary has to compute $K = \mathsf{CDH}_G(G_1, G_2)$ to get some information on the password. Indeed, based on CDH and the ROM, the distribution of G_1, G_2, T_D, T_R is computationally indistinguishable from the uniform distribution over $E_{a,b}^2 \times \{0,1\}^{2l}$. Using this property and the property of the admissible encoding (cf. Definition 2), we know that l_1 and l_2 are bit strings computationally indistinguishable from random ones. In the ideal cipher model, this implies that the z_1, z_2, T_D, T_R are indistinguishable from random strings as well. For this reason, an adversary who does not know the elliptic curve parameters, cannot compute them, even if he has a list of curves parameters.

5 Our Proposal of Password Based EC-DH Key Exchange without Encryption

In the EC-DH-EKE scheme (Figure 2), we use the admissible representation in order to encrypt properly elliptic curves points. As an additional benefit, this protocol also ensures the privacy of the elliptic curve parameters. Following this last idea, we modify further the EC-DH-EKE protocol in order to base the authentication directly on the knowledge of the elliptic curve parameters instead of the knowledge of an additional password.

Our proposal is similar to our EC-DH-EKE variant: points are represented by an admissible representation. But we did not encrypt the representations anymore. Since the distribution of l_1, l_2, T_D, T_R is computationally indistinguishable from the uniform distribution, exchanging these values in clear makes no difference from an eavesdropper point of view. This enables to avoid the use of an ideal cipher in the security analysis. In the sequel, we denote our scheme EC-DH-ARKE which stands for Elliptic Curve Diffie-Hellman Admissibly Represented Key Exchange.

In our scheme, the dictionary of passwords becomes a set of different elliptic curves parameters indexed by a table. This table is not secret. Before an authentication, the device holder typesets an index over the reader, the latter can then verifies whether this index corresponds to the stored elliptic curve parameters.

5.1 Parameters

Let k be a security parameter and N a polynomial integer in k. Let $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$ be 3 hash functions with $\{0,1\}^l$ as range.

Let $\mathbb{F}_{2^n}$ be a field such that there exist efficient admissible encodings and such that $2^n = \mathcal{O}(2^{2k})$. Let $S = \{a_i, b_i, G_i, q_i\}_{i \in [1,N]}$ be a set of elliptic curve parameters such that G_i is a generator of the prime order group of $E_i = E_{a_i,b_i}$ of order q_i and let F_i be the associated admissible encoding. It is assumed that the cofactor is the same (2) for each group of points. We also assume that the prime integers q_i are pair-wise distinct. This last condition is sufficient to ensure that no isomorphism exists between any pair (E_i, E_j) with $i \neq j$.

5.2 The EC-DH-ARKE Protocol

During the initialization phase, each reader receives the set S as input and each device receives one element of S as parameters. It can further define its own public discrete logarithm based pair of public/secret keys with these parameters. The index i related to these parameters is given to the device owner. We stress that the set S does not need to remain secret. We use the index in order to enable a user to typeset data related to the parameter.

At the beginning of each authentication, the device holder has to typeset one index and then the reader verifies that the index corresponds to the elliptic curve parameters used by the device. The protocol is illustrated in Figure 3.

Device		*Reader*
password : $E_{a_\pi,b_\pi}, q_\pi, G_\pi$		password : $E_{a_\pi,b_\pi}, q_\pi, G_\pi$
Pick α		Pick β
Compute $G_1 = \alpha \cdot G_\pi$		Compute $G_2 = \beta \cdot G_\pi$
Compute $l_1 = \mathcal{I}_{F_\pi}(G_1)$		Compute $l_2 = \mathcal{I}_{F_\pi}(G_2)$
	$\xrightarrow{l_1}$	
	$\xleftarrow{l_2}$	
Compute $K = \alpha \cdot G_2$		Compute $K = \beta \cdot G_1$
$= \alpha \cdot F_\pi(l_2)$		$= \beta \cdot F_\pi(l_1)$
Compute $\mathcal{K} = \mathcal{H}_0(K, l_1, l_2)$		Compute $\mathcal{K} = \mathcal{H}_0(K, l_1, l_2)$
Compute $K_{\mathsf{enc}} = \mathcal{H}_1(\mathcal{K}, 1)$		Compute $K_{\mathsf{enc}} = \mathcal{H}_1(\mathcal{K}, 1)$
Compute $K_{\mathsf{mac}} = \mathcal{H}_1(\mathcal{K}, 2)$		Compute $K_{\mathsf{mac}} = \mathcal{H}_1(\mathcal{K}, 2)$
Compute $T_D = \mathcal{H}_2(K_{\mathsf{mac}}, l_2)$		Compute $T_R = \mathcal{H}_2(K_{\mathsf{mac}}, l_1)$
	$\xrightarrow{T_D}$	
	$\xleftarrow{T_R}$	
Abort if T_R invalid		Abort if T_D invalid

Fig. 3. Our proposal EC-DH-ARKE

5.3 Security Result

Our proposal is secure in the random oracle model under the AET-CDH assumption. More concretely:

Theorem 1 (AKE security). *Let S be a randomly chosen set of N elliptic curve parameters as above. Let π be an uniformly chosen index in $[1, N]$. Assume that $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathcal{H}_2$ are random oracles. Let $\mathcal{A}$ be an adversary in the BPR model against the AKE security of our scheme within a time T, with less than q_s interactions with the parties, q_p eavesdroppings and q_h hash queries. We have:*

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{EC-DH-ARKE}}(\mathcal{A}) \leq q_s \mathsf{Succ}^{q_h-\mathsf{paet-cdh}}(T') + q_p \mathsf{Succ}^{\mathsf{cdh}}(T') + \varepsilon$$

where $T' = T + \mathcal{O}(Q^2)$, where $Q = q_s + q_h + q_p$ and ε is negligible if q_s, q_p and q_h are polynomial in k.

The security of this protocol relies on two ideas:

1. a passive eavesdropper does not get any information on the exchanged data whenever the CDH is a hard problem for any curve in S;
2. an active adversary can find the password by an online dictionary attack with a probability $1/N$. In fact, an adversary can always be turned into an algorithm, which solves the PAET-CDH problem with almost the same probability.

Remark 7. By definition of an admissible encoding, the inversion algorithm runs in probabilistic time (see Definition 2), thus the implementation of the protocol could result on a non-constant execution runtime (that is depending of the curve used, i.e. of the password); which in a practical application may represent a privacy risk. In [11], we explain that a polynomial adversary cannot exploit this information to distinguish two curves.

5.4 Security Proof

Proof. We use a sequence of game in order to prove the security of the protocol. In the sequel $\Pr[G_i]$ denotes the probability in the game G_i that the adversary outputs the good guess $b' = b$.

Game G_0: This is the real security game. A set of N parameters is chosen, the device receives one element of S and the reader receives the same, while the set S is given to the adversary. The reader and the device act as described in Figure 3. We assume that $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$ are random oracles into $\{0,1\}^l$.

Once the TEST query is sent, following a randomly chosen bit b, the key K_{enc} or a random string is returned. Hence

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{EC-DH-ARKE}}(\mathcal{A}) = |\Pr[G_0] - 1/2|$$

Game G_1: We simulate the device and the reader for each query to the SEND, EXECUTE, TEST and REVEAL oracles, as the real players would do. We also simulate the random oracles $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$. This does not change the adversary advantage but modifies the duration of the simulation because of the necessary table lookups. We thus have $T'[G_1] = T + \mathcal{O}(Q^2)$.

Game G_2: We abort the simulation if a collision occurs while simulating one of the random oracles. A collision occurs with a probability $Q^2/2^{l+1}$ for each random oracle. We thus have:

$$|\Pr[G_2] - \Pr[G_1]| \leq 5 \times \frac{Q^2}{2^{l+1}}$$

From this game, we are almost sure that the values of $\mathcal{K}$ are different. This property is also true for $K_{\mathsf{enc}}, K_{\mathsf{mac}}, T_D, T_R$.

Game G_3: We simulate the EXECUTE oracle using random values. To distinguish this game from the previous one, the adversary needs to solve the CDH problem over a curve for at least one pair (l_1, l_2) exchanged during one of the EXECUTE queries. For this reason, we have:

$$|\Pr[G_3] - \Pr[G_2]| \leq q_p \mathsf{Succ}^{cdh}(T')$$

Game G_4: We abort the simulation when we get a collision on elliptic curve points chosen at the beginning. Since there are q_π points in the curve, we have that:

$$|\Pr[G_4] - \Pr[G_3]| \leq \frac{Q^2}{q_\pi}$$

From this game, we know that the admissible representations returned by the simulation are pair-wise distinct.

Game G_5: We abort when one triplet $(\mathsf{CDH}_{G_\pi}(F_\pi(l_1), F_\pi(l_2)), l_1, l_2)$ is queried a second time to $\mathcal{H}_0$, while l_1, l_2 are values exchanged during one instance I initiated by the query SEND. When this second query to $\mathcal{H}_0$ occurs, we know that the adversary knows the value $\mathcal{K}$ of the instance I. We have assumed that the adversary does not make two identical queries to any random oracle.

Before this abortion, the adversary does not have any advantage over the password π since he has observed random values (due to the admissible representations property, see Section 2.1) that he could not verify, without querying $\mathcal{H}_0$ with a correct query. Once this event happens, we determine l, the value amongst l_1 or l_2, that we sent when we simulated the SEND query. We then get all the triplets queried to the random oracle $\mathcal{H}_0$ by the adversary, which contains l. These triplets form an answer to the PAET-CDH problem. For this reason we have:

$$|\Pr[G_5] - \Pr[G_4]| \leq q_s \mathsf{Succ}^{q_h-\mathsf{paet}-\mathsf{cdh}}(T')$$

Game G_6: We do not use $\mathcal{H}_1$ and $\mathcal{H}_2$ anymore when we simulate the execution of the protocol. For this reason, the REVEAL query does not give any information such as the SEND query concerning T_R and T_D. We thus have:

$$\Pr[G_6] = 1/2$$

Furthermore, since the adversary did not compute $\mathcal{K}$ in any instance, there is no difference from the adversary point of view between G_5 and G_6. So $\Pr[G_5] = \Pr[G_6]$. □

Note that in the above security result, the ROM hypothesis is needed only to simulate the flow part (T_D, T_R), the ROM is not used in the simulation of the first part (key exchange, l_1, l_2) of the protocol. Only our complexity assumptions (cf. Section 3) are necessary for this part. This result holds in the forward secrecy setting (cf. Remark 1) as well.

6 Conclusion and Further Works

This paper describes a new and efficient Password-based Authenticated Key Exchange protocol which is especially adapted for elliptic curves. Particularly, it enables to keep the elliptic curves parameters hidden. In the context of contactless ID documents, this opens a way for implementing realistic solutions which preserve privacy of the owners; especially their nationality. As extensions of this work, a good perspective is to apply our technique on the enhanced versions of EKE which are secure in a more general model (e.g. UC) or with standard assumptions (standard model).

An implementation in a PKI context with the property that the curve parameters remain hidden is also a possible application of our idea. For instance, a smartcard could contain a certified public key which is stored directly in its admissible representation.

Note that the implementation issues are discussed in the characteristic 2 context for simplicity. Applications to characteristic $p > 2$ when p is approximately a power of 2 (i.e. a pseudo-Mersenne prime as for the curves recommended by NIST) are also possible.

Acknowledgements. The authors thank the reviewers for their comments.

References

1. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
2. Barreto, P.: Why public elliptic curves parameters are public, Tales from the Cryptographers (2005), http://www.larc.usp.br/~pbarreto/tales1.html
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
4. Bellare, M., Rogaway, P.: The AuthA protocol for password-based authenticated key exchange. In: IEEE P1363, pp. 136–3 (2000)
5. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: IEEE Symposium on Research in Security and Privacy, pp. 72–84 (1992)
6. Bellovin, S.M., Merritt, M.: Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In: ACM Conference on Computer and Communications Security, pp. 244–250 (1993)
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
8. Boyd, C., Montague, P., Nguyen, K.Q.: Elliptic curve based password authenticated key exchange protocols. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001. LNCS, vol. 2119, pp. 487–501. Springer, Heidelberg (2001)
9. Bresson, E., Chevassut, O., Pointcheval, D.: Security proofs for an efficient password-based key exchange. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM Conference on Computer and Communications Security, pp. 241–250. ACM, New York (2003)
10. Bresson, E., Chevassut, O., Pointcheval, D.: New security results on encrypted key exchange. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 145–158. Springer, Heidelberg (2004)
11. Bringer, J., Chabanne, H., Icart, T.: Password based key exchange with hidden elliptic curve public parameters. Cryptology ePrint Archive, Report 2009/468 (2009), http://eprint.iacr.org/
12. Coron, J.-S., Icart, T.: A random oracle into elliptic curves. Cryptology ePrint Archive, Report 2009/340 (2009), http://eprint.iacr.org/
13. Icart, T.: How to hash into elliptic curves. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 303–316. Springer, Heidelberg (2009)

14. Jablon, D.P.: Strong password-only authenticated key exchange. ACM Computer Communications Review 26, 5–26 (1996)
15. International Civil Aviation Organization. Machine readable travel documents website, http:/mrtd.icao.int/
16. Patel, S.: Number theoretic attacks on secure password schemes. In: IEEE Symposium on Security and Privacy, pp. 236–247. IEEE Computer Society, Los Alamitos (1997)
17. Shallue, A., van de Woestijne, C.: Construction of rational points on elliptic curves over finite fields. In: Hess, F., Pauli, S., Pohst, M.E. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 510–524. Springer, Heidelberg (2006)
18. Ullmann, M., Kugler, D., Neumann, H., Stappert, S., Vogeler, M.: Password authenticated key agreement for contactless smart cards. In: RFIDSec (2008)

Okamoto-Tanaka Revisited: Fully Authenticated Diffie-Hellman with Minimal Overhead*

Rosario Gennaro, Hugo Krawczyk, and Tal Rabin

IBM T.J. Watson Research Center
Hawthorne, New York 10532
rosario@us.ibm.com, hugo@ee.technion.ac.il, talr@us.ibm.com

Abstract. This paper investigates the question of whether a key agreement protocol with the same communication complexity as the original Diffie-Hellman protocol (DHP) (*two* messages with a *single* group element per message), and similar low computational overhead, can achieve forward secrecy against active attackers in a *provable* way. We answer this question in the affirmative by resorting to an old and elegant key agreement protocol: the Okamoto-Tanaka protocol [22]. We analyze a variant of the protocol (denoted mOT) which achieves the above goal. Moreover, due to the identity-based properties of mOT, even the sending of certificates (typical for authenticated DHPs) can be avoided in the protocol.

As additional contributions, we apply our analysis to prove the security of a recent multi-domain extension of the Okamoto-Tanaka protocol by Schridde et al. and show how to adapt mOT to the (non id-based) certificate-based setting.

1 Introduction

Since the invention of the Diffie-Hellman protocol (DHP) [10], much work has been dedicated to armor the protocol against active ("man in the middle") attacks. Designing *authenticated Diffie-Hellman protocols* has proven to be very challenging at the design and analysis level, especially when trying to optimize performance (both computation and communication). This line of work has been important not only from the practical point of view but also for understandings what are the *essential limits* for providing authentication to the DHP.

In particular, it has been shown that one can obtain an *authenticated* DH protocol with the same communication as the basic unauthenticated DHP (at least if one ignores the transmission of public key certificates); namely, a 2-message exchange where each party sends a single DH value, and where the two messages can be sent in any order. A prominent example of such protocols is MQV [18] (and its provably-secure variant HMQV [17]) where the cost of computing a session key is as in the basic unauthenticated DHP plus half the cost of one exponentiation (i.e., one off-line exponentiation and 1.5 on-line exponentiations).

Protocols such as the 2-message MQV are "implicitly-authenticated protocols;" that is, the information transmitted between the parties is computed

* Extended Abstract. Full version available at http://eprint.iacr.org/2010/068.

without access to the parties' long-term secrets while the authentication is accomplished via the computation of the session key that involves the long-term private/public keys of the parties. Unfortunately, implicitly-authenticated protocols, while offering superb performance, are inherently limited in their security against active attackers. Indeed, as shown in [17], such protocols can achieve perfect forward secrecy (PFS) *against passive attackers only.*

Recall that PFS ensures that once a session key derived from a Diffie-Hellman value is erased from memory, there is no way to recover the session key even by an attacker that gains access to the long-term authentication keys of the parties after the session is established. PFS is a major security feature that sets DHPs apart from other key agreement protocols (such as those based in PK encryption) and is the main reason for the extensive use of DHPs in practice (e.g., IPsec and SSH). Adding PFS against active attackers to protocols like MQV requires increased communication in the form of additional messages and/or explicit signatures.

In this paper we investigate the theoretical and practical question of whether the limits of DHPs can be pushed further and obtain a protocol with full security against active attackers (including PFS) while preserving the communication complexity of a basic DHP (*two* messages with a *single* group element per message) and low computational overhead. We answer this question in the affirmative by departing from implicitly authenticated protocols and resorting to an old and elegant key agreement protocol: the Okamoto-Tanaka protocol [22]. We analyze a variant of the protocol (denoted mOT) which achieves the above minimal communication, incurs a negligible computational overhead relative to a basic DHP over an RSA group, and yet achieves *provable* security including *full PFS* against active attackers[1]. Moreover, due to the identity-based [24] properties of mOT, even the sending and verification of certificates is avoided in the protocol.

Our Results. The protocol mOT we analyze is a "stripped down" version of the "affiliation-hiding" key exchange protocol by Jarecki *et al.* [15] (a version of key agreement where parties want to hide who certified their public keys). We remove all the extra steps designed to obtain affiliation-hiding (which is not a concern for our paper) and focus on the 2-message version of the [15] protocol (the latter includes a third message and the transmission of additional authentication information for the parties to confirm they indeed have the same key). We present a rigorous proof of security for mOT in the Canetti-Krawczyk (CK) Key-Agreement Protocol model [5]. The security of the protocol in this model, including weak PFS (i.e., against passive attacks only), is proven in the random oracle model under *the standard RSA assumption.* For the proof of *full* PFS against active attackers (and only for this proof) we resort to non-black-box assumptions in the form of the "knowledge of exponent" assumptions. We stress that our goal is to prove full security (including full PFS) for the 2-message protocol *without* the extra key

[1] There are DH protocols that provide full PFS against active attacks with just two messages, but they require to send (and process) additional information, e.g. explicit signatures [27] or encrypted challenges [16].

confirmation steps: indeed the 3-message protocol with key confirmation can be proven secure (including full PFS) under the standard RSA assumption without requiring extra assumptions (this proof is actually implicit in [15]).

Modified Okamoto-Tanaka (mOT). The modified Okamoto-Tanaka protocol mOT, is described in Figure 1 (for a precise specification see Section 3). We describe the protocol as an identity-based protocol using a KGC (key generation center) as this setting provides added performance advantages to the protocol. Following [15], we include hashing operation on identities as well as the hashing and squaring operations in the computation of the session key K (these steps are not part of the original Okamoto-Tanaka).

The Modified Okamoto-Tanaka (mOT) Protocol

Setting: A Key Generation Center (KGC) chooses RSA parameters $N = pq$ (such that p and q are random safe primes), and exponents d, e, and a random generator g of QR_N, the (cyclic) subgroup of quadratic residues $\bmod N$.

KGC publishes N, e, g, two hash functions H (with range QR_N) and H' (with range of the desired length of the session key), and distributes to each user U with identity id_U a private key $S_U = H(id_U)^d \bmod N$.

Key agreement: A and B choose ephemeral private exponents x and y, respectively.

$$\begin{tikzcd} A \arrow[rr, shift left, "\alpha = g^x S_A \bmod N"] & & B \arrow[ll, shift left, "\beta = g^y S_B \bmod N"] \end{tikzcd}$$

$$\bar{K}_A = (\beta^e / H(id_B))^{2x} \bmod N \qquad \bar{K}_B = (\alpha^e / H(id_A))^{2y} \bmod N$$

$$\bar{K} = \bar{K}_A = \bar{K}_B = g^{2xye} \bmod N$$

A and B set the session key to $K = H'(\bar{K}, id_A, id_B, \alpha, \beta)$

Fig. 1. A and B share session key K. See Section 3 for full details.

Security Proof and Full PFS "for free". The security result that sets our protocol and work apart is our proof of *full PFS* for mOT, namely, *perfect forward secrecy against fully active attackers*. The proof of full PFS (and only this proof) requires two additional "non-black-box" assumptions: one is the well-known KEA1 (knowledge of exponent) assumption [8,1] related to the hardness of the Diffie-Hellman problem and the second is similar in spirit but applies to the discrete logarithm problem (see Section 4). Enjoying full PFS is a major advantage of mOT relative to efficient two-message protocols such as MQV that can only offer weak PFS. Indeed, in spite of mOT transmitting a single group element in each of the two messages, it overcomes the inherent PFS limitations of implicitly authenticated DHPs by involving the sender's private key in the computation of each protocol's message. Most importantly, as we explain below, this *full security against active attackers* is achieved with zero communication and negligible computational overhead relative to the basic DHP. We believe this to be not just a practical feature of mOT but also a significant contribution to the theory of key agreement protocols showing that armoring the original DHP against active attackers can be achieved essentially "for free".

Performance. The cost of mOT remains essentially the same as in the basic (unauthenticated) DHP: one message per party, that can be sent in any order, with each message containing a single group element. No additional authentication information needs to be transmitted. Thanks to the identity-based properties of the protocol, public-key certificates need not be sent or verified. The only extra operation is one exponentiation to the e-th power, which can be chosen to be 3, and one squaring; that is, just three modular multiplications in all. However, note that mOT works over an RSA group and therefore exponentiations are more expensive than over elliptic curves (where protocols like MQV can be run). Yet, we also note that mOT can be implemented with short exponents, say 160-bit exponents when the modulus is of size 1024 (or a 224-bit exponent with a 2048-bit modulus). Our proof of the protocol holds in this case under the common assumption that in the RSA group the discrete logarithm problem remains hard also for these exponent sizes. In terms of practical efficiency, for moderate security parameters (160-200 bit exponents) the cost of one on-line exponentiation in mOT is competitive with the 1.5 exponentiations over elliptic curves required by MQV. For larger security parameters the advantage is fully on the elliptic curve side though in this case one has to also consider the overhead incurred by certificate processing in a protocol like MQV (which is costly especially for ECDSA-signed certificates).

Of course, beyond the practical performance considerations, mOT holds a significant security advantage over 2-message MQV, namely, its full PFS against active attackers. The fact that mOT can do so well with almost no overhead over the underlying basic Diffie-Hellman protocol, and with full security against active attacks, is an important theoretical (and conceptual) aspect of our work pointing to the limits of what is possible in this area.

More discussion on the performance mOT can be found in the full version.

The Need for a Key Generation Center (KGC). As an identity-based protocol, mOT avoids the need for certificates (a significant communication and computational advantage). The id-based setting, however, introduces the need for a KGC that generates and distributes keys to users. This results in a different trust model than the traditional certification authority (CA) that certifies public keys but does not generate or know the private keys of parties. Note, however, that in mOT the private keys are used only for authentication. Thus, while a KGC can impersonate a party, it cannot learn keys exchanged by that party (we note – see full version – that the PFS property holds also against a corrupted KGC). Note that a regular CA can also impersonate parties at will by issuing certificates with the user's name but with a private key known to the CA. Interestingly, as we show below, mOT can be modified to work also in the traditional CA setting.

Further results. In the full version we extend the above proofs and analysis to a recent extension of the Okamoto-Tanaka protocol proposed by Schridde, Smith and Freisleben [25] that allows the execution of the protocol between users that belong to different domains, i.e., to different key generation centers (KGC).

The full version also shows that the mOT protocol and its extension from [25] can be modified to work as "traditional" (i.e., not ID-based) key agreement protocol.

Related work. Key agreement protocols (KAPs) have played an important role in the development of identity-based cryptography, with Okamoto [21], Okamoto and Tanaka [22], Gunther [12] being early examples of id-based cryptography. (Even earlier, the work by Blom [2] on key distribution can be seen as a precursor of id-based schemes.) With the flourishing of pairings-based cryptography, many more id-based KAPs have been designed; yet getting them right has been a challenging task. See the survey by Boyd and Choo [3] and Chen, Cheng, and Smart [6] for good descriptions and accounts of the main properties of many of these protocols. Even to date it seems that very few (e.g., [4,29]) were given full proofs of security (many others were broken or enjoy only a restricted notion of security, such as partial resistance to known-key attacks). In all, the mOT protocol studied here compares very favorably with other id-based and traditional KAPs in provability and security properties (e.g., PFS) as well as performance-wise.

We already discussed the relationship of our work with [15]. We stress again that the security analysis there is for the protocol with the extra key confirmation messages, while we analyze the minimalistic 2-message protocol in Figure 1.

Multi-domain extensions of id-based KAPs have been proposed in [7,19] but without full proofs of security. The multi-domain extension of the mOT protocol that we fully analyze here is from Schridde et al. [25] which also contains a good discussion of the benefits of multi-domain identity-based protocols.

In general, while interactive authenticated KAPs, especially those authenticated with signatures, can easily accommodate certificates (which a party can send together with its signature), avoiding the need for certificates constitutes a significant practical simplification of many systems. In particular, they provide more convenient solutions for revocation and less management burden [28]. The Okamoto-Tanaka example shows that the identity-based setting can sometimes even improve performance.

Open questions. We believe that the mOT protocol is remarkable for its "minimalism", providing full and provable authenticated key-agreement security (including full PFS) with the same communication and minimal computational overhead relative to the underlying unauthenticated DHP. Yet there are several ways one could hope to improve on this protocol and on our results;

achieving any of these improvements would bring us even closer to the "ultimate" authenticated DHP:

(i) Find a protocol with the same communication/computation/security characteristics as mOT but which works over arbitrary dlog groups (in particular elliptic curves). In this case, the minimalism of mOT would translate into optimal practical performance (even a certificate-based protocol with these properties would be very useful). (ii) Prove the full PFS security of mOT without resorting to non-black-box assumptions (while we believe that proofs under these assumptions carry a very strong evidence of security, using more standard assumptions is obviously desirable). (iii) Improve on mOT by avoiding the vulnerability of the protocol to the exposure of the DH values g^x, g^y or the ephemeral exponents x, y.

2 Preliminaries

Let $SPRIMES(n)$ be the set of n-bit long safe primes. Recall that a prime p is safe if $\frac{p-1}{2}$ is prime. Let $N = pq$ be the product of two random primes in $SPRIMES(n)$; denote $p = 2p' + 1$ and $q = 2q' + 1$. Let e be an integer which is relatively prime to $\phi(N) = 4p'q'$.

We say that the RSA Assumption (with exponent e) holds if for any probabilistic polynomial time adversary $\mathcal{A}$ the probability that $\mathcal{A}$ on input N, e, R, where $R \in_R Z_N^*$, outputs x such that $x^e = R \bmod N$ is negligible in n. The probability of success of $\mathcal{A}$ is taken over the random choices of p, q, R and the coin tosses of $\mathcal{A}$.

Remark (semi-safe primes). For simplicity, we assume that $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes, namely, p' and q' are prime. We can relax this assumption to require that $\gcd(p', q') = 1$ and that neither p' or q' have a prime factor smaller than 2^ℓ for a given security parameter ℓ. With these assumptions we get that QR_N is cyclic and that a random element in QR_N is a generator with overwhelming $(1 - 2^{-\ell})$ probability, two properties that we use in our construction and analysis.

Throughout the paper we use the following well-known result of Shamir [23]:

Lemma 1. *Let N, e, d be RSA parameters and f be an integer relatively prime to e. There is an efficient procedure that given N, e, f (but not d) and a value $(x^f)^d \bmod N$, for $x \in Z_N^*$, computes $x^d \bmod N$.*

The cyclic group QR_N. If N is an RSA modulus product of safe primes, then the subgroup QR_N of quadratic residues in Z_N^* is cyclic of order $p'q'$. Let g be a random generator of QR_N (such generator can be found by squaring a random element in Z_N^* (this algorithm yields a generator with overwhelming probability and the resulting distribution is statistically close to uniform). In protocols and proofs below we are going to generate random elements in the group generated by g according to the uniform distribution and with known exponents (i.e., their dlog to the base g). Such a random element X could be generated by choosing an integer

$x \in [1..p'q']$ uniformly at random and setting $X = g^x \bmod N$. But this option implies knowledge of the factorization of N (knowing the value $p'q'$ is equivalent to factoring N). Parties who do not know the value $p'q'$ can approximate the uniform distribution over $\langle g \rangle$ as follows: generate $x \in [1..\lfloor N/4 \rfloor]$ and set $X = g^x \bmod N$. It is not hard to see (cf. [9]) that if p' and q' are of the same size (as required here) then the uniform distributions over $[1..p'q']$ and $[1..\lfloor N/4 \rfloor]$ are statistically close with an exponentially small gap.

Let g be a random generator of QR_N and let $X = g^x \bmod N$ and $Y = g^y \bmod N$ two random elements in QR_N. We say that the Computational Diffie-Hellman (CDH) Assumption (for N and g) holds if for any probabilistic polynomial time adversary $\mathcal{A}$ the probability that $\mathcal{A}$ on input N, g, X, Y outputs Z such that $Z = g^{xy} \bmod N$ is negligible in n. The probability of success of $\mathcal{A}$ is taken over the random choices of p, q, x, y and the coin tosses of $\mathcal{A}$. We know from [26] that the hardness of factoring N (and therefore the RSA Assumption) implies the CDH Assumption.

3 The Modified Okamoto-Tanaka Protocol

Protocol Setup. A key generation center KGC (for "trusted authority") chooses an RSA key (N, e, d), where N is the product of two safe primes p, q. As usual e, d are such that $ed = 1 \bmod \phi(N)$. The KGC also chooses a random generator g for the subgroup of quadratic residues QR_N. The public key of the KGC is (N, e, g) and its secret key is d.

Two hash functions H, H' are public parameter. The first function H outputs elements in the group generated by g (this can be achieved by setting H to be the square $\bmod N$ of another hash function with range Z_N^*). The second hash function H' outputs k-bit strings, where k is the length of the required session key.

Each user in the system has an identity; for convenience we sometimes associate a name to an identity. For example, Alice will be the name of a party while her identity is denoted id_A. We also denote $A = H(id_A)$ and $B = H(id_B)$ (thus, A, B are elements in QR_N). When Alice requests her secret key from the KGC, she receives the value $S_A = A^d \bmod N$ as her secret key (we can think of this as the KGC's RSA signature on Alice's id).

The key agreement. Alice chooses a random integer x from a set S that we specify below. She then computes $X = g^x \bmod N$ and sends to Bob $\alpha = X \cdot S_A \bmod N$. Bob chooses a random y in S, sets $Y = g^y \bmod N$, and sends to Alice the value $\beta = Y \cdot S_B \bmod N$. Alice computes a shared secret value $\bar{K}_A = (\beta^e/B)^{2x} \bmod N$ while Bob computes it as $\bar{K}_B = (\alpha^e/A)^{2y} \bmod N$. (Alice and Bob check that the incoming value is in Z_N^* or else they abort the session.)

Notice that the values $\bar{K}_A, \bar{K}_B$ are equal:

$$\bar{K}_A = Y^{2xe} \cdot (S_B^e/B)^{2x} = Y^{2xe}(B^{ed}/B)^{2x} = g^{2xye} = ((XS_A)^e/A)^{2y} = \bar{K}_B$$

Alice and Bob set $\bar{K}$ as this shared secret value $\bar{K} = \bar{K}_A = \bar{K}_B = g^{2xye} \bmod N$ and set the session key to $K = H'[\bar{K}, id_A, id_B, \alpha, \beta]$. Since we want both parties to compute the same session key we need to determine an ordering between id_A and id_B, and between α and β in the input to the hash; for example, a lexicographic ordering (note that we are not assuming necessarily that there is a definite role of "initiator" and "responder" in the protocol, and hence we do not use such roles to determine the ordering of the above values).

Protection of ephemeral values. We specify that the ephemeral Diffie-Hellman values X, Y chosen by the parties be given the same protection level as the private keys S_A, S_B (indeed, learning these ephemeral values is equivalent to learning the private keys). In particular, if these values are stored in the (less secure) session state they need to be stored encrypted under a (possibly symmetric) key stored with the private key. (This is analogous to the need for protecting the ephemeral value k in a DSA signature.) In addition, we specify that in the computation of the session key, the hashing of the value $\bar{K}$ be performed in protected memory and only the session key be exported to a session state or application (learning the shared secret $\bar{K}$ value opens some attack venues as explained in the full version).

The exponents set S and performance considerations. The performance cost of the protocol is dominated by the exponentiation operations; hence the choice of the set S from which ephemeral exponents are selected is important. A first choice would be to define S as the interval $[1..\lfloor\sqrt{N}/2\rfloor]$. In this case, as shown in [11] (following the results in [14]), the two distributions

$$\{g^x \text{ for } x \in_R [1..\lfloor N/4 \rfloor]\} \quad \text{and} \quad \{g^x \text{ for } x \in_R [1..\lfloor\sqrt{N}/2\rfloor]\}$$

are computationally indistinguishable under the assumption that factoring N is hard. Therefore, one can use exponents of length *half the modulus* without any loss in security. However, performance can be significantly improved by setting S to be the set of exponents of length κ, where κ is twice the security parameter (e.g., $\kappa = 224$). Indeed in this case, the security of the protocol relies on the common assumption that discrete log (over Z_N^*) is hard also when the exponents are of length κ. Indeed (see Lemma 3.6 in [11]), this assumption implies that the two distributions

$$\{g^x \text{ for } x \in_R [1..\lfloor N/4 \rfloor]\} \quad \text{and} \quad \{g^x \text{ for } x \in_R [1..2^\kappa]\}$$

are computationally indistinguishable, and hence using the short or long exponents is equivalent. Therefore, we recommend the protocol to be implemented using short exponents; in particular, we use this case when discussing the protocol's performance.

3.1 Proof of the mOT Protocol

We prove first the following theorem showing the basic security of the mOT protocol. In the full paper we prove further security properties, namely, resistance to KCI and to reflection attacks, and weak PFS. We defer the proof of full PFS (which requires a more involved proof and additional assumptions) to Section 4.

Theorem 1. *Under the RSA assumption, if we model H, H' as random oracles, the* mOT *protocol is a secure identity-based key agreement protocol.*

Proof. The proof is carried in the Canetti-Krawczyk Key-Agreement security model [5] (a succinct summary of this model is presented in the full version) and it follows a typical simulation/reduction argument: We assume an efficient KA-attacker $\mathcal{M}$ that breaks the security of the mOT protocol and use it to build an algorithm that inverts RSA on random inputs.

We start by noting the following fact about an attacker against mOT: Since the session identifier $(id_A, id_B, \alpha, \beta)$ is hashed together with the shared secret value $\bar{K}$ to obtain the session key K, we know that two different sessions necessarily correspond to two different session keys. Moreover, since the hash function H' is modelled as a random oracle then the only way for the attacker to calculate, identify, or distinguish a session key is by computing the value $\bar{K}$ and explicitly querying it from H'.[2]

We call the algorithm that we build for inverting RSA a "simulator" (denoted SIM) since it works by simulating a run of the mOT protocol against the KA-attacker $\mathcal{M}$ which is assumed to win the test-session game with non-negligible probability.

Input to SIM. The input to SIM is a triple (N, e, R) where N, e are chosen with the same RSA distribution as used in the mOT protocol and R is a random element in Z_N^*. The goal of SIM is to output $R^d \bmod N$ where d is such that $ed = 1 \bmod \phi(N)$.

Some conventions: We often omit the notation "mod$N$" when operating in the group Z_N^*. When saying that we chose a random element u from QR_N we mean choosing $v \in_R Z_N^*$ and setting $u = v^2 \bmod N$. To choose a value u in Z_N^* with a known "RSA signature" $s = u^d$, we first choose s and then set $u = s^e$. If $U = g^u, W = g^w$ are elements of QR_N we denote with $DH_g(U, W)$ the value g^{uw}, i.e. the result of the Diffie-Hellman transform in base g applied to U, W.

SIM runs a virtual execution of the mOT protocol (consisting of multiple sessions) against the attacker $\mathcal{M}$, simulating all protocol actions, including the determination of private keys and responses to queries to the functions H, H' made by $\mathcal{M}$. In particular, we allow SIM to "program" the hash functions H, H' (as long as outputs are chosen independently of each other and with uniform distribution) as is customary when modeling H, H' as random oracles.

Identities and keys. Each participant in the protocol has an identity, id_P, possibly chosen by $\mathcal{M}$; we denote $P = H(id_P)$. Of all party identities participating in the protocol, SIM chooses one at random; we denote it by id_B and will refer to this party as Bob. For each participant id_P *other than Bob*, SIM chooses a

[2] Note that if parties A and B have a session where they exchanged messages α (from A to B) and β (from B to A), and another session where the same messages were exchanged but in the reverse direction, both sessions will have the same key. However, as long as one of A and B is honest, each session will have at least one fresh message (except for the negligible probability that two random values in QR_N coincide), hence the above cannot happen even with the attacker's intervention (who can only choose one message in each session).

random value $p \in_R QR_N$ and sets $P = H(id_P) = p^e$. In this way, SIM also knows the private key of the participant, i.e., $S_P = P^d = p$ (note that P and S_P are elements of QR_N). For Bob, SIM sets $H(id_B) = B = R^2 \bmod N$, where R is the input to SIM (note that R^2 is random in QR_N).

Choosing a QR_N generator. SIM sets the random generator g of QR_N to be used in the protocol as following: it chooses random $\bar{r} \in_R QR_N$, sets $r = \bar{r}^e$, and $g = (rB)^e$. Note that with these choices $B = g^d/r$ and $\bar{r} = r^d$; also note that g and B are random in QR_N and independent.

Guessed test session. Before starting the simulation of session establishments, SIM chooses at random a (future) session that it conjectures will be chosen by $\mathcal{M}$ as the test session. SIM does so by guessing the holder of the test session among all the parties in the orchestrated protocol run (we refer to this party as Alice) and guessing the order number of the session among all of Alice's sessions. This allows SIM to know when the guessed session is activated at Alice in the protocol's run. In addition, SIM also guesses that the peer to the test session will be Bob (defined above). We specify that, if at any point in SIM's simulation, it is determined from the protocol's run that the guessed session is not to be chosen as the real test session by $\mathcal{M}$ (e.g., if either Alice or Bob are corrupted, or another test session is chosen by $\mathcal{M}$, etc.) the simulator aborts. The probability that the guessed session will actually be chosen by $\mathcal{M}$ as the test session is non-negligible (as long as the simulation of the protocol by SIM is correct).

Session Interactions (non-test sessions). Attacker $\mathcal{M}$ can choose to initiate and schedule sessions between any two participants and can input its own values into the various sessions, either by utilizing corrupted players or by delivering messages allegedly coming from honest parties. The simulator SIM needs to act on behalf of honest parties in these interactions. Simulating the actions of any uncorrupted party other than Bob is simple for SIM, as it knows their private keys and can choose their ephemeral exponents. Sessions in which Bob is a participant are more problematic since SIM does not know Bob's private key $S_B = B^d$. Whenever Bob is activated in a session, SIM will set the value $\beta = g^b/\bar{r}$ as the outgoing message from Bob where $b \in_R [1..\lfloor N/4 \rfloor]$ is chosen afresh with each activation of Bob and the value $\bar{r}$ is fixed and defined above. Clearly, β is distributed uniformly over QR_N as in the real runs of mOT. While SIM cannot compute session keys with such choice of β we will still see that it can answer the attacker's session-key queries.

Response to party corruption and session key queries (non-test sessions). If at any point $\mathcal{M}$ corrupts a party, SIM provides all information for that party including the private key (which SIM knows). Note that if the attacker asks to corrupt Bob, SIM aborts since it is a sign that SIM did not guess correctly the test session. Session-key queries for sessions where one of the messages was generated by an honest party other than Bob, can be answered by SIM who chooses the ephemeral exponent for the session. The problematic cases are sessions where Bob is a peer and for which the incoming message to Bob was chosen by the attacker (rather than by SIM itself), and provided to Bob as coming from some party id_C (we refer to it as Charlie), which may be honest or corrupted, but different than Bob[3]. In this case SIM does not know the ephemeral exponents of either party to the session so it cannot compute the session key. Instead the simulation proceeds as follows.

The idea is that as long as $\mathcal{M}$ does not query the session value $\bar{K}$ from the random oracle H', then SIM can answer the session key query with a random value. However, if $\mathcal{M}$ does know the value $\bar{K}$, and it actually queries H' on this value, then SIM needs to answer consistently. Specifically, we are dealing with a session where the peers are Bob and Charlie, whose hashed identities are $B = H(id_B)$ and $C = H(id_C)$, respectively, and the exchanged values are γ, chosen by the attacker, and β chosen by SIM as specified above. Thus, the session key is $H'(\bar{K}, id_C, id_B, \gamma, \beta)$ for the appropriately computed $\bar{K}$. Before answering the session-key query, SIM needs to check whether an input of the form $(Q, id_C, id_B, \gamma, \beta)$ was queried from H' where $Q = \bar{K}$. If such a query with $Q = \bar{K}$ was indeed performed then SIM will answer the session-key query with the existing value $H'(Q, id_C, id_B, \gamma, \beta)$. If not, SIM will choose a random value ρ in the range of H' and will return ρ as the value of the session key.

The main question is how will SIM verify whether $Q = \bar{K}$ for a prior query. The value $\bar{K}$ can be represented as $DH_g(Z^2, \beta^e/B)$ for $Z = \gamma/S_C$. By our choice of $B = g^d/r$, $\beta = g^b/\bar{r}$ and $r = \bar{r}^e$, we have that $\beta^e/B = (g^{eb}\bar{r}^{-e})/(g^d r^{-1}) = g^{eb-d}$ and therefore,

$$\bar{K} = DH_g(Z^2, \beta^e/B) = DH_g(Z^2, g^{eb-d}) = Z^{2(eb-d)} \tag{1}$$

Now, since exponentiation to the e is a permutation over Z_N^*, we have that $Q = \bar{K}$ if and only if $Q^e = \bar{K}^e$, and by Equation (1) this is the case if and only if $Q^e = (Z^{2(eb-d)})^e = Z^{2(e^2b-1)}$. But this last computation can be performed by SIM who knows all the involved values, including b that SIM chose and Z (since $Z = \gamma/S_C$ and SIM knows both γ and S_C).[4]

Simulating the test session. When $\mathcal{M}$ activates the session at Alice that SIM chose as its guess for the test session, SIM acts as follows. Let the identity of Alice be id_A and denote $A = H(id_A)$. Since Alice is assumed to be the holder of the test session, it means that it is Alice (or SIM in our case) who chooses the outgoing message α from the session, not the attacker. SIM sets this message to the value $\alpha = (rB)^f S_A$, where r, B are as described at the begining of the simulation, $S_A = A^d$ is Alice's private key (which SIM knows) and f is chosen as $f = te+1$ for $t \in_R [1..\lfloor N/4 \rfloor]$. With this choice, α's distribution is statistically close to uniform over QR_N. Indeed, we have $(rB)^{te+1} = (g^d)^{te+1} = g^d g^t$ with

[3] We assume for the time being that Bob does not run a KA session with itself (thus $C \neq B$). The case where both session peers have the same identity is called a reflection attack and is proved in the full version.

[4] We note for future reference, that knowing S_C is not strictly necessary for SIM to carry this simulation step. If SIM does not know S_C (as in some other proofs in this paper) it does not know Z either. Instead SIM will use $Z^e = \gamma^e/C$ which it does know, and instead of checking $Q^e = Z^{2(e^2b-1)}$ it will check the equivalent $Q^{e^2} = (Z^e)^{2(e^2b-1)}$.

$t \in_R [1..\lfloor N/4 \rfloor]$ (recall that in the real protocol Alice chooses $\alpha = g^x S_A$ with g^x also statistically close to uniform distribution over QR_N). It also makes it independent of other values in the protocol including B and β.

The peer to the test session is Bob (or else SIM aborts) and the incoming message is denoted by β. This value can be chosen by the attacker (which delivers it to Alice as coming from Bob) or by Bob itself. In the latter case, β is chosen by SIM as described above for other sessions activated at Bob. In case $\mathcal{M}$ chooses β, it can be any arbitrary value. Below, we make no assumption on β other than being in Z_N^*. The session key in this case is $K = H'(\bar{K}, id_A, id_B, \alpha, \beta)$ where $\bar{K}$ is computed as follows: if $X = g^x$ denotes the value α/S_A then $\bar{K} = (\beta^e/B)^{2x}$. Now, by our choice of parameters α, g, B, r (in particular, $rB = g^d$), we have that $X = \alpha/S_A = (rB)^f = (g^d)^f$ and hence $x = df \bmod \phi(N)/4$. Thus,

$$\bar{K} = (\beta^e/B)^{2df} \tag{2}$$

Since SIM cannot compute this value (it does not know the ephemeral exponent of either peer to the session) we need to show how SIM responds to a test-session query (assuming the guessed session is indeed chosen by $\mathcal{M}$ as the test session). Upon such a query, SIM will check if there was any query made to H of the form $(Q, id_A, id_B, \alpha, \beta)$ and if so, it will check if Q equals the session value $\bar{K}$. This is done by checking whether $Q^e = (\beta^e/B)^{2f}$ (which involves values known to SIM). Indeed, note that $Q = \bar{K}$ if and only if $Q^e = \bar{K}^e$ (as exponentiation to e is a permutation) and using Equation (2) we have $\bar{K}^e = (\beta^e/B)^{2f}$. If SIM identifies such a Q, SIM has learned $\bar{K}$ from which it can compute its target RSA forgery as we explain below. If not, SIM responds to the session-key query with a random value. From now on, it monitors $\mathcal{M}$'s queries to H to see if a $Q = \bar{K}$ is identified, in which case SIM learns the session key and outputs the forgery.

Computing the forgery R^d. The goal of SIM is to compute $R^d \bmod N$ where $R \in_R Z_N^*$ was given to SIM as input. We now show that whenever SIM learns the session key corresponding to the test session (as shown above) it can compute R^d. Indeed, it is easy to see from Equation (2) that $(B^{2f})^d = \beta^{2f}/\bar{K}$, and since SIM chose $B = R^2$ then $(R^{4f})^d = \beta^{2f}/\bar{K}$. Using Lemma 1 and the fact that $4f$ is relatively prime to e we derive R^d from $(R^{4f})^d$.

Finally, we note that in order to win the test-session game with non-negligible advantage it must be that $\mathcal{M}$ queries the correct $\bar{K}$ from H with non-negligible probability, then SIM is guaranteed to learn $\bar{K}$, and hence compute R^d, also with non-negligible probability.

In the full version we prove further security properties of mOT , such as resistance to *reflection* and *key compromise* attacks.

4 Proof of the PFS Property of the mOT Protocol

In this section we prove that the Modified Okamoto-Tanaka protocol enjoys full *Perfect Forward Secrecy (PFS)* against active attackers. For this proof we need to resort to two additional assumptions[5] (on top of the RSA Assumption required for the proof of the basic security of the protocol, i.e. Theorem [1]). The first assumption is the well-known "Knowledge of Exponent Assumption" introduced by Damgard [8] (and further used and studied in [13,1]). Intuitively, it states that to compute a DH value g^{xy} out of a triple g, g^x, g^y one has to necessarily know either x or y. We will refer to this assumption (called KEA1 in [1]) as KEA-DH.

Knowledge of Exponent Assumption for Diffie-Hellman (KEA-DH). Let G be a cyclic group, and let g, h be distinct generators of G. The assumption says that for every algorithm $\mathcal{M}$ that on input G, g, h outputs (y, z) where $y = g^x$ and $z = h^x$ for some integer x, there exists an algorithm $\mathcal{M}^*$ which outputs x.

A fully formal statement of the assumption can be found in [1]; in particular, it is assumed that for every set of random coins used by $\mathcal{M}$, if $\mathcal{M}$ outputs $(y = g^x, z = h^x)$ then for the same set of random coins $\mathcal{M}^*$ outputs x.

Our second assumption is close in spirit to KEA-DH but it applies to the discrete logarithm problem; we refer to it as KEA-DL*. The idea is as follows: Under the discrete log assumption, given a pair $(g, B = g^b)$ (for random b) it is hard to find b. But what if in addition to the pair $(g, B = g^b)$ one is also given a dlog oracle where one can input any value in G other than B, and receive its dlog to base g. Obviously, one can find b by querying, for example, the value Bg; more generally, one can query BV where $V = B^i g^j$ for known i, j, and compute b out of the dlog of BV. The KEA-DL* assumption states that if one is allowed a single query to the oracle then the above strategy is the only feasible one. Namely, if an algorithm finds b by querying a single value from the oracle then there is another algorithm that outputs values i, j as above. In addition we also need to assume that the knowledge of the e-th root of B (for a fixed value e) does not help the attacker to find the dlog of B in the above game. More formally[6]:

Modified Knowledge of Exponent Assumption for Discrete Log (KEA-DL*). Let G be the subgroup of Quadratic Residues in Z_N^* where N is an RSA modulus. We modify the **KEA-DL** assumption to allow $\mathcal{M}$ to receive also the e-root of B (where e is an RSA exponent). The modified assumption is as follows:

1. $Chall_{DL^*}$ provides $\mathcal{M}$ with $N, e, g, B = g^b$ where g, B are random quadratic residues in Z_N^*;
2. $\mathcal{M}$ is allowed to query an element $V \in G$;
3. $Chall_{DL^*}$ responds with the discrete log of BV and the e-root of B;
4. $\mathcal{M}$ outputs an integer b'; $\mathcal{M}$ wins if $b = b'$.

[5] It is important to note that, as shown in the full version, weak PFS – i.e. against passive attackers – is a direct consequence of Theorem [1] and does not require additional assumptions.

[6] A fully formal statement of the assumption quantifies over each set of random coins of algorithms $\mathcal{M}$ and $\mathcal{M}*$; the details are similar to the treatment of the KEA-DL* assumption in [1] and are omitted from this extended abstract.

The KEA-DL* assumption states that for every algorithm $\mathcal{M}$ that wins the above game, there exists an algorithm $\mathcal{M}^*$ which outputs integers i, j such that $V = B^i g^j$.

Theorem 2. *Under the RSA,* **KEA-DH** *and* **KEA-DL*** *assumptions, if we model H, H' as random oracles, then the* mOT *protocol enjoys* perfect forward secrecy (PFS) *(against passive and active attackers).*

Proof. The PFS case differs from the non-PFS case, proven in Theorem 1, in that, after completing the test session between Alice and Bob, the attacker is given the values $S_A = A^d$ and $S_B = B^d$, i.e., the private keys of the peers to the session. Only after receiving these values, the adversary needs to distinguish the test key from random.

Recall that due to the fact that the session key is the hash of the resulting secret value $\bar{K} = g^{2xye}$, where the hash is modeled as a random oracle, distinguishing the key is equivalent to finding $\bar{K}$. Thus, in the sequel we assume that a successful attacker is one that guesses this value g^{2xye}.

Examining the proof of Theorem 1 we can see that in all the simulations in that proof, the simulator knows the secret key S_A of Alice, and therefore it can provide it to the adversary upon corruption of Alice.

The difficulty in proving full PFS is the need to provide the attacker $\mathcal{M}$ with $S_B = B^d$ before $\mathcal{M}$ outputs its guess for the session key. This is very different than the case of Theorem 1 where the simulator first receives the attacker's guess and only then it uses this value to compute the forgery B^d. Still, with some significant changes to the simulation and some added assumptions, we will be able to prove the theorem by transforming a successful mOT attacker $\mathcal{M}$ into an RSA forger F that inverts RSA on a random input. We show this reduction now.

The RSA Forger F**.** The forger F is given as input an instance $N, e, R \in_R QR_N$ of the RSA problem and needs to compute $R^d \bmod N$ with non-negligible probability. F starts by running a simulation of the mOT protocol against attacker $\mathcal{M}$ (which we assume to guess the test session key with non-negligible probability). For this F sets up the public parameters of mOT as N, e, g where N, e are from F's input and $g = h^e \bmod N$ for h chosen by F at random in QR_N. As in the proof of Theorem 1, F generates private keys for all parties except Bob[7] by programming the random oracle H (i.e., for each party id_P, F chooses $p \in_R QR_N$ and sets $H(id_P) = P = p^e$, and $S_P = p$).

For Bob, F chooses the value $B = H(id_B)$ by programming H as follows. It sets a value $U = g^u \bmod N$ where F chooses u as a random integer in the range $[1..\lfloor N/2 \rfloor]$ (note that with this choice of u, the distribution of the value U is statistically close to uniform in QR_N – by a similar argument as in Section 2).

[7] Bob is the peer to the test session - as in the previous proof we assume that the simulator successfully guesses the test session and its peers, an event that happens with non-negligible probability.

Then, it flips an unbiased coin *coin*, and sets

$$H(id_B) = B = \begin{cases} R^2 \bmod N & \text{if } \ coin = 0 \quad \text{(where } R \text{ is part of } F\text{'s input)} \\ U^e \bmod N & \text{if } \ coin = 1 \end{cases}$$

Notice that the distribution of B is correct, i.e., (statistically close to) random in QR_N and independent of other values in the protocol. To simulate all the sessions other than the test session, F follows the same simulation as in Theorem 1. In other words, we keep the proof of Theorem 1 intact up to the point in that proof titled "Simulating the test session".

It remains to show how F simulates the test session interaction with $\mathcal{M}$ in the PFS case, and how this simulation results in the computation of R^d. For this we are going to first modify the attacker $\mathcal{M}$ into an attacker $\bar{\mathcal{M}}$ that behaves like $\mathcal{M}$ but, in addition, in runs where $\mathcal{M}$ guesses correctly the test session, $\bar{\mathcal{M}}$ will output some additional values that will allow F to complete its forgery. Thus, F will be running against the modified $\bar{\mathcal{M}}$ rather than against $\mathcal{M}$. We now show how we transform $\mathcal{M}$ into $\bar{\mathcal{M}}$ via several intermediate "games". We start by presenting a first game that represents the interaction with a KA-attacker in the test session experiment in the PFS setting.

The PFS Game. The following game represents the test session interaction between a "PFS challenger", $Chall_{PFS}$, and the KA-attacker $\mathcal{M}$ where $\mathcal{M}$ is allowed to corrupt Bob after the test session key is complete. In this case, $\mathcal{M}$ sends the incoming message β (allegedly coming from Bob) into the test session held by Alice, and Alice outputs a value α. After these values are set, the attacker receives Bob's secret key $S_B = B^d$ and $\mathcal{M}$ wins the game if it outputs $\bar{K} = g^{2exy}$.

PFS Game:

1. $Chall_{PFS}$ sends to $\mathcal{M}$ the values N, e, g, B, X.
   Here X represents the value α sent by Alice; indeed, since we assume that $\mathcal{M}$ can also be given S_A (which F chooses and hence knows) then providing $\alpha = X \cdot S_A$ is equivalent to providing X.
2. The adversary sends a value β which in turn determines a value Y defined as $Y = \beta / B^d$.
   Note that β, which is chosen by $\mathcal{M}$, may not be an element of QR_N, and the same holds for Y. However, $Y^2 \bmod N$ is necessarily in QR_N and hence generated by g. Defining $y = \log_g(Y^2 \bmod N)$, and thanks to the squaring operation in the computation of the session key in mOT, we get that the session key g^{2xye} equals X^{ye}.
3. $Chall_{PFS}$ sends B^d.
4. $\mathcal{M}$ sends $\bar{K}$. The adversary, $\mathcal{M}$, wins if $\bar{K} = X^{ye}$.

We now show how to transform an attacker $\mathcal{M}$ that wins the above PFS game (with non-negligible probability) into a modified attacker $\bar{\mathcal{M}}$ (with the same probability of success) that F will use to compute its RSA forgery. We arrive to $\bar{\mathcal{M}}$ from $\mathcal{M}$ through a series of adversaries $(\mathcal{M}, \mathcal{M}^*, \mathcal{M}_1, \mathcal{M}_1^*, \bar{\mathcal{M}})$ defined in the following games.

The adversary $\mathcal{M}^*$: Note that $\mathcal{M}$, in the above PFS game, can be changed to also output Y^{2e} (computed as β^{2e}/B^2). So in a winning run on input g, X, attacker $\mathcal{M}$ would output $\bar{K} = X^{ye}$ and also $Y^{2e} = g^{ye}$. By invoking the **KEA-DH** assumption there is another attacker $\mathcal{M}^*$ that behaves as $\mathcal{M}$ but in addition it outputs, in winning runs, $ye = \log_g Y^{2e}$ in step 4.

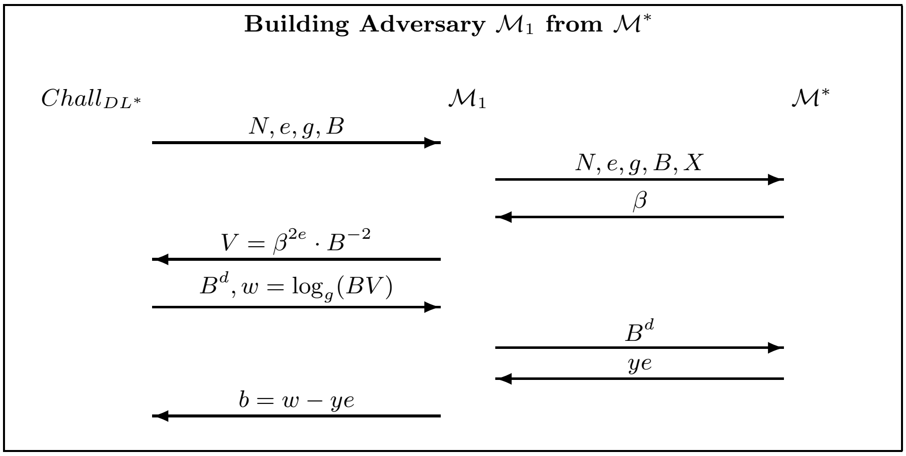


Fig. 2. Creating a **KEA-DL*** adversary

The adversary $\mathcal{M}_1$: We now use the modified PFS attacker $\mathcal{M}^*$ to build an attacker $\mathcal{M}_1$ that interacts with a challenger $Chall_{DL^*}$ in a **KEA-DL*** game. The actions of $\mathcal{M}_1$ are described in Figure 2: $\mathcal{M}_1$ uses $\mathcal{M}^*$ as a subroutine (where $\mathcal{M}_1$ acts as the PFS challenger with respect to $\mathcal{M}^*$) and uses responses received from $\mathcal{M}^*$ to answer $Chall_{DL^*}$ queries. Now, since in a successful run of $\mathcal{M}^*$ the last value ye output by $\mathcal{M}^*$ satisfies $g^{ye} = Y^{2e} = \beta^{2e}/B^2 = V$, and the value w output by $Chall_{DL^*}$ satisfies $w = \log_g BV = b + ye$, then the value $b = w - ye$ answered by $\mathcal{M}_1$ is correct (i.e., $g^b = B$) and $\mathcal{M}_1$ wins the KEA-DL* game. In other words, each successful run of $\mathcal{M}^*$ (which happens with non-negligible probability) induces a successful run of $\mathcal{M}_1$ in the **KEA-DL*** game.

The adversary $\mathcal{M}_1^*$: By the **KEA-DL*** Assumption, there is an adversary $\mathcal{M}_1^*$ that behaves exactly as $\mathcal{M}_1$ except that together with V it also outputs i, j such that $V = B^i g^j$. Replacing $\mathcal{M}_1$ with $\mathcal{M}_1^*$ in Figure 2, we get the same flows except that now the values i, j as above are added to the second flow from $\mathcal{M}_1$ to $Chall_{DL*}$. We refer to this as the *modified game of Figure 2*.

The hybrid adversary $\bar{\mathcal{M}}$: Using $\mathcal{M}_1^*$ we build an attacker $\bar{\mathcal{M}}$ that interacts in a PFS game as represented in Figure 3. In the first flow $\bar{\mathcal{M}}$ receives inputs from $Chall_{PFS}$ as in a regular PFS game. Next $\bar{\mathcal{M}}$ uses these inputs to call $\mathcal{M}_1^*$ in the modified game of Figure 2. In this modified game, $\mathcal{M}_1^*$ uses these same values as its first flow to $\mathcal{M}^*$. Upon receiving the β response from $\mathcal{M}^*$,

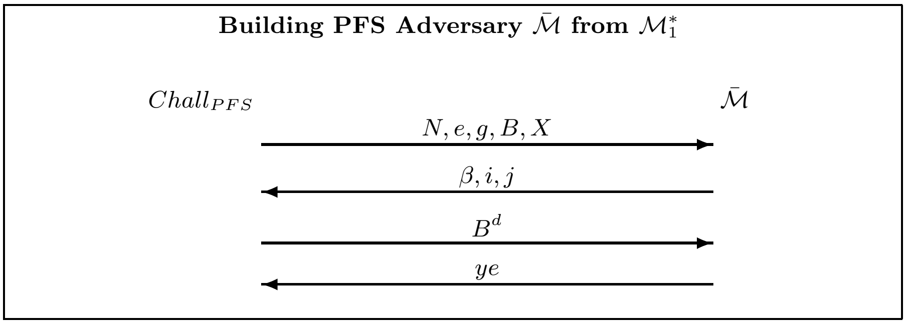


Fig. 3. A Hybrid PFS Adversary $\bar{\mathcal{M}}$ (values in second flow satisfy $\beta^{2e}B^{-2} = B^i g^j$)

$\mathcal{M}_1^*$ produces the value $V = \beta^{2e}B^{-2}$ as well as i, j such that $V = B^i g^j$. Then, $\bar{\mathcal{M}}$ sends to $Chall_{PFS}$ the values β, i, j (note that i, j are not part of the basic PFS game but additional outputs produced by $\bar{\mathcal{M}}$). In the next flow, $\bar{\mathcal{M}}$ receives from $Chall_{PFS}$ the value B^d as in a PFS game, which $\bar{\mathcal{M}}$ uses as the third flow from $\mathcal{M}_1^*$ to $\mathcal{M}^*$. Finally, upon receiving the response ye from $\mathcal{M}^*$, $\bar{\mathcal{M}}$ outputs this same value as its fourth message in the PFS game.

It is important to observe that the flows involving the value w produced by the dlog oracle in Figure 2 are *not* used by $\bar{\mathcal{M}}$; this is important since in a real execution of the PFS game – as part of the interaction with a mOT attacker – there is no such dlog oracle (this only exists as an artifact of the proof).

We note that in a successful run of $\bar{\mathcal{M}}$, the values β, i, j, ye output by $\bar{\mathcal{M}}$ satisfy the following equations:

$$g^{ye} = Y^{2e} = (\beta^e/B)^2 \bmod N \tag{3}$$

$$\beta^{2e}B^{-2} = V = B^i g^j \bmod N \tag{4}$$

which yield

$$g^{ye} = B^i g^j \bmod N \tag{5}$$

Also note that by virtue of the above sequence of games, if $\mathcal{M}$ succeeds with non-negligible probability then $\bar{\mathcal{M}}$ succeeds in outputting the above values also with non-negligible probability.

Summarizing the above transformations, we showed that the existence of a successful PFS attacker $\mathcal{M}$ implies (via the KEA assumptions) the existence of a second PFS attacker $\bar{\mathcal{M}}$ that in addition to the normal outputs of $\mathcal{M}$ (i.e., β and the session key guess), it also outputs the values i, j, ye that satisfy the above equations.

Applying the above to the key agreement setting of mOT we get that one can take a KA-attacker $\mathcal{M}$ that successfully breaks the PFS property of mOT and transform it into an equivalent KA-attacker $\bar{\mathcal{M}}$, that differs from $\mathcal{M}$ in that it also outputs the above values i, j, ye during a successful interaction in the test session experiment. We use this modified KA-attacker $\bar{\mathcal{M}}$ to complete the description of the forger F.

Back to the RSA Forger F. We have seen before how F simulates a run of mOT against a KA-attacker $\mathcal{M}$ except for the test session interaction. Here we complete the description of this part of the simulation and show how F computes its RSA forgery. For this we consider a run of F (as described so far) against the modified KA-adversary $\bar{\mathcal{M}}$ defined above; that is, $\bar{\mathcal{M}}$ behaves exactly as $\mathcal{M}$ but, in successful runs of $\mathcal{M}$, in addition to β it outputs i, j and in addition to the guess of the session key it outputs ye.

The way F uses $\bar{\mathcal{M}}$ to generate a forgery depends on two values: whether i output by $\bar{\mathcal{M}}$ is 0 or not, and whether the *coin* chosen by F (see above) is 0 or 1. We now show these different cases.

Case $i = 0$. With probability 1/2 we also have that $coin = 0$ and therefore $B = R^2 \bmod N$. In this case, F is running $\bar{\mathcal{M}}$ on $g = h^e$, $B = R^2$ and $X \in_R QR_N$. Assuming the run of $\bar{\mathcal{M}}$ is successful and $i = 0$, the forger F obtains the value $j = ye \bmod p'q'$ in the second message from $\bar{\mathcal{M}}$ in Figure 3 (recall that by (5) we have $ye = ib + j \bmod p'q'$). F then uses j to compute Y^2 as follows:

$$h^j = h^{ye} = (g^d)^{ye} = g^y = Y^2 \bmod N,$$

and uses Y^2 to compute $(R^4)^d = B^{2d} = (\beta^2/Y^2) \bmod N$. Using Lemma 1 and the fact that 4 and e are relatively prime F can compute, out of the value $(R^4)^d$, the required forgery $R^d \bmod N$.

Important: In the above case, F only needs to know j to complete its forgery; therefore it does not need to complete the third and fourth flows in Figure 3 (that involve B^d) before it can forge.

Case $i \neq 0$. With probability 1/2 we also have that $coin = 1$ and therefore $B = U^e = g^{ue} \bmod N$ where, by definition, u is a random integer in the range $[1..\lfloor N/2 \rfloor]$. In this case, F knows $B^d = U$ and hence can complete the full run against $\bar{\mathcal{M}}$ in Figure 3. That is, F runs $\bar{\mathcal{M}}$ sending (g, B, X) in the first message and U in the third message. F receives from $\bar{\mathcal{M}}$ the values i, j in the second message and $y' = ye$ in the last message. F will use i, j, y' to find two values s and t from which it will derive (whp) the factorization of N.

Specifically, F sets $s = ie$ and $t = y' - j$ (these operations are over the integers). It holds (mod $p'q'$) that

$$t = y' - j = ye - j = ib = ieu = su \bmod p'q'$$

where the third equality is from Equation (5) and $b = eu \bmod p'q'$ holds by the choice $B = U^e = g^{ue}$.

Thus, the integer $t - su$ is a multiple of $p'q'$ that F can compute as it knows s, t, u. If this number is non-zero then F factors N (as it is well known, the factorization of N can be found from such a multiple of $p'q'$, e.g. [20]). Now, note that there are at least two values of the integer u that will result in the same element U in QR_N (since $u \in_R [1..\lfloor N/2 \rfloor]$ and $\lfloor N/2 \rfloor \geq 2p'q'$). Since $\bar{\mathcal{M}}$ knows U but not the specific u (indeed, from the value U one cannot learn which of the possible values of u was chosen by F), the probability that t and s (that are derived from $\bar{\mathcal{M}}$ outputs) result in $t - su$ being zero is at most 1/2 (at most one of the u's can satisfy this equation).

In all, we have that if F interacts with a successful run of $\mathcal{M}$ (which implies a successful run of $\bar{\mathcal{M}}$) and $i = 0$ then with probability 1/2 F correctly computes R^d. If the run of $\mathcal{M}$ is successful and $i > 0$ then with probability 1/2 F factors N and hence can produce R^d as well. Since $\mathcal{M}$ (and $\bar{\mathcal{M}}$) succeed with non-negligible probability so does F in computing the forgery $R^d \bmod N$. □

Acknowledgment. Research was sponsored by US Army Research laboratory and the UK Ministry of Defense and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
2. Blom, R.: An optimal class of symmetric key generation systems. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 335–338. Springer, Heidelberg (1985)
3. Boyd, C., Choo, K.-K.R.: Security of two-party identity-based key agreement. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 229–243. Springer, Heidelberg (2005)
4. Boyd, C., Mao, W., Paterson, K.G.: Key Agreement Using Statically Keyed Authenticators. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 248–262. Springer, Heidelberg (2004)
5. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
6. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. Int. J. Inf. Sec. 6(4), 213–241 (2007)
7. Chen, L., Kudla, C.: Identity Based Authenticated Key Agreement Protocols from Pairings. In: 16th IEEE Computer Security Foundations Workshop - CSFW 2003, pp. 219–233. IEEE Computer Society Press, Los Alamitos (2003)
8. Damgård, I.: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
9. De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: STOC '94, pp. 522–533. ACM Press, New York (1994)
10. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Trans. Info. Theor. 22(6), 644–654 (1976)
11. Goldreich, O., Rosen, V.: On the security of modular exponentiation with application to the construction of pseudorandom generators. Journal of Cryptology 16(2), 71–93 (2003)

12. Gunther, C.G.: An Identity-Based Key-Exchange Protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)
13. Hada, S., Tanaka, T.: On the Existence of 3-round Zero-Knowledge Protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 408. Springer, Heidelberg (1998)
14. Hastad, J., Schrift, A., Shamir, A.: The Discrete Logarithm Modulo a Composite Hides O(n) Bits. J. Comput. Syst. Sci. 47(3), 376–404 (1993)
15. Jarecki, S., Kim, J., Tsudik, G.: Beyond Secret Handshakes: Affiliation-Hiding Authenticated Key Exchange. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 352–369. Springer, Heidelberg (2008)
16. Krawczyk, H.: SKEME: A Versatile Secure Key Exchange Mechanism for Internet. In: 1996 Internet Society Symposium on Network and Distributed System Security, NDSS (1996)
17. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
18. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient Protocol for Authenticated Key Agreement. Designs, Codes and Cryptography 28, 119–134 (2003)
19. McCullagh, N., Barreto, P.S.L.M.: A New Two-Party Identity-Based Authenticated Key Agreement. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 262–274. Springer, Heidelberg (2005)
20. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
21. Okamoto, E.: Key Distribution Systems Based on Identification Information. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 194–202. Springer, Heidelberg (1988)
22. Okamoto, E., Tanaka, K.: Key Distribution System Based on Identification Information. IEEE Journal on Selected Areas in Communications 7(4), 481–485 (1989)
23. Shamir, A.: On the Generation of Cryptographically Strong Pseudorandom Sequences. ACM Trans. Comput. Syst. 1(1), 38–44 (1983)
24. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
25. Schridde, C., Smith, M., Freisleben, B.: An Identity-Based Key Agreement Protocol for the Network Layer. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 409–422. Springer, Heidelberg (2008)
26. Shmuely, Z.: Composite Diffie-Hellman Public-Key Generating Systems are Hard to Break, Technical Report 356, CS Dept., Technion, Israel (1985)
27. Shoup, V.: On formal models for secure key exchange (version 4) (November 15, 1999), http://www.shoup.net/papers/
28. Smetters, D.K., Durfee, G.: Domain-based Administration of Identity-Based Cryptosystems for Secure E-Mail and IPSEC. In: SSYM 2003: Proceedings of the 12th Conference on USENIX Security Symposium, Berkeley, CA, USA, p. 15. USENIX Association (2003)
29. Wang, Y.: Efficient Identity-Based and Authenticated Key Agreement Protocol. Cryptology ePrint Archive, Report 2005/108 (2005), http://eprint.iacr.org/2005/108/

Deniable Internet Key Exchange*

Andrew C. Yao[1] and Yunlei Zhao[2]

[1] ITCS, Tsinghua University, Beijing, China
[2] Software School, Fudan University, Shanghai, China
ylzhao@fudan.edu.cn

Abstract. In this work, we develop a family of *non-malleable* and *deniable* Diffie-Hellman key-exchange (DHKE) protocols, named deniable Internet key-exchange (DIKE). The newly developed DIKE protocols are of conceptual clarity, provide much remarkable privacy protection to protocol participants, and are of highly practical (online) efficiency.

For the security of the DIKE protocols, we formulate the notion of *tag-based robust non-malleability* (TBRNM) for DHKE protocols, which ensures robust non-malleability for DHKE protocols against concurrent man-in-the-middle (CMIM) adversaries and particularly implies concurrent forward deniability for both protocol participants. We show that the TBRNM security and the session-key security (SK-security) in accordance with the Canetti-Krawczyk framework are mutually complementary, thus much desirable to have DHKE protocols that enjoy both of them simultaneously. We prove our DIKE protocol indeed satisfies both (privacy preserving) TBRNM security and SK-security (with post-specified peers). The TBRNM analysis is based on a variant of the knowledge-of-exponent assumption (KEA), called concurrent KEA assumption introduced and clarified in this work, which might be of independent interest.

1 Introduction

The Internet Key-Exchange (IKE) protocols [21,22] are the *core* cryptographic protocols to ensure Internet security, which specifies key exchange mechanisms used to establish shared keys for use in the Internet Protocol Security (IPsec) standards [23]. The IPsec and IKE are intended to protect messages communicated in the IP layer, i.e., "layer 3" of ISO-OSI, which process the transmission of messages using the network addresses *possibly without knowing end-user peers' identities*. The IKE and IPsec can in turn be used to offer confidentiality, authentication and privacy for communication protocols in the higher layers of ISO-OSI.

The standard of IKE key-exchange has gone through two generations. The first generation IKEv1 [21] uses public-key encryption as the authentication mechanism, and the IKEv2 [22] uses signatures as the authentication mechanism with the SIGMA protocol [24] serving as the basis.

* This work was supported in part by the National Basic Research Program of China Grant Nos.2007CB807900, 2007CB807901, the National Natural Science Foundation of China Grant Nos.60553001, 60703091, and the QiMingXing Program of Shanghai. Preliminary version of this work appeared in Cryptology ePrint Archive 2007/191.

The IKEv2 protocol is based on DHKE [13], and works in the "post-specified peer" setting [22], where the information of who the other party is does not necessarily exist at the session initiation stage and is learnt by the party only after the protocol run evolves. Actually, this is quite a common case for KE protocols in practice, particularly for the purpose of preserving players' privacy. For example, the key-exchange session may take place with any one of a set of servers sitting behind a (url/ip) address specified in the session activation; Or, a party may respond to a request (for a KE session) coming from a peer that is not willing to reveal its identity over the network and, sometimes, even not to the responder before the latter has authenticated itself (e.g., a roaming mobile user connecting from a temporary address, or a smart-card that authenticates the legitimacy of the card-reader before disclosing its own identity) [7].

For key-exchange protocols, both security and privacy are desired. Among privacy concerns, deniability is an essential privacy property, and has always been a central concern in personal and business communications, with off-the-record communication serving as an essential social and political tool [16,12,14]. Given that many of these interactions now happen over digital media (email, instant messaging, web transactions, virtual private networks), it is of critical importance to provide these communications with "off-the-record" or deniability capability to protocol participants.

A protocol is called *forward* deniable, if it ensures deniability for both the sender and the receiver simultaneously. Forward deniability essentially implies that the protocol is *statistical* zero-knowledge (ZK) [19] for both the sender and the receiver, in the sense that both the view of the sender and that of the receiver can be statistically simulated by an efficient algorithm alone without any interactions.

Whenever deniability of messages is desired, in general, we can just run a forward deniable authentication protocol [16] for each message to be sent. However, the beauty of using forward deniable key-exchange is that if the key-exchange protocol is deniable, then all the transactions (of *public* messages) using the session-key produced by the key-exchange protocol can be deniable (i.e., simulatable) for both the protocol participants. Moreover, for the IKE protocol that is the core cryptographic protocol to ensure Internet security, offering deniability by IKE running at the IP layer within the IPsec standard [23] is much more desirable, because it enables various privacy services to be offered at the higher layers with uncompromised quality. Note that a privacy problem at the IP layer can cause irreparable privacy damage at the application layer. For example, an identity connected to an IP address, if not deniable, certainly nullifies an anonymous property offered by a fancy cryptographic protocol running at the application level. (If deniability is not desired, for some cases, then a non-repudiable proof, e.g., a signature, can always be issued at the application level.)

1.1 Our Contributions

In this work, we develop a family of *non-malleable* [15] and *deniable* DHKE protocols, named deniable Internet key-exchange (DIKE), which adds novelty and new value to the IKE key-exchange standard [21,22] and the SIGMA protocol [24]. The newly developed DIKE protocols are of conceptual clarity, provide much remarkable privacy protection to protocol participants, are of highly practical (online) efficiency, and of well compatibility with the IKEv2 and SIGMA protocols.

For the security of the DIKE protocols, we formulate the notion of *tag-based robust non-malleability* (TBRNM) for Diffie-Hellman key-exchange protocols, which ensures robust non-malleability for DHKE protocols against concurrent man-in-the-middle (CMIM) adversaries. Roughly speaking, TBRNM says that a CMIM adversary can successfully finish a session of a DHKE protocol only if it does know both the secret-key and the DH-exponent corresponding to the public-key and the DH-component alleged and sent by the CMIM adversary for that session. The TBRNM formulation takes security and privacy in an integrity, which particularly implies concurrent forward deniability (actually, concurrent non-malleable statistical zero-knowledge CNMSZK) for both the protocol initiator and the protocol responder. We show that the TBRNM security and the session-key security (SK-security) formulated in the Canetti-Krawczyk framework (CK-framework) [6] are mutually complementary, thus much desirable to have DHKE protocols that enjoy both of them simultaneously. simultaneously.

We prove our DIKE protocol is indeed both TBRNM secure and SK-secure (with post-specified peers). The TBRNM analysis is conducted in the *restricted* random oracle (RO) model introduced by Yung, et al [41], in order to bypass the subtleties of deniability loss for simulation with unrestricted ROs [33,35,41], and is based on a variant of the knowledge-of-exponent assumption (KEA) [9]. In particular, we revisit the KEA assumption, demonstrate and clarify the subtleties and insufficiency of employing the KEA assumption to argue the security of DH-based *interactive* cryptographic protocols when they are run *concurrently in the public-key model* (as is the focus of this work). This motivates us to introduce a new extended KEA assumption, called concurrent KEA (CKEA) assumption. Interestingly, the CKEA assumption can be viewed as the non-black-box counterpart of the gap Diffie-Hellman (GDH) assumption [34], while the original KEA assumption is that of the computational Diffie-Hellman (CDH) assumption. As we shall show, the CKEA-based approach for achieving concurrent non-malleability and deniability can be a useful paradigm in DH-based cryptographic practice, with a reasonable trade-off between practical efficiency and formal provable security. The SK-security (with post-specified peers) of our DIKE protocol is proved under the GDH assumption in the RO model.

2 Preliminaries

If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots ; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots ; r)$. If S is a finite set then $x \leftarrow S$ is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement.

On a system parameter n (also written as 1^n), a function $\mu(\cdot)$ is negligible if for every polynomial $p(\cdot)$, there exists a value N such that for all $n > N$ it holds that $\mu(n) < 1/p(n)$. Let $X = \{X(n,z)\}_{n\in N, z\in\{0,1\}^*}$ and $Y = \{Y(n,z)\}_{n\in N, z\in\{0,1\}^*}$ be distribution ensembles. Then we say that X and Y are computationally (resp., statistically) indistinguishable, if for every probabilistic polynomial-time (resp., any power-unbounded) algorithm D, for all sufficiently large n's, and every $z \in \{0,1\}^*$, $|\Pr[D(n,z,X(n,z)) = 1] - \Pr[D(n,z,Y(n,z)) = 1]|$ is negligible in n.

Let G' be a finite Abelian group of order N, and $G = \langle g \rangle$ be a unique subgroup of G', generated by the generator g, of prime order q which is ensured by requiring $gcd(t, q) = 1$ for $t = N/q$. Denote $Z_q = \{0, 1, \cdots, q-1\}$ and $Z_q^* = \{1, 2, \cdots, q-1\}$, denote by 1_G the identity element of G' and by $G/1_G$ the set of elements of G except 1_G. In the specification of this paper, w.l.o.g., we assume G' is the multiplicative group Z_p^* of order $N = p - 1$ for a large prime p, and G is the unique subgroup of order q for some prime divisor q of $N = p - 1$. Typically, the length of p (i.e., the length of group element for a DL-based cryptographic system), denoted $|p| = n$, is treated as the *system* parameter, and the length of q, denoted $|q| = k$, is treated as the *security* parameter. The value $t = (p - 1)/q$ is called the *cofactor*. The specification can be trivially applicable to the groups based on elliptic curves. In elliptic curve systems, G' is the group of points $E(L)$ on an elliptic curve E defined over a finite field L, and G is a subgroup of $E(L)$ of prime order q. For elliptic curve based groups, the cofactor t is typically very small.

Let $H, H_K : \{0, 1\}^* \rightarrow \{0, 1\}^{|q|}$ be hash functions, which are modeled as random oracles in security analysis. Here, for presentation simplicity, we have assumed H, H_K are of the same output length. In practice, they may be of different output lengths.

Definition 1 (Computational Diffie-Hellman (CDH) assumption). *Let G be a cyclic group of prime order q generated by an element g, for two elements $X = g^x$, $Y = g^y$ in G, where $x, y \in Z_q$, we denote by $CDH(X, Y) = g^{xy \bmod q} \bmod p$ (the mod operation is usually omitted for presentation simplicity). An algorithm is called a* CDH solver for *G if it takes as input pairs of elements (X, Y) (and also a generator g of G) and outputs the value of $CDH(X, Y)$. We say the CDH assumption holds in G if for any probabilistic polynomial-time (PPT) CDH solver, the probability that on a pair (X, Y), for $X, Y \leftarrow G$ (i.e., each of x and y is taken uniformly at random from Z_q), the solver computes the correct value $CDG(X, Y)$ is negligible. The probability is taken over the random coins of the solver, and the choice of X, Y uniformaly at random in G.*

The gap DH assumption (GDH) [34] essentially says that in the group G, computing $CDH(X, Y)$, for $X, Y \leftarrow G$, is strictly harder than deciding if $Z = CDH(U, V)$ for an arbitrary triple $(U, V, Z) \in G^3$.

Definition 2 (Gap Diffie-Hellman (GDH) assumption [34]). *Let G be a cyclic group generated by an element g, and a decision predicate algorithm $\mathcal{O}$ be a (*full*) Decisional Diffie-Hellman (DDH) Oracle for the group G and generator g such that on input (U, V, Z), for* arbitrary *$(U, V) \in G^2$, oracle $\mathcal{O}$ outputs 1 if and only if $Z = CDH(U, V)$. We say the GDH assumption holds in G if for any PPT CDH solver for G, the probability that on a pair of random elements $(X, Y) \leftarrow G$ the solver computes the correct value $CDG(X, Y)$ is negligible, even when the algorithm is provided with the (full) DDH-oracle $\mathcal{O}$ for G. The probability is taken over the random coins of the solver, and the choice of X, Y (each one of them is taken uniformly at random in G).*

Definition 3 (Knowledge-of-Exponent Assumption (KEA) [9,25]). *Let G be a cyclic group of prime order q generated by an element g, and consider algorithms that on input a triple $(g, C = g^c, z)$ output a pair $(Y, Z) \in G^2$, where c is taken uniformly at random from Z_q^* and $z \in \{0, 1\}^*$* is an arbitrary string that is generated independently of C.

Such an algorithm $\mathcal{A}$ is said to be a KEA algorithm if with non-negligible probability (over the choice of g, c and $\mathcal{A}$'s random coins) $\mathcal{A}(g, g^c, z)$ outputs $(Y, Z) \in G^2$ such that $Z = Y^c$. Here, $C = g^c$ is the random challenge to the KEA algorithm $\mathcal{A}$, and z captures the auxiliary input of $\mathcal{A}$ that is independent of the challenge C.

We say that the KEA assumption holds over G, if for every efficient (probabilistic polynomial-time) KEA algorithm $\mathcal{A}$ for G there exists another efficient algorithm $\mathcal{K}$, referred to as the KEA-extractor, for which the following property holds except for a negligible probability: let (g, g^c, z) be an input to $\mathcal{A}$ and ρ a vector of random coins for $\mathcal{A}$ on which $\mathcal{A}$ outputs $(Y, Z = Y^c)$, then on the same inputs and random coins $\mathcal{K}(g, C, z, \rho)$ outputs the triple $(Y, Z = Y^c, y)$ where $Y = g^y$.

The KEA assumption is derived from the CDH assumption, and is a *non-black-box* assumption by nature [1]. Since its introduction in [9], the KEA assumption has been used in a large body of works, particularly in the literature of deniable authentication and key-exchange (e.g., [20,2,1,11,25,10,12,38,39], etc).

3 DIKE Implementation and Advantageous Features

Let $(A = g^a, a)$ (resp., $(X = g^x, x)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of the initiator $\hat{A}$, and $(B = g^b, b)$ (resp., $(Y = g^y, y)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of the responder player $\hat{B}$, where a, x, b, y are taken randomly and independently from Z_q^*.

The deniable Internet key-exchange protocol, for the main model of [21,22,23], is depicted in Figure 1 (page 334), where $CERT_{\hat{A}}$ (resp., $CERT_{\hat{B}}$) is the public-key certificate of $\hat{A}$ (resp., $\hat{B}$) issued by some trusted Certificate Authority (CA) within the underlying public-key infrastructure (PKI), and sid is the session-identifier that is assumed to be set by some "higher layer" protocol that "calls" the KE protocol and ensures no two sessions run at a party are of identical session-identifier [7]. Throughout this work, we assume no proof-of-knowledge/possession (POK/POP) of secret-key is mandated during public- key registration, but the CA will check the non-identity subgroup (i.e., $G/1_G$) membership of registered public-keys. Also, each party checks the $G/1_G$ membership of the DH-component from its peer.

3.1 Some Advantageous Features of DIKE

Our DIKE enjoys remarkable privacy protection for both protocol participants. Note that all authentic messages, $NMZK(\hat{B}, y)$ and $NMZK(b, y)$ (resp., $NMZK(a, x)$), from $\hat{B}$ (resp., $\hat{A}$) can be computed *merely* from its peer's DH-exponent x (resp., y) and one's own public messages; Furthermore, one party sends the authentic messages involving its secret-key only after being convinced that its peer does "know" the corresponding DH-exponent. This ensures forward deniability for both the protocol participants. IKEv2 and SIGMA do not enjoy these privacy properties, due to the underlying signatures used. Note also that the DIKE protocol works in the post-specified-peer setting, and the messages from one party do not bear the information of its peers's ID and public-key.

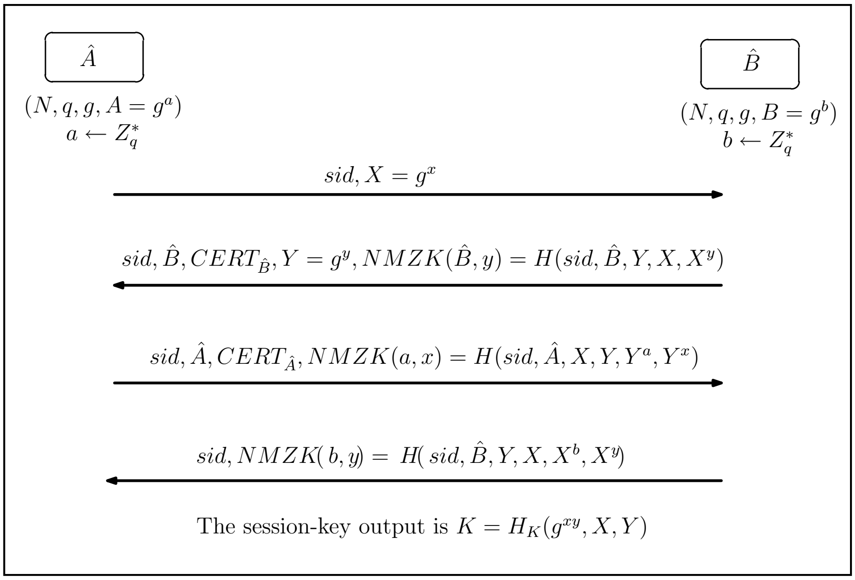

$\hat{A}$
$(N, q, g, A = g^a)$
$a \leftarrow Z_q^*$
$\hat{B}$
$(N, q, g, B = g^b)$
$b \leftarrow Z_q^*$
$sid, X = g^x$
$sid, \hat{B}, CERT_{\hat{B}}, Y = g^y, NMZK(\hat{B}, y) = H(sid, \hat{B}, Y, X, X^y)$
$sid, \hat{A}, CERT_{\hat{A}}, NMZK(a, x) = H(sid, \hat{A}, X, Y, Y^a, Y^x)$
$sid, NMZK(b, y) = H(sid, \hat{B}, Y, X, X^b, X^y)$
The session-key output is $K = H_K(g^{xy}, X, Y)$

Fig. 1. Deniable Internet Key-Exchange (the main model)

Besides some hashing operations and the validation of peer's public-key certificate, the player $\hat{A}$ computes (Y^q, Y^a, Y^x) and (X, B^x), the player $\hat{B}$ computes (X^q, X^b, X^y) and (Y, A^y). Note that the computation of (Y^q, Y^a, Y^x) (resp., (X^q, X^b, X^y)) *in parallel* actually amounts to about 1.5 modular exponentiations. The DH-component X (resp., Y) can always be off-line pre-computed by $\hat{A}$ (resp., $\hat{B}$). Moreover, if the peer's identity is pre-specified, $\hat{A}$ (resp., $\hat{B}$) can further off-line pre-compute the value B^x (resp., A^y). That is, the total computational complexity at each player side is about 3.5 exponentiations, and the on-line computational complexity at each player side can remarkably be only 1.5 exponentiations. We note that if the underlying signatures used in SIGMA are implemented with the Digital Signature Standard (DSS) [17], the computational complexity of SIGMA is about 4.5 exponentiations at each player side in total, and the online complexity is about 2.5 exponentiations (with offline partial signature generation). For communication complexity, by waiving the use and exchanges of signatures, our deniable IKE is of improved communication complexity, in comparison with that of SIGMA.

Our DIKE protocol is of well compatibility with IKEv2/SIGMA and the (H)MQV protocols [26,25,31]. By compatibility with SIGMA/IKEv2, we mean that in case some players are not of discrete logarithm (DL) public-keys, they still can use the Sign-then-MAC mechanism of SIMGA/IKEv2 to authenticate messages from them. In more details, in this case, any one of the last two messages in our deniable IKE can be replaced by the corresponding message flow in SIGMA/IKEv2. By compatibility with (H)MQV, we mean that both (H)MQV and DIKE work for players of DL public-keys, and can be of the same system parameters.

4 Security Formulation and Analysis

In this section, we formulate tag-based robust non-malleability for DHKE protocols based on the CNMZK argument-of-knowledge (CNMZKAOK) formulation [19,36,40], investigate the subtleties of the KEA assumption for arguing the security of DH-based interactive protocols running concurrently in the public-key model and introduces the CKEA assumption, and finally show both the TBRNM security and the SK-security of the DIKE protocol.

4.1 Formulating (Privacy-Preserving) TBRNM for DHKE Protocols

We consider an adversarial setting, where polynomially many instances (i.e., sessions) of a DHKE protocol $\langle \hat{A}, \hat{B} \rangle$ are run concurrently over an asynchronous network like the Internet. To distinguish concurrent sessions, each session run at the side of an uncorrupted player is labeled by a tag, which is the concatenation, in the order of session initiator and then session responder, of players' identities, public-keys, and DH-components available from the session transcript; A session-tag is complete if it consists of a complete set of all these components, e.g., $(\hat{A}, A, X, \hat{B}, B, Y)$. Informally speaking, two sessions are matching if they are of the same session-tag.

We assume all communication channels, among all the concurrent sessions of $\langle \hat{A}, \hat{B} \rangle$, are unauthenticated and controlled by a PPT (CMIM) adversary $\mathscr{A}$. This means that the honest player instances cannot directly communicate with each other, since all communication messages are done through the adversary. All honest player instances are working independently with independent random tapes in different sessions (but with the same public-key), and answer messages sent by $\mathscr{A}$ promptly. Once a session is finished, the honest players always erase the ephemeral (private) state information generated during the session, and only keep in privacy the session key output. Sessions can also be expired, and for expired sessions the session keys are also erased.

The CMIM adversary $\mathscr{A}$ (controlling all communication channels) can do whatever it wishes. In particular, $\mathscr{A}$ can interact with polynomial number of instances of $\hat{A}$ in the name of any player playing the role of the responder; such sessions are called the left-sessions. At the same time, $\mathscr{A}$ can interact with polynomial number of instances of $\hat{B}$ in the name of any player playing the role of the initiator; such sessions are called the right-sessions. For presentation simplicity, we assume the number of left-sessions is equal to that of right-sessions, which is $s(n)$ for some positive polynomial $s(\cdot)$. The adversary $\mathscr{A}$ can decide to simply relay the messages of honest player instances. But, it can also decide to block, delay, divert, or modify messages arbitrarily at its wish.

The CMIM adversary $\mathscr{A}$ also takes some arbitrary auxiliary input $z \in \{0,1\}^*$, which captures arbitrary information collected/eavesdropped by $\mathscr{A}$ over the network from the executions of arbitrary (*possibly different*) protocols prior to its actual session interactions with the instances of $\hat{A}$ or $\hat{B}$. For example, z may consist of a CDH triple (X, B, g^{xb}) that is collected over the Internet where $B = g^b$ is the public-key of the player $\hat{B}$, or just the secret-key b in case the CMIM attacker ever broke in the machine of $\hat{B}$. But, the auxiliary input z, collected prior to the actual session interactions of $\langle \hat{A}, \hat{B} \rangle$, is assumed to be independent of the ephemeral DH-components to be generated and exchanged by the instances of $\hat{A}$ and $\hat{B}$ (specifically, we can consider an experiment where the ephemeral DH-components to be exchanged by the instances of uncorrupted players are generated only after the auxiliary string z is fixed.)

We denote by $view_{\mathscr{A}}(1^n, \hat{A}, A, \hat{B}, B, z)$ the random variable describing the view of $\mathscr{A}$ in its concurrent interactions with the instances of $\hat{A}$ and $\hat{B}$, which includes the input $(1^n, \hat{A}, A, \hat{B}, B, z)$, $\mathscr{A}$'s random tape, and all messages received in the $s(n)$ left sessions and the $s(n)$ right sessions (for protocols in the RO model, $\mathscr{A}$'s view also includes the RO, see [3] for more details). Here, for presentation simplicity, we have assumed that $\mathscr{A}$ concurrently interacts with any polynomial number of instances of two players: one is the initiator player $\hat{A}$ and one is the responder player $\hat{B}$. In a way, the two players $\hat{A}$ and $\hat{B}$ (which can be identical) stand for all uncorrupted players in the system. In general, $\mathscr{A}$ can concurrently interact with any polynomial number of instances of any polynomial number of players. Our definitional framework, as well as the security analysis, can be extended to this general setting, by noting that honest players of different public-keys work independently.

Definition 4 (Tag-based robust non-malleability (TBRNM) for DHKE). *A DHKE protocol, $\langle \hat{A}, \hat{B} \rangle$, is called tag-based robust non-malleable, if for any PPT CMIM adversary $\mathscr{A}$ there exists a PPT simulator/extractor S such that for any sufficiently large n, any pair of uncorrupted players $\hat{A}$ and $\hat{B}$ (of public-key A and B respectively), and any auxiliary string $z \in \{0,1\}^*$, the output of $S(1^n, \hat{A}, A, \hat{B}, B, z)$ consists of two parts (str, sta) such that the following hold, where z captures the arbitrary (possibly public-key dependent) information collected by $\mathscr{A}$ prior to its actual session interactions of $\langle \hat{A}, \hat{B} \rangle$ but is independent of the ephemeral messages (particularly, the DH-components) to be generated and exchanged by the instances of $\hat{A}$ and $\hat{B}$:*

– ***Statistical simulatability.*** *The following ensembles are statistically indistinguishable: $\{view_{\mathscr{A}}(1^n, \hat{A}, A, \hat{B}, B, z)\}_{n\in N, \hat{A}\in\{0,1\}^*, A\in G/1_G, \hat{B}\in\{0,1\}^*, B\in G/1_G, z\in\{0,1\}^*}$ and $\{S_1(1^n, \hat{A}, A, \hat{B}, B, z)\}_{n\in N, \hat{A}\in\{0,1\}^*, A\in G/1_G, \hat{B}\in\{0,1\}^*, B\in G/1_G, z\in\{0,1\}^*}$, where $S_1(1^n, \hat{A}, A, \hat{B}, B, z)$ denotes (the distribution of) the first output of S, i.e., str.*
– ***Knowledge extraction.*** *sta consists of a set of $2s(n)$ strings, $\{\tilde{w}^l_1, \tilde{w}^l_2, \cdots, \tilde{w}^l_{s(n)}, \tilde{w}^r_1, \tilde{w}^r_2, \cdots, \tilde{w}^r_{s(n)}\}$, satisfying the following:*
  - *For any i, $1 \le i \le s(n)$, if the i-th left-session (resp., right-session) in str is aborted or with a tag identical to that of one of the right-sessions (resp., left-sessions), then $\tilde{w}^l_i = \perp$ (resp., $\tilde{w}^r_i = \perp$);*
  - *Otherwise, i.e., the i-th left-session (resp., right-session) in str is successfully completed and is of session-tag different from those of all right-sessions (resp., left-sessions), then $\tilde{w}^l_i = (\tilde{b}^l_i, \tilde{y}^l_i)$ (resp., $\tilde{w}^r_i = (\tilde{a}^r_i, \tilde{x}^r_i)$), where $\tilde{b}^l_i$ (resp., $\tilde{a}^r_i$) is the discrete-logarithm of the public-key $\tilde{B}^l_i$ (resp., $\tilde{A}^r_i$) set and alleged by the CMIM adversary $\mathscr{A}$ for the i-th left-session (resp., right-session) in the name of $\hat{\tilde{B}}^l_i$ (resp., $\hat{\tilde{A}}^r_i$), and $\tilde{y}^l_i$ (resp., $\tilde{x}^r_i$) is the discrete-logarithm of the DH-component $\tilde{Y}^l_i$ (resp., $\tilde{X}^r_i$) set and sent by the CMIM adversary $\mathscr{A}$ in the i-th left-session (resp., right-session).*

Furthermore, we say the DHKE protocol $\langle \hat{A}, \hat{B} \rangle$ is of privacy-preserving TBRNM, if it additionally satisfies: (1) the transcript of each session can be generated merely from *the DH-exponents (along with some public system parameters, e.g., players' public-key and identity information, etc); (2) messages from one party do not bear the identity and public-key information of its peer.*

TBRNM vs. SK-security. We make some brief comparisons between TBRNM and the SK-security in accordance with the CK-framework.

– At a high level, the SK-security essentially says that a party that completes a session has the following guarantees [6]: (1) if the peer to the session is uncorrupted then the session-key is unknown to anyone except this peer; (2) if the *unexposed* peer completes a matching session then the two parties have the same shared key.

  Roughly speaking, besides others, TBRNM ensures the enhanced guarantee of the above (2): if the *possibly malicious* peer completes a matching session, then the two parties, not only, have the same shared key, *but also and more importantly, the (possibly malicious) peer does "*know*" both the DH-exponent (and thus the shared session-key) and the secret-key corresponding to the DH-component and public-key sent and alleged by it in the test-session.* We suggest this kind of security guarantee is very essential to DHKE protocols, particularly when they are run concurrently over the Internet.
– The TBRNM formulation follows the simulation approach [19,36,40] of adaptive tag-based CNMZKAOK, which can actually be viewed as an extended and much strengthened version of the latter. In particular, TBRNM implies concurrent forward deniability for both the protocol initiator and the responder. The SK-security definition follows the indistinguishability approach, which particularly does not take deniability into account.
– Recall that the TBRNM formulation is w.r.t. any PPT CMIM adversary of *arbitrary* auxiliary input. In particular, the adversary's auxiliary input can be dependent on player's public-key, e.g., consisting of a CDH triple (X, B, g^{xb}) or just the secret-key b. That is, the TBRNM formulation implicitly captures the adversarial leakage of static secret-keys of uncorrupted players. Static secret-key exposure *for uncorrupted players* was not captured by the SK-security in [6] (static secret-key exposure and party corruption were separately treated in [6]). But, the TBRNM formulation does not take into account the following abilities of the CMIM adversary in: exposing ephemeral private state for incomplete sessions, exposing session-keys for completed sessions, and party corruption, which are however captured by the SK-security in the CK-framework.

From the above clarifications, the TBRNM security and the SK-security can be viewed mutually complementary, and thus it is much desirable to have DHKE protocols that enjoy both the SK-security and the TBRNM security simultaneously.

4.2 KEA Assumption Revisited, and the CKEA Assumption

Subtleties of employing the KEA assumption in the public-key model. Note that, for the KEA assumption in Definition 3, the requirement of independence between the challenge C and the auxiliary input z plays a critical role. For example, when using KEA for provable security of cryptographic protocols running concurrently in the public-key model, in some cases the challenge C is actually the player's *public-key*. In this case, a valid answer $(A, B = A^c)$, with respect to the challenge C, could be just got by an adversary $\mathscr{A}$ from its auxiliary input that models arbitrary information collected/eavesdropped by $\mathscr{A}$ over the network from executions of other (possibly *different*) protocols *before the interaction of the protocol at hand takes place*. Note that, in this case, it is impossible to efficiently extract the value a from the internal state and auxiliary input of the adversary $\mathscr{A}$. This shows that for protocols with provable security based on the KEA assumption w.r.t. *public* challenges, the independence requirement between the auxiliary input z and the public challenge C (corresponding to player's public-key) can significantly limit the composability of the protocol in practice. In other words, in practice it is unrealistic to assume adversary's auxiliary input to be independence of player's public-keys, particularly for protocol running concurrently in the public-key model. To bypass this subtlety of KEA with public challenges and to render robust composability to cryptographic protocols, in this work we insist using *ephemeral* fresh challenges in designing and analyzing protocols in the public-key model with the KEA assumption.

Subtleties of employing the KEA assumption for interactive *protocols in the* concurrent *setting.* The KEA assumption was originally introduced to argue the (non-malleability) security of public-key encryption (that is a non-interactive cryptographic primitive) [9]. But, when arguing the security of *interactive* protocols running concurrently against CMIM adversaries, we note that, in many scenarios (particularly for DH-based authentication and key-exchange as is the focus of this work), the KEA assumption is insufficient. The reason is that, in such settings, the CMIM adversary can potentially get access to a list of (polynomially many) DDH-oracles, with each being w.r.t. an element taken randomly and independently in G by an honest player instance.

For example, consider a two party protocol $\langle \hat{A}, \hat{B} \rangle$, where $\hat{A}$ generates and sends $X = g^x \in G$ and $\hat{B}$ generates and sends $Y = g^y \in G$; After (or during) the exchange of X and Y, each party uses the shared DH-secret g^{xy} to authenticate some values, and aborts in case the authenticated values from its peer are deemed to be invalid. Now, consider a CMIM adversary who, on a system parameter 1^n, simultaneously interacts with $s(n)$ instances of $\hat{A}$ (by playing the role of $\hat{B}$) and $s(n)$ instances of $\hat{B}$ (by playing the role of $\hat{A}$), where $s(\cdot)$ is a positive polynomial. On an arbitrary value $Z \in G$, a random element X_i generated by $\hat{A}$ (or $\hat{B}$), $1 \leq i \leq s(n)$, and another arbitrary element $Y_j \in G$ where Y_j may also be one of the random elements generated by $\hat{A}$ or $\hat{B}$, the CMIM adversary $\mathscr{A}$ can simply use Z (as the supposed DH-secret) to authenticate a value to the party who sends X_i: if the party aborts, $\mathscr{A}$ concludes $Z \neq CDH(X_i, Y_j)$, otherwise it concludes $Z = CDH(X_i, Y_j)$. This simple protocol example demonstrates that in the concurrent settings for (DH-based) interactive protocols, the CMIM adversary can actually get access to polynomially many DDH-oracles.

The concurrent KEA (CKEA) assumption. The above discussion motivates us to introduce the following assumption, named concurrent knowledge-of-exponents assumption (in reminiscence of the motivation for arguing the *concurrent* security of interactive cryptographic schemes against CMIM adversaries).

Definition 5 (Concurrent knowledge-of-exponents assumption (CKEA)). *Suppose G is a cyclic group of prime order q generated by an element g, 1^n is the system* *parameter, $p(\cdot)$ and $q(\cdot)$ are positive polynomials. Let a decision predicate algorithm $\mathcal{O}_\mathcal{C}$ for $\mathcal{C} = \{C_1 = g^{c_1}, \cdots, C_{p(n)} = g^{c_{p(n)}}\}$ (where c_i, $1 \leq i \leq p(n)$, is taken uniformly at random from Z_q^*) be a DDH-Oracle (w.r.t. the random challenge set $\mathcal{C}$) for the group G and generator g, such that on input (X, Y, Z), for arbitrary $(X, Y) \in G^2$, the oracle $\mathcal{O}_\mathcal{C}$ outputs 1 if and only if $X \in \mathcal{C}$ and $Z = CDH(X, Y)$. Consider algorithms that on input a triple $(g, \mathcal{C}, z)$, with oracle access to $\mathcal{O}_\mathcal{C}$, output a set of triples $\{(X_1, Y_1, Z_1), \cdots, (X_{q(n)}, Y_{q(n)}, Z_{q(n)})\} \subseteq (G^3)^{q(n)}$,* where $z \in \{0,1\}^*$ is an arbitrary string that is generated independently of $\mathcal{C}$. *(Specifically, we can consider an experiment where the DH-components in the set $\mathcal{C}$ are generated only after the auxiliary string z is fixed.) Such an algorithm $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ is said to be a CKEA algorithm if with non-negligible probability (over the choice of $g, c_1, \cdots, c_{p(n)}$ and $\mathcal{A}$'s random coins) $\mathcal{A}(g, \mathcal{C}, z)$ outputs $\{(X_1, Y_1, Z_1), \cdots, (X_{q(n)}, Y_{q(n)}, Z_{q(n)})\} \subseteq (G^3)^{q(n)}$ such that $X_i \in \mathcal{C}$ and $Z_i = CDH(X_i, Y_i)$ for all i, $1 \leq i \leq q(n)$.*

We say that the CKEA assumption holds over G, if for every PPT CKEA-algorithm $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ there exists another efficient PPT algorithm $\mathcal{K}$, referred to as the CKEA-extractor, such that for any polynomials $p(\cdot), q(\cdot)$ and sufficiently large n the following property holds except for a negligible probability: let $(g, \mathcal{C}, z)$ be the input to $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$, ρ a vector of random coins for $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ and ϖ a vector of answers given by $\mathcal{O}_\mathcal{C}$ on queries made by $\mathcal{A}^{\mathcal{O}_\mathcal{C}}$ on which $\mathcal{A}$ outputs $\{(X_1, Y_1, Z_1), \cdots, (X_{q(n)}, Y_{q(n)}, Z_{q(n)})\} \subseteq (G^3)^{q(n)}$ such that $X_i \in \mathcal{C}$ and $Z_i = CDH(X_i, Y_i)$ for all i, $1 \leq i \leq q(n)$, then on the same inputs and random coins and oracle answers $\mathcal{K}(g, \mathcal{C}, z, \rho, \varpi)$ outputs $\{(X_1, Y_1, Z_1, y_1), \cdots, (X_{q(n)}, Y_{q(n)}, Z_{q(n)}, y_{q(n)})\}$ where $Y_i = g^{y_i}$ for all i, $1 \leq i \leq q(n)$.

We note that the CKEA assumption can be viewed as the non-black-box counterpart of the gap Diffie-Hellamn assumption, while the original KEA assumption is that of the CDH assumption. As we shall show in this work, the CKEA assumption is powerful for achieving highly practical cryptographic protocols provably secure against CMIM adversaries in concurrent settings like the Internet. We suggest the CKEA-based approach for achieving concurrent non-malleability can be a useful paradigm in DH-based cryptographic practice, with a reasonable trade-off between practical efficiency and formal provable security.

4.3 Simulation with Restricted RO

When employing the simulation paradigm for proving the security of cryptographic protocols in the RO model, the RO is usually programmed by the simulator (i.e., the simulator provides random answers to RO queries, provided that multiple identical RO-queries are answered with the same answer). But a subtlety here is: simulation with (programmable) RO may lose deniability in general [33,35,41].

To overcome the deniability loss of simulation with programmable RO, the works of [33,35] proposed the unprogrammable RO model where all parties have access to an unprogrammable (fixed) RO. A further investigation, made in [41] (particularly for *interactive* protocols), showed that, in most cases (particularly for the subtleties observed in [33,35,41]), the problem lies in the ability of the simulator in defining (i.e., programming) the RO on queries (first) made by the simulator itself in order to simulate honest parties of private inputs. Specifically, the simulator runs the underlying adversary as a subroutine and mimics honest parties in its simulation. Typically, honest parties (e.g., honest ZK provers) possess some private inputs and get access to un unprogrammable RO in reality; The simulator (in its simulation) has to take the advantage of its ability in programming the RO (to be more precise, programming the RO on queries first made by the simulated honest parties) in order to successfully simulate messages generated by honest parties. But, the simulated honest-party messages may not necessarily be generated with the unprogrammable RO actually accessed by the honest parties in reality. This is precisely the reason for the problems, particularly the loss of deniability, observed in [33,35,41] for simulation with programmable RO.

The work of [41] proposed the *restricted* RO model, where all parties (particularly, all honest parties and the simulator) *except the adversary* get access to an unprogrammable RO but the adversary (who is polynomial-time and possesses no private inputs) is still allowed to access a programmable RO. We can simply view that the restricted RO model is identical to the original RO model, except for that the simulator is confined to programming the RO only on queries first made by the adversary (run by the simulator as its subroutine). Clearly, the restricted RO model is a hybrid of the original programmable RO model [4] and the unprogrammable RO model [33,35]. The restricted RO model allows efficient (interactive) protocol implementations, while still reasonably avoiding the loss of some properties (particularly, deniability) caused by simulation with fully programmable RO.

4.4 Security Results and Overview

For the security of the DIKE protocol (depicted in Figure 1), we prove that it enjoys both the (privacy-preserving) TBRNM security and the SK-security with post-specified peers. Specifically, the DIKE protocol is *privacy-preserving* tag-based robust non-malleable in the restricted RO model under the GDH assumption and the CKEA assumption. In particular, as a warm-up, we show that the DIKE protocol also implies a 3-round adaptive tag-based concurrent non-malleable statistical (straight-line) zero-knowledge argument of knowledge for discrete logarithm (DL), which is presented in Section 5. We then prove that the DIKE protocol, *with exposable DH-exponents and pre-computed DH-components*, is SK-secure in the CK-framework with post-specified peers under the GDH assumption in the random oracle model. We suggest that the (restricted) RO and the CKEA assumption might be unavoidable to achieve *highly practical* DHKE protocols of the TBRNM security (particularly with the SK-security *simultaneously*). But, the proof details of TBRNM and SK-security are somewhat tedious and conceptually less interesting. For space limitation and to avoid potential sidetracking, the reader is referred to the full paper for complete proof details. Below, we mainly provide high-level overviews of the TBRNM analysis (particularly, the tricks of using CKEA assumption and restricted RO in the TBRNM analysis) and the SK-security analysis.

TBRNM analysis overview. For the TBRNM analysis, the polynomial-time simulator S generates DH-components and DH-exponents by itself by emulating honest player instances. But, different from honest player instances, S uses the DH-exponents (generated by S itself) merely for DDH-tests in its simulation. To this end, S maintains a DDH-test list, denoted $\mathcal{L}_{DDH}$, and stores all DDH-test records into $\mathcal{L}_{DDH}$. The key observation is: what can be done by the simulator S can also be done by another efficient oracle machine $S^{\mathcal{O}_\mathcal{C}}$ on the same common input and the random coins of S *except the coins used to generate the DH-components*, where $\mathcal{O}_\mathcal{C}$ is a DDH-oracle and $\mathcal{C} = \{X_1^l, \cdots, X_{s(n)}^l, Y_1^r, \cdots, Y_{s(n)}^r\}$ is the set of all the DH-components generated by S. Specifically, $S^{\mathcal{O}_\mathcal{C}}$ works just as S does, but with the following modifications: (1) $S^{\mathcal{O}_\mathcal{C}}$ just sets the DH-component for the i-th left-session (resp., the j-th right-session) to be the value X_i^l (resp., Y_j^r), $1 \leq i, j \leq s(n)$, given in the set of $\mathcal{C}$, rather than generating them by itself as S does. (2) Whenever $S^{\mathcal{O}_\mathcal{C}}$ needs to perform a DDH-test w.r.t. a DH-component in $\mathcal{C}$, it queries the DDH-test to its oracle $\mathcal{O}_\mathcal{C}$ and stores the record of the DDH-test into $\mathcal{L}_{DDH}$. Whenever $S^{\mathcal{O}_\mathcal{C}}/S$ needs to extract the DH-exponent and/or secret-key corresponding to the DH-component and/or public-key sent and alleged by the CMIM adversary $\mathscr{A}$, $S^{\mathcal{O}_\mathcal{C}}/S$ runs the CKEA-extractor $\mathcal{K}$ on the same common input, the random coins of $S^{\mathcal{O}_\mathcal{C}}$ *that just correspond to the coins of S except the coins used to generates the DH-components X_i^l's and Y_j^r's*, and $\mathcal{L}_{DDH}$ that corresponds to the vector of records of DDH-tests performed by $\mathcal{O}_\mathcal{C}$. By the CKEA assumption, $\mathcal{K}$ will successfully extract the corresponding DH-exponents and/or secret-keys with overwhelming probability.

For the use of restricted RO, whenever S needs to send one of the values $NMZK(\hat{B}, y)$, $NMZK(a, x)$ and $NMZK(b, y)$, it first checks whether the value has been defined by checking all RO queries made by $\mathscr{A}$ and performing corresponding DDH-tests. If the value to be sent has already been defined (by $\mathscr{A}$'s RO query), the value is set to be the already defined one, otherwise, S sends a random value. If S sends a random value, from this point on whenever $\mathscr{A}$ makes an RO query, S checks whether the previously sent random value is the answer to the RO query (again by performing DDH-tests). Note that in the later case (i.e., the value to be sent has not been defined), S does not try to use its knowledge of DH-exponents (generated by itself) to honestly generate such values, to ensure that those DH-exponents are used merely for DDH-tests in order to comply with the CKEA assumption. If $\mathscr{A}$ never makes an RO query with the previously sent random value as the RO answer, the RO on this point remains undefined. In particular, S never defines it on its own, which ensures S works in the restricted RO model. By the above tricks, the simulator S works in *strict* polynomial-time and its simulation is *straight-line* (without rewinding $\mathscr{A}$).

SK-security analysis overview. The core of the SK-security analysis is to prove that any PPT CMIM attacker can successfully finish an *unexposed* session in the name of some uncorrupted player only if that uncorrupted player (impersonated by the CMIM attacker) does indeed send the authenticated value, say, $NMZK(a, x)$ or $NMZK(b, y)$, in the corresponding matching session. In more details, we prove that: for the DIKE protocol $\langle \hat{A}, \hat{B} \rangle$ (depicted in Figure 1) *with exposable DH-exponents and pre-computed DH-components*, where the players $\hat{A}$ and $\hat{B}$ may be identical, the probability of the following events is negligible under the GDH assumption in the random oracle model:

Event-1. The CMIM adversary $\mathscr{A}$ successfully finishes the j-th right-session for some j, $1 \leq j \leq s(n)$, where $\mathscr{A}$ sends $\widetilde{NMZK}(a, \tilde{x}_j^r)$ in the third-round in the name of $\hat{A}$ (actually, any uncorrupted player) with respect to the DH-component Y_j^r sent by the uncorrupted player $\hat{B}$ in the second-round, while the uncorrupted player $\hat{A}$

did not send $\widetilde{NMZK}(a, \tilde{x}_j^r)$ in any left-session and $\mathscr{A}$ does not know the discrete-logarithm of Y_j^r (i.e., $\mathscr{A}$ did not make the state-reveal query against the j-th right-session at the uncorrupted player $\hat{B}$ in accordance with the CK-framework).

Event-2. The CMIM adversary $\mathscr{A}$ successfully finishes the i-th left-session for some i, $1 \leq i \leq s(n)$, where $\mathscr{A}$ sends $\widetilde{NMZK}(b, \tilde{y}_i^l)$ in the fourth-round in the name of $\hat{B}$ (actually, any uncorrupted player) with respect to the DH-component X_i^l sent by the uncorrupted player $\hat{A}$ in the first-round, while the uncorrupted player $\hat{B}$ did not send $\widetilde{NMZK}(b, \tilde{y}_i^l)$ in any right-session and $\mathscr{A}$ does not know the discrete-logarithm of X_i^l (i.e., $\mathscr{A}$ did not make the state-reveal query against the i-th left-session at the uncorrupted player $\hat{A}$ in accordance with the CK-framework).

Now, suppose the DIKE protocol is not SK-secure, which roughly means that $\mathscr{A}$ can distinguish the session-key $H_K(X, Y, g^{xy})$ of an *unexposed* test-session, say a left-session $(\hat{A}, sid)$ at the side of the uncorrupted player $\hat{A}$, from a random value. Let $X = g^x$ (resp, $Y = g^y$) be the DH-component sent by $\hat{A}$ (resp., $\hat{B}$), and $NMZK(b, y) = H(sid, \hat{B}, Y, X, X^y, X^b)$ be the authentication value sent by $\hat{B}$ (maybe impersonated by $\mathscr{A}$) in the fourth-round of this test-session. By the above discussions, we have that with overwhelming probability the uncorrupted player $\hat{B}$ does indeed send $NMZK(b, y)$ in one of right-sessions. This implies that in the RO model with overwhelming probability, the (left) test-session has matching (right) session $(\hat{B}, sid)$ in which $\hat{B}$ sends Y in the second-round (after receiving X in the first-round but not necessarily in the peer name of $\hat{A}$) and $NMZK(b, y)$ in the fourth-round.

As the session-key is computed as $H_K(X, Y, g^{xy})$ and H_K is a random oracle, there are only two strategies for the adversary $\mathscr{A}$ to distinguish $H_K(X, Y, g^{xy})$ from a random value:

– *Key-replication attack:* $\mathscr{A}$ succeeds in forcing the establishment of a session (other than the test-session or its matching session) that has the same session-key output as the test-session. In this case, $\mathscr{A}$ can learn the test-session key by simply querying that session to get the same key (without having to expose the test-session or its matching session).
– *Forging attack:* At some point in its run, $\mathscr{A}$ queries the RO H_K with (X, Y, g^{xy}).

The possibility of the key-replication attack is trivially ruled out in the RO model, by observing that X is only sent by $\hat{A}$ in the test-session and Y is only sent by $\hat{B}$ in the matching session.

The success of the forging attack says $\mathscr{A}$ can successfully output $(X, Y, CDH(X, Y))$. Recall that, with overwhelming probability, X and Y are only sent *by uncorrupted players* in the test-session and its matching session. As both the test-session and its matching session are assumed to be unexposed in accordance with the CK-framework (and thus $\mathscr{A}$ does not know the DH-exponent x or y), then we can exploit the assumed ability of $\mathscr{A}$ in performing the successful forging attack to break the CDH assumption (with the assistance of the DDH-oracle $\mathcal{O}_X$ or $\mathcal{O}_Y$), which in turn violates the GDH assumption.

4.5 Discussions on the Resistance against Some Concrete Attacks

The both TBRNM security and SK-security of our DIKE protocol imply the resistance to most concrete yet essential security attacks against DHKE protocols (some of which are beyond the SK-security), particularly, unknown key share (UKS), key compromise impersonation (KCI), cutting-last-message attack, perfect forward security (PFS), reflection attacks, etc. In this section, we make informal discussions on the resistance to some of these concrete attacks, with more details deferred to the full paper.

Resistance against unknown key share attack. Informally speaking, by a successful UKS attack an adversary can successfully make two uncorrupted parties compute the same session-key in two sessions but have different views of who the peer to the exchange was, even if the adversary actually does not know the corresponding session-key.

For a successful UKS attack against our DIKE protocol, between two sessions of different pairs of players, we have the following observations: As the session-key is derived from $H(X, Y, g^{xy})$ and the two sessions are of the same session-key, with overwhelming probability in the RO model these two sessions must be of the same DH-components, say (X, Y), and furthermore, in the same (initiator and responder) order. Note that, with overwhelming probability, there are at most two sessions (involving uncorrupted players) of the same DH-components exchanged in the same order, as uncorrupted players generate DH-components randomly and independently. In other words, besides the two sessions suffering from the UKS attack, there exist no other sessions of the same (ordered) DH-components (X, Y). This implies that each one of the two sessions (suffering from the UKS attack) is of a *distinct* tag, i.e., different from the tags of all other sessions. By the tag-based robust non-malleability, the adversary must know both of the corresponding DH-exponents x and y (and also the secret-keys corresponding to the public-keys alleged by the adversary in the two sessions). This particularly implies that the adversary does know the session-key $H(X, Y, g^{xy})$, which violates the assumed success of the UKS attack.

Resistance against cutting-last-message attack. Suppose the player $\hat{B}$ sends the last message in the run of a DHKE protocol $\langle \hat{A}, \hat{B} \rangle$, the cutting-last-message attack, suffered by IKEv2, works as follows [29]: A man-in-the-middle $\mathscr{A}$ interacts with the uncorrupted $\hat{B}$ in the name of $\hat{A}$ in a session (referred to as the test-session), while concurrently interacting with the uncorrupted $\hat{A}$ in the name of $\hat{M} \neq \hat{B}$ in another session (referred to as the matching session). $\hat{M}$ just relays messages between $\hat{A}$ and $\hat{B}$ in these two sessions, but aborts the matching session after receiving the last message from $\hat{B}$ in the test-session. Such a simple attack results in authentication failure as follow: $\hat{B}$ is perfectly fooled to believe that it has shared a session key with $\hat{A}$ in the test-session, while $\hat{A}$ thinks it only ever took part in an aborted session with $\hat{M}$ in the matching session. (As suggested in [7], this cutting-last-message attack can be prevented by adding an additional fifth-round of "acknowledgement" from $\hat{A}$ to $\hat{B}$, but increasing the system complexity.)

Such cutting-last-message attack is simply ruled out for our DIKE protocol by the tag-based robust non-malleability of DIKE. Specifically, for the above cutting-last-message attack, with overwhelming probability the tag of the completed test-session (i.e., the one in which $\hat{B}$ believes it has shared a session-key with $\hat{A}$) must be *distinct*;

In particular, it is different from the tag of the aborted matching session in which $\mathscr{A}$ interacts with $\hat{A}$ in the name of $\hat{M} \neq \hat{B}$. By the tag-based robust non-malleability, it implies that $\mathscr{A}$ has to know $\hat{A}$'s secret-key a and the DH-exponent generated by $\hat{A}$ in the aborted matching session, in order to successfully complete the test-session with $\hat{B}$ in the name of $\hat{A}$. In particular, after receiving the second-round message $(\hat{B}, Y = g^y, NMZ(\hat{B}, y))$ from $\hat{B}$ in the test-session, the CMIM adversary $\mathscr{A}$ cannot compute and send to $\hat{A}$ the message of $(\hat{M}, Y = g^y, NMZK(\hat{M}, y))$ in the name of $\hat{M} \neq \hat{B}$ in the matching session.

Implication of perfect forward secrecy. Informally, a key-exchange protocol is of the PFS property, if the leakage of the static secret-key of an uncorrupted player does not compromise the security of the session-keys established by the player for unexposed yet expired sessions, which have been erased from memory before the leakage occurred [25]. In other words, once an unexposed session is expired and the session-key is erased from its holder's memory, then the session-key cannot be learned by the attacker even if the player is subsequently corrupted. The PFS property of our DIKE protocol is from the observation that: the computation of the session-key $H_K(X, Y, g^{xy})$ does not involve players' secret-keys. Note also that secret-key leakage has already been captured by the TBRNM formulation.

Resistance against reflection attack. In a *reflection* attack, an attacker simply copies the authentic messages from $\hat{A}$ and sends them back to $\hat{A}$ as the messages coming from the other copy of $\hat{A}$. With respect to the protocol structure of our deniable IKE, to mount a successful reflection attack against the DIKE protocol an adversary has to set $Y = X$ and $\hat{B} = \hat{A}$ so that $(X, Y) = (Y, X)$ and $NMZK(b, y) = NMZK(a, x)$. But, this play is frustrated with our DIKE, by briefly noting that the adversary cannot provide the proof-of-knowledge of $y = x$, i.e., $NMZK(\hat{B}, y) = NMZK(\hat{A}, x)$, in the second-round.

5 Protocol Variants and Implications

Deniable IKE: the aggressive model. In accordance with the aggressive model of IKE [21,22], we present the 3-round variant of our DIKE protocol in Figure 2.

Most security properties of the deniable IKE of the main model are essentially inherited by this 3-round protocol variant in the aggressive model, except for the full deniability for the responder player $\hat{B}$. Specifically, the player $\hat{B}$ only enjoys *completed-session deniability*, in the sense that if a malicious player $\hat{A}$, denoted as $\hat{A}^*$, completes the session then $\hat{B}$'s deniability will be guaranteed. But if the (possibly malicious) $\hat{A}^*$ just aborts the session after receiving the second-round message, the deniability for $\hat{B}$ is not ensured. Note that the initiator player $\hat{A}$ still has full deniability. We remark that such kind of completed-session deniability for the responder is still very useful and reasonable. For instance, consider the scenario where $\hat{A}$ is a client and $\hat{B}$ is a (bank or shop) server: in such a scenario it is the client $\hat{A}$ who cares more about its privacy and full deniability does guarantee for it, while the server cares less about deniability and the completed-session deniability may still be deemed to be good enough for it.

3-round adaptive tag-based concurrent non-malleable (statistical straight-line) zero-knowledge argument of knowledge (CNMZKAOK) for DL. Let $A = g^a \in G$ be the

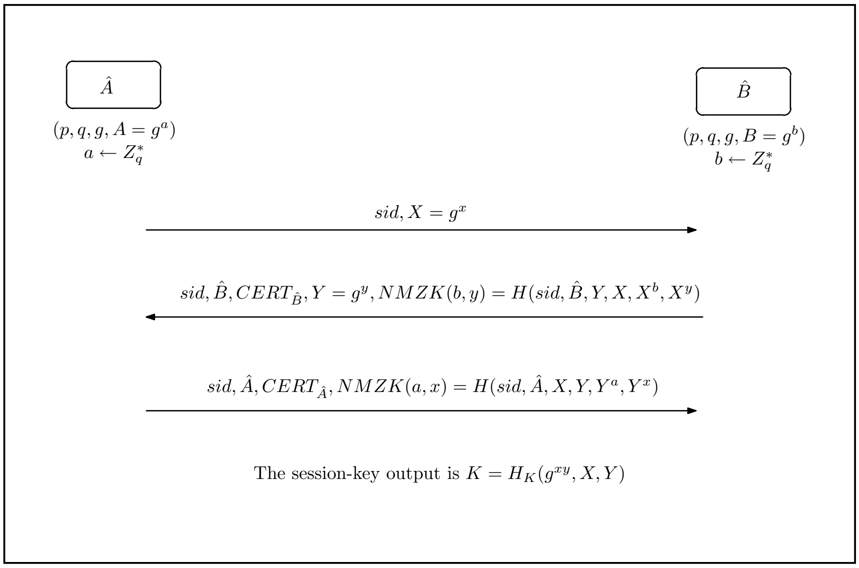

$\hat{A}$
$(p, q, g, A = g^a)$
$a \leftarrow Z_q^*$
$\hat{B}$
$(p, q, g, B = g^b)$
$b \leftarrow Z_q^*$
$sid, X = g^x$
$sid, \hat{B}, CERT_{\hat{B}}, Y = g^y, NMZK(b, y) = H(sid, \hat{B}, Y, X, X^b, X^y)$
$sid, \hat{A}, CERT_{\hat{A}}, NMZK(a, x) = H(sid, \hat{A}, X, Y, Y^a, Y^x)$
The session-key output is $K = H_K(g^{xy}, X, Y)$

Fig. 2. Deniable Internet Key-Exchange (the aggressive model)

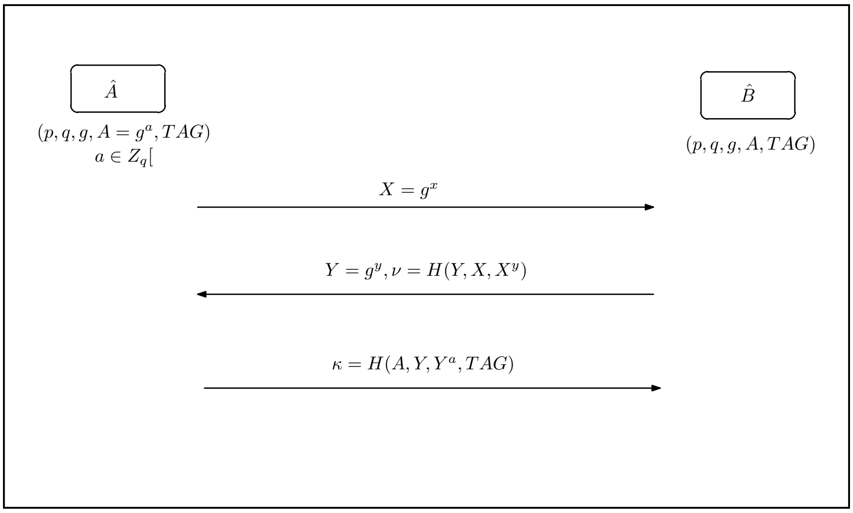

$\hat{A}$
$(p, q, g, A = g^a, TAG)$
$a \in Z_q[$
$\hat{B}$
(p, q, g, A, TAG)
$X = g^x$
$Y = g^y, \nu = H(Y, X, X^y)$
$\kappa = H(A, Y, Y^a, TAG)$

Fig. 3. Adaptive tag-based straight-line CNMZKAOK for DL in the restricted RO model

common input (where the group G is specified by the parameters (p, q, g)), $a \in Z_q$ be the private input of the prover $\hat{A}$, Tag be the session-tag, and H be a hash function that is assumed to be a (restricted) random oracle. The protocol of adaptive tag-based CNMZKAOK for DL is depicted in Figure 3 (page 345), where the DH-component X

(resp., Y) is taken randomly and independently from $G/1_G$ by $\hat{A}$ (resp., $\hat{B}$) and each player checks the $G/1_G$ membership of its peer's DH-component.

Note that in the protocol specification, for presentation simplicity, the session-tag Tag is predetermined and known to both the prover $\hat{A}$ and the verifier $\hat{B}$ prior to the protocol run. In an actual adversarial setting, the session-tag may be set by the CMIM adversary adaptively during the protocol run based on its view in all the concurrent (left and right) sessions. The security analysis given in the full paper, which is based upon the CKEA assumption in the restricted RO model, is w.r.t. this general adversarial setting of adaptive tag selection.

The 3-round adaptive tag-based CNMZKAOK protocol for DL, depicted in Figure 3, further implies a 3-round concurrent and *forward* deniable authentication protocol [15], based on the CKEA assumption and the DL assumption in the restricted RO model, by viewing messages to be authenticated as the session-tags.

Acknowledgment. We are indebted to Frances F. Yao and Bin Zhu for many contributions to the earlier versions of this work, though they finally declined the coauthorship.

References

1. Bellare, M., Palacio, A.: The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
2. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
3. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 273–289. Springer, Heidelberg (1994)
4. 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)
5. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively Secure Multi-Party Computation. In: ACM Symposium on Theory of Computing, pp. 639–648 (1996)
6. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 453. Springer, Heidelberg (2001)
7. Canetti, R., Krawczyk, H.: Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002)
8. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-Box Concurrent Zero-Knowledge Requires $\tilde{\Omega}(log\ n)$ Rounds. In: ACM STOC 2001, pp. 570–579 (2001)
9. Damgård, I.: Towards Practical Public-Key Systems Secure Against Chosen Ciphertext Attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
10. Dent, A.: Cramer-Shoup Encryption Scheme is Plantext Aware in the Standard Model. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 289–307. Springer, Heidelberg (2006)
11. Di Raimondo, M., Gennaro, R.: New Approaches for Deniable Authentication. In: Proc. of 12nd ACM Conference on Computer and Communications Security (ACM CCS'05), pp. 112–121. ACM Press, New York (2005)

12. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Deniable Authentication and Key Exchange. In: ACM CCS'06, pp. 466–475 (2006); Full version appears in Cryptology ePrint Archive Report No. 2006/280
13. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
14. Dodis, Y., Katz, J., Smith, A., Walfish, S.: Composability and On-line Deniability of Authentication. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 146–162. Springer, Heidelberg (2009)
15. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography. SIAM Journal on Computing 30(2), 391–437 (2000); Preliminary version in ACM Symposium on Theory of Computing, pp. 542–552 (1991)
16. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. In: ACM Symposium on Theory of Computing, pp. 409–418 (1998)
17. FIPS Pub 186-2, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-2, US Department of Commerce/National Institute of Standard and Technology, Githersburg, Maryland, USA (January 27, 2000)
18. Goldreich, O., Micali, S., Wigderson, A.: How to Play Any Mental Game. In: ACM Symposium on Theory of Computing, pp. 218–229 (1987)
19. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. In: ACM Symposium on Theory of Computing, pp. 291–304 (1985)
20. Hada, S., Tanaka, T.: On the Existence of 3-Round Zero-Knowledge Protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998)
21. Harkins, D., Carreal, D. (eds.): The Internet Key-Exchange (IKE), RFC 2409 (November 1998)
22. Kaufman, C.: Internet Key Exchange (IKEv2) Protocol. The Internet Engineering Task Force: INTERNET-DRAFT (October 2002)
23. Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol. Request for Comments 2401 (1998)
24. Krawczyk, H.: SIGMA: the "SIGn-and-MAc" Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003)
25. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
26. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An Efficient Protocol for Authenticated Key Agreement. Designs, Codes and Cryptography 28, 119–134 (2003)
27. Lindell, Y.: General Composition and Universal Composability in Secure Multi-Party Computation. In: IEEE Symposium on Foundations of Computer Science, pp. 394–403 (2003)
28. Lindell, Y.: Lower Bounds and Impossibility Results for Concurrenet Self Composition. Journal of Cryptology 21(2), 200–249 (2008)
29. Mao, W.: Modern Cryptography: Theory and Practice. Prentice Hall PTR, Englewood Cliffs (2004)
30. Maurer, U., Wolf, S.: Diffie-hellman oracles. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 268–282. Springer, Heidelberg (1996)
31. Menezes, A.: Another Look at HMQV. Cryptology ePrint Archive, Report No. 2005/205
32. Naor, M., Reingold, O.: Number-Theoretic Constructions of Efficient Pseudo-Random Functions. Journal of the ACM 1(2), 231–262 (2004)
33. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)

34. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
35. Pass, R.: On Deniabililty in the Common Reference String and Random Oracle Models. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)
36. Pass, R., Rosen, A.: New and Improved Constructions of Non-Malleable Cryptographic Protocols. In: ACM Symposium on Theory of Computing, pp. 533–542 (2005)
37. Pass, R., Rosen, A.: Concurrent Non-Malleable Commitments. In: IEEE Symposium on Foundations of Computer Science, pp. 563–572 (2005)
38. Stinson, D.R., Wu, J.: An Efficient and Secure Two-Flow Zero-Knowledge Identification Protocol. Cryptology ePring Archive, Report 2006/337
39. Stinson, D.R., Wu, J.: A Zero-Knowledge Identification and Key Agreement Protocol. Cryptology ePring Archive, Report 2007/116
40. Yao, A.C., Yung, M., Zhao, Y.: Adaptive Concurrent Non-Malleability with Bare Public-Keys. Cryptology ePrint Archive, Report 2010/107
41. Yung, M., Zhao, Y.: Interactive Zero-Knowledge with Restricted Random Oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 21–40. Springer, Heidelberg (2006)

A New Human Identification Protocol and Coppersmith's Baby-Step Giant-Step Algorithm

Hassan Jameel Asghar[1], Josef Pieprzyk[1], and Huaxiong Wang[1,2]

[1] Center for Advanced Computing – Algorithms and Cryptography, Department of Computing, Faculty of Science, Macquarie University, Sydney, NSW 2109, Australia
{hasghar,josef,hwang}@science.mq.edu.au

[2] Division of Mathematical Sciences, School of Physical & Mathematical Sciences, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore
hxwang@ntu.edu.sg

Abstract. We propose a new protocol providing cryptographically secure authentication to unaided humans against passive adversaries. We also propose a new generic passive attack on human identification protocols. The attack is an application of Coppersmith's baby-step giant-step algorithm on human identification protcols. Under this attack, the achievable security of some of the best candidates for human identification protocols in the literature is further reduced. We show that our protocol preserves similar usability while achieves better security than these protocols. A comprehensive security analysis is provided which suggests parameters guaranteeing desired levels of security.

Keywords: Human Computer Cryptography; Human Identification Protocols; Entity Authentication.

1 Introduction

Secure human identification protocols are a form of user authentication protocols through which the human user proves his/her identity to a remote computer server using an insecure public terminal and through an insecure channel. Such protocols are based on shared-key cryptography and are potentially more secure than traditional authentication methods such as passwords, biometrics, tokens or combinations of them, since the adversary has more powers in the threat model. The adversary can view the alphanumerics entered by the user, see the computations done at the terminal and above all observe the information exchanged between the user and the remote server during a protocol session. Much stronger models allow the adversary to actively interfere with the communication channel. This scenario was conceived by Matsumoto and Imai in [4].

Although, human identification protocols secure against active attacks is coveted, it is extremely hard to construct one that ensures both acceptable security and good human executability. To date there have been a handful of proposals known to resist some active attacks [1,6,8,13]. Among them only the *sum of k* *mins* protocol proposed in [1] has been constructed to be secure against generic active adversaries; yet, the protocol falls short of usability. The other protocols only consider security against a known set of active attacks. Due to the difficulty of constructing usable protocols secure against active attacks, recently the focus of the research community has been on security against passive (eavesdropping) adversaries [2,3,10,12]. Despite this being a weaker threat model, there is still no widely accepted human identification protocol secure against passive adversaries and this remains an open problem. Recently proposed protocols can be used for only a small number of authentication sessions if a certain level of usability is desired. Loosely speaking, most of these protocols consist of a shared secret that is a set of k objects out of n. The protocol involves a series of challenges from the server and corresponding responses by the user, which are constructed as a function of the secret and the challenges. Since the challenges and their responses are communicated in the open, the adversary can always do a search after learning a few challenge-response pairs to find the secret. We shall call protocols belonging to this general category, *k-out-of-n protocols*. Hopper and Blum showed a bound on the achievable security level for this class of protocols assuming a certain generic time-memory tradeoff attack to be optimal [1, §6]. In other words, all k-out-of-n protocols are susceptible to this time-memory tradeoff attack, even if they do not have any other weaknesses. This bound severely reduces accomplishable security level for small values of k. A smaller value of k is necessary to ensure that human memory and computational requirements are low.

Hopper and Blum also proposed two protocols for human identification. One of them is the now famous HB protocol based on the problem of learning parity in the presence of noise. However, the variant of this protocol proposed as a human identification protocol uses a small hamming weight k. We call this protocol the k-weight HB protocol to distinguish it from the HB family of protocols used for pervasive devices. The other protocol is the sum of k mins protocol mentioned before. We use the variant that is constructed to be secure against passive adversaries only. The two protocols achieve good security against passive adversaries but fall short of the theoretically achievable bound imposed by the time-memory tradeoff attack. The reason for this is an attempt to maintain a reasonable level of usability.

In this paper, we propose a protocol that achieves better security than the aforementioned protocols while preserving a similar usability level. We also propose a new time-memory tradeoff attack on human identification protocols that has lower time-complexity than the one sketched in [1, §6]. The main component of the attack is Coppersmith's baby-step giant-step algorithm, which has its application in solving the restricted hamming weight discrete logarithm problem [14]. Once again, this attack can be applied to all k-out-of-n protocols. It performs better than the attack mentioned in [1] on the two proposed protocols in that paper, further reducing their security level. Our protocol shows better security under both time-memory tradeoff attacks. We also rigorously analyze other possible attacks to demonstrate their efficacy, or lack thereof, in breaking our protocol. Our focus is restricted to passive adversaries.

2 Related Work

The first human identification protocol that was constructed to be secure in the aforesaid threat model was proposed by Matsumoto and Imai in [4]. This scheme was shown to be insecure by Wang, Hwang and Tsai in [6] and the authors proposed some improvements but with a severe loss in usability. Matsumoto proposed another scheme in [7] based on techniques from linear algebra, but it can only be used for a small number of authentication sessions. Yang and Teng proposed a couple of protocols in [8] which with the suggested memory requirement of three 20 to 40-bit secrets, is surely infeasible for most humans. Hopper and Blum proposed two human identification protocols in [1]. One of them is the k-weight HB protocol and the best known attack on this protocol is the meet-in-the-middle attack [1, §3.1, pp. 58]. The protocol requires the human to toss a coin with a fixed probability between 0 and 0.5, which is arguably not achievable without additional aid. The other proposal by the authors is the sum of k mins protocol. The version of this protocol secure against passive adversaries is, in our opinion, the most secure and usable protocol published to date. Our protocol performs slightly better, in that it does not use ordered pairs and hence the user does not necessarily have to remember the secret objects in a specific order. Furthermore, the security of our protocol is much higher for similar parameter values.

Li and Shum [13] describe two protocols which resist a large number of passive attacks and can be used for a large number of authentication sessions. However, the security analysis of their protocols does not take time-memory tradeoff attacks into account. Weinshall [10] proposed a slightly different protocol in which the user has to remember images instead of alphanumeric secrets. The protocol was cryptanalyzed and broken by Golle and Wagner in [11]. Use of images as memory aids has been employed previously, such as in [5]. However, straightforward protocols in which the user is required to click on a subset of secret pictures in a set of given pictures is susceptible to shoulder-surfing or passive observer attacks. For a comprehensive survey of these protocols and others see [9]. More recently, the authors of [2] and [3] proposed a slightly different concept in which the internal properties of images are used as secrets. However, the security of the protocols cannot be concretely demonstrated as it relies on some unproven assumptions. Another recent attempt is by Bai et. al. in [12], but their protocol was shown insecure by Li et al. [17].

3 Preliminaries: Definitions and Threat Model

Throughout this text, the prover is denoted by $\mathcal{H}$ and the verifier by $\mathcal{C}$. This is to acknowledge that in the real world, the prover will be a human user and the verifier will be a remote computer (server). The goal of an identification protocol is to authenticate $\mathcal{H}$ to $\mathcal{C}$. We restate the definitions of identification protocols and human executable protocols from [1] for reference. A protocol is defined as a sequence of interactions between a pair of public and probabilistic interactive turing machines (ITMs) $(\mathcal{H}, \mathcal{C})$. The result of the interaction between these two (ITMs) with respective inputs x and y is denoted by $\langle \mathcal{H}(x), \mathcal{C}(y) \rangle$. The transcripts of bits exchanged between $\mathcal{H}$ and $\mathcal{C}$ during this interaction is denoted by $T(\mathcal{H}(x), \mathcal{C}(y))$.

Definition 1. *An* identification protocol *is a pair of public, probabilistic interactive turing machines (ITMs)* $(\mathcal{H}, \mathcal{C})$ *with shared auxiliary input* z*, such that the following conditions hold:*
– *For all auxiliary inputs* z, $\Pr[\langle \mathcal{H}(z), \mathcal{C}(z) \rangle = \mathsf{accept}] > p_0$
– *For each pair* $x \neq y$, $\Pr[\langle \mathcal{H}(x), \mathcal{C}(y) \rangle = \mathsf{accept}] < 1 - p_0$ *where* $0.5 < p_0 \leq 1$.

When $\langle \mathcal{H}, \mathcal{C} \rangle = \mathsf{accept}$, we say that $\mathcal{H}$ authenticates to $\mathcal{C}$. For human computational ability, we shall use the following definition from [1]:

Definition 2. *An identification protocol* $(\mathcal{H}, \mathcal{C})$ *is* (α, β, τ)-human executable *if at least a* $(1 - \alpha)$ *portion of the human population can perform the calculations* $\mathcal{H}$ *unaided and without errors in at most* τ *seconds with probability greater than* $(1 - \beta)$.

The goal is to minimize α, β and τ. Concrete values to these parameters can be assigned by either an intuitive approximation or where it is hard to do so, actual user experiments can be carried out [1].

3.1 Security Definitions

The adversary, denoted by $\mathcal{A}$, has passive access to the channel between $\mathcal{H}$ and $\mathcal{C}$. In this light, the adversary views both $\mathcal{H}$ and $\mathcal{C}$ as oracles. The following definition treats $\mathcal{H}$ and $\mathcal{C}$ as Interactive Turing Machines. In the identification protocols discussed in this paper the computations done by $\mathcal{H}$ have to be carried out by a human user. Therefore, all such computations should be done by the human mentally, or else any security proved against $\mathcal{A}$ will be superfluous.

The following security definition is taken from [1] and involves the passive adversary $\mathcal{A}$ defined above.

Definition 3. *An identification protocol* $(\mathcal{H}, \mathcal{C})$ *is* (p, m) *secure against passive adversaries if for all computationally bounded adversaries* $\mathcal{A}$ *and for all auxiliary inputs* z,

$$\Pr[\langle \mathcal{A}(T^m(\mathcal{H}(z), \mathcal{C}(z))), \mathcal{C}(z) \rangle = \mathsf{accept}] \leq p$$

Here $T^m(.,.)$ *represents the transcript of* m *independent communication sessions between* $\mathcal{H}$ *and* $\mathcal{C}$.

If $\langle \mathcal{A}(T^m(\mathcal{H}, \mathcal{C})), \mathcal{C} \rangle = \mathsf{accept}$ for some m, we say that $\mathcal{A}$ impersonates $\mathcal{H}$. We define a challenge-response identification protocol as the following sequence of messages communicated between $\mathcal{H}$ and $\mathcal{C}$:

$$\mathsf{request}, (c_1, r_1), (c_2, r_2), \ldots, (c_m, r_m), \mathsf{accept/reject}$$

The message request is sent by $\mathcal{H}$ to $\mathcal{C}$. It symbolizes a request to start a protocol and contains information about $\mathcal{H}$ such as its identity. Each pair (c_i, r_i) consists of a challenge c_i drawn from some *challenge space* by $\mathcal{C}$ and a response r_i composed from a *response space* by $\mathcal{H}$. At the end of m challenge-response pairs, $\mathcal{C}$ sends the message accept or reject. We call this sequence of messages as one *session* (or an authentication session) of the challenge-response identification protocol.

4 Proposed Protocol

Notation. We refer to a vector of n elements or an n-tuple as an ordered list of n elements. If $\mathbf{c}$ is a vector of n-elements, then $\mathbf{c}[i]$ denotes the ith element of $\mathbf{c}$, for $0 \le i \le n-1$; the index i is called the ith location in $\mathbf{c}$. Let n be such that $n = ab$ for positive integers a and b. We call a the *jump constant* for reasons that will be clear later. Let k and d be positive integers. Let $\mathbf{c}$ be a vector of n integers drawn uniformly at random from the set $\{0, 1, \ldots, d-1\}$.

We first describe the protocol formally and then show an implementation that is human friendly. Sections 4.1 and 4.2 give a less technical description of the protocol with example values for the parameters.

Protocol 1.

Setup: Let n, a, b, m, k and d be public parameters, with $n = ab$. $\mathcal{C}$ and $\mathcal{H}$ choose $k+1$ locations in $\mathbf{c}$. These locations are essentially a set of integers: $s_0, s_1, \ldots, s_k$, where $0 \le s_i \le n-1$. s_0 is called the starting location. All these locations constitute the secret.

```
1: repeat
2:     $\mathcal{C}$ updates $m \leftarrow m-1$, generates the vector $\mathbf{c}$ and sends it to $\mathcal{H}$.
3:     $\mathcal{H}$ assigns $t \leftarrow s_0$.
4:     Vertical movement: $\mathcal{H}$ updates $t \leftarrow (t + a \cdot \mathbf{c}[s_0]) \bmod n$.
5:     for $1 \le i \le k$ do
6:         Horizontal movement: $\mathcal{H}$ updates $t \leftarrow (t + \mathbf{c}[s_i]) \bmod n$.
7:     $\mathcal{H}$ sends $\mathbf{c}[t]$ to $\mathcal{C}$.
8:     if the answer is incorrect then
9:         $\mathcal{C}$ outputs reject.
10: until $m = 0$
11: $\mathcal{C}$ outputs accept.
```

It is clear that the above protocol is an identification protocol in which $\mathcal{C}$ accepts the legitimate prover with probability 1 and the success probability of an impersonator is less than or equal to d^{-m}.

4.1 User Friendly Implementations

Both graphical and textual implementations are possible for our protocol. In a graphical implementation, we can use n graphical objects such as software icons. In this case, the user's secret is a set of k icons out of n. Here we illustrate an example text-based implementation.

We represent the secret space, i.e. the locations in **c**, by an alphabet. For instance, the English alphabet is a candidate. In this subsection, we abuse the notation a little to denote the human user by $\mathcal{H}$. The vector **c** is presented to $\mathcal{H}$ in the form of a grid with $a \times b$ cells, where $b = n/a$ (a is chosen such that it divides n). Each location in **c** is mapped to a unique character in the secret space alphabet. Each cell in the grid contains a unique character from the secret space, below which is the corresponding random digit from **c**. The secret is then a string from this alphabet, instead of a set of integers of vector locations. Thus, the starting location is also a character from this alphabet.

Given a challenge "grid", $\mathcal{H}$ locates the cell containing the starting character. This corresponds to the location s_0 in the formal description. $\mathcal{H}$ then looks at the digit corresponding to this cell. Let the digit be d_0. $\mathcal{H}$ moves d_0 steps vertically downwards, in a circular way, thus reaching a new character location in the same grid. Call this location l_0. $\mathcal{H}$ then looks at the digit in the cell containing s_1. Let d_1 be the digit. It now moves horizontally to the right of the location l_0 and moving to the start of the next row if the end of the row is reached. This results in the new location l_1. $\mathcal{H}$ continues to move horizontally according to the digits corresponding to the rest of its secret locations. If the bottom right corner of the grid is reached, $\mathcal{H}$ moves to the top left corner, thus moving in a cycle. At the end of this procedure, $\mathcal{H}$ simply outputs the digit corresponding to the final character location thus reached. Figure 1 shows an example. Here $a = 12$ and $b = 6$, which implies that $n = 72$. In this example and also through most of the text, we will choose $d = 10$, as this is a common base for humans. The alphabet is composed of the characters a, . . . , z, A, . . . , Z, 0, . . . , 9 and special characters: !, @, #, \$, %, $\wedge$, &, $*$, (,). Notice that the order of the characters remains the same for all challenges and only the digits corresponding to these characters change for different challenges. For the graphical implementation, we simply replace the alphabet with a corresponding set of graphical objects. While the text-based implementation is shown here as an illustration of our protocol, we recommend graphical implementation as it is more user-friendly and has less security issues, such as resistence to dictionary attacks.

a	b	c	d	e	f	g	h	i	j	k	l
3	2	6	9	2	1	7	5	4	4	6	8
m	n	o	p	q	r	s	t	u	v	w	x
1	6	7	4	9	7	5	3	2	7	6	1
y	z	A	B	C	D	E	F	G	H	I	J
3	2	5	1	5	2	9	6	6	8	6	0
K	L	M	N	O	P	Q	R	S	T	U	V
3	1	7	4	9	7	5	3	4	7	6	1
W	X	Y	Z	0	1	2	3	4	5	6	7
6	3	8	2	6	8	3	2	9	5	8	0
8	9	!	@	#	$	%	∧	&	∗	(	)
4	7	2	0	9	1	5	3	6	3	4	9

Fig. 1. An example challenge grid

4.2 Different Ways of Computation

The above mentioned procedure is one way the human user can perform the protocol steps. There are a number of other ways in which the protocol can be executed. The user can choose any method he or she prefers.

For instance, a different and probably more efficient way is sketched here.

1. Ignore the starting location and add all the digits corresponding to the remaining k secret locations.
2. From the starting location, move d_0 steps vertically downwards (continuing from the top if the bottom of the grid is reached), where d_0 is the digit corresponding to the starting location.
3. Divide the sum obtained in Step 1 by the jump constant a to get a quotient and a remainder. The quotient is the number of vertical steps and the remainder is the number of horizontal steps to be taken. The user can then follows these steps from the location reached in Step 2 and finally output the digit corresponding to the location thus reached.

If the jump constant a is a multiple of 10 then the above division can be performed by most humans mentally. Thus to make this method easy for most users, n and a can be chosen to be 200 and 20 respectively. These parameters are easy for humans to use. k can be chosen somewhere between 10 and 15.

5 Security Analysis

Recall that for security, we consider the computationally bounded passive adversary of Definition 3. The adversary can view every challenge-response pair. The adversary knows the description of Protocol 1 and the public parameters specified in that protocol. The only thing hidden from the adversary is the set of secret locations shared by $\mathcal{C}$ and $\mathcal{H}$. In this section, we assume that the calculations $\mathcal{H}$ can be performed by a human mentally without any additional aid and thus $\mathcal{A}$ gains no advantage in observing the human user's behavior. We also assume that the secret locations are chosen uniformly at random from the set of all possible secret locations.

Given m challenge-response pairs $(\mathbf{c}_1, r_1), \dots, (\mathbf{c}_m, r_m)$, the goal of $\mathcal{A}$ is to impersonate $\mathcal{H}$ either by partially or completely learning the secret locations or by impersonating $\mathcal{H}$ by guessing the answers without knowledge of the secret. We assume that all these challenge-response pairs correspond to successful authentication sessions between $\mathcal{H}$ and $\mathcal{C}$. We first look at some obvious attacks following which we shall show more sophisticated attacks.

5.1 Some Obvious Attacks

Random Guess 1. The most obvious impersonation attack is to randomly guess the answers when prompted for a challenge. Since the output is in the range $\{0, 1, \dots, d-1\}$, a random guess from this set will be successful with probability $1/d$ for each challenge. For m challenges, this probability is d^{-m}.

Random Guess 2. Another method for impersonation is to guess the secret and then answer a challenge by following the steps of Protocol 1 correctly. We see that there are a total of $n\binom{n+k-1}{k}$ possible ways of choosing a secret. Thus the probability of success in guessing the correct secret is $(n\binom{n+k-1}{k})^{-1}$.

The hitherto mentioned attacks are online attacks and effective measures exist to prevent them. For instance, if there are more than a threshold number (say 3) of unsuccessful consecutive login attempts, the user account can be blocked.

Brute Force. The bruteforce attack has complexity $O(n\binom{n+k-1}{k})$. It works by trying all possible secret locations satisfying m challenge-response pairs. In the next section, we shall see that after observing m challenge-response pairs, the expected number of candidates for the secret is: $n\binom{n+k-1}{k}/d^m$. To reduce this number to a unique secret, $\mathcal{A}$ needs on average: $n\binom{n+k-1}{k}/d^m \approx 1 \Rightarrow m \approx \log_d(n\binom{n+k-1}{k})$ challenge-response pairs. Thus $O(\log_d(n\binom{n+k-1}{k}))$ challenge-response pairs are enough to find the secret uniquely. With a lower number of challenge-response pairs, there are multiple candidates and even a computationally unbounded adversary cannot distinguish between them. For concrete values, we see that if $n = 200$ and $k = 15$, the brute force attack has complexity roughly 2^{83}. An attack with this complexity is generally considered intractable.

5.2 Algebraic Interpretation

Given m challenge-response pairs $(\mathbf{c}_1, r_1), \ldots, (\mathbf{c}_m, r_m)$, we now consider the problem of finding the secret locations $s_0, \ldots, s_k$. We attempt to describe this problem algebraically. The following notations will be used henceforward: If $\mathfrak{A}$ is a set, then $|\mathfrak{A}|$ denotes the number of elements in $\mathfrak{A}$. If x and y are two strings, then $x||y$ represents their concatenation. If x and y are two integers, $[x, y]$ represents the interval of integers between x and y inclusive. A string of n-elements can be transformed into a vector of n-elements in a natural way, and the two terms will be used interchangeably in the following.

For any challenge-response pair $(\mathbf{c}, r)$, a location $\hat{r}$ satisfying $\mathbf{c}[\hat{r}] = r$ is called a *satisfying location* for $(\mathbf{c}, r)$. For $1 \leq i \leq m$, let $\mathfrak{R}_i$ be the set of all satisfying locations for $(\mathbf{c}_i, r_i)$. Let $\mathfrak{R}^m = \mathfrak{R}_1 \times \cdots \times \mathfrak{R}_m$ be the m-ary cartesian product over these m sets. We represent the elements of $\mathfrak{R}^m$ as m-element vectors, $\hat{\mathbf{r}} = [\hat{r}_1 \cdots \hat{r}_m]^T$, in an obvious way. For any vector $\mathbf{x}$, let $|\mathbf{x}|$ denote the number of elements in $\mathbf{x}$. Define the weight of $\mathbf{x}$ as:

$$\mathsf{wt}(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|} x_i$$

where $x_i = \mathbf{x}[i]$, the ith element of $\mathbf{x}$. Define the matrix C as:

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,n} \end{bmatrix}$$

where $c_{i,j} = \mathbf{c}_i[j]$. Then, for each $\hat{\mathbf{r}} \in \mathfrak{R}^m$ we have:

$$\left[aC\; C \right] \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \hat{\mathbf{r}} \bmod n \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are n-element vectors with $\mathsf{wt}(\mathbf{x}) = 1$ and $\mathsf{wt}(\mathbf{y}) = k$. Clearly, $\mathbf{x}$ corresponds to the starting secret location and $\mathbf{y}$ corresponds to the k remaining secret locations. Notice that $\mathbf{y}$ need not be a binary vector since each location can be chosen more than once. There are $|\mathfrak{R}^m|$ such equations and we cannot write this as a system of fewer equations as for each satisfying location $\hat{r}$ for a challenge-response pair $(\mathbf{c}, r)$, r is independent of $\hat{r}$. This is true since each element of $\mathbf{c}$ is generated uniformly at random from $\{0, 1, \ldots, d-1\}$.

We see that for, $1 \leq i \leq m$, the expected value $E[|\mathfrak{R}_i|]$ is n/d. This implies that $E[|\mathfrak{R}^m|] = (n/d)^m$. There is exactly one element in $\mathfrak{R}^m$ that contains the satisfying location corresponding to $\mathcal{H}$'s secret. We denote this by $\hat{\mathbf{r}}_s$. For more compact notation, define $\mathbf{s} = [\mathbf{x}\,\mathbf{y}]^T$ and $C' = [aC\; C]$. Finding a unique $\mathbf{s}$ satisfying $C'\mathbf{s} = \hat{\mathbf{r}}_s \bmod n$ thus translates into finding the secret locations of $\mathcal{H}$. There are $2n$ unknowns in $\mathbf{s}$. If m is small, the number of solutions for $\mathbf{s}$ is very large. On the other hand, higher m means a higher value of $E[|\mathfrak{R}^m|] = (n/d)^m$, which in turn means more equations to be solved, since $\mathcal{A}$ has no way to distinguish $\hat{\mathbf{r}}_s$ from other elements in $\mathfrak{R}^m$.

Let Solve Equations be an algorithm that finds solutions (possibly multiple) to the linear system defined in Equation 1. Further, let $\tau(n, k, m, d)$ be the time complexity of this algorithm. We have the following probabilistic algorithm for finding $\mathbf{s}$.

Algorithm 1

```
Input: The matrix $C'$ and the set $\mathfrak{R}^m$.
1: Initialize an empty set $\mathfrak{S}$.
2: for each $\hat{\mathbf{r}} \in \mathfrak{R}^m$ do
3:     Run Solve Equations on input $C'\mathbf{s} = \hat{\mathbf{r}} \bmod n$. If any solutions are
       found, assign them to the set $\mathfrak{S}$.
4: Output an element uniformly at random from $\mathfrak{S}$.
```

This algorithm will perform well probabilistically if $|\mathfrak{S}|$ is small. As we have found before, the expected size of $\mathfrak{R}^m$ is $(n/d)^m$. There are a total of n^m possible different output vectors for $C'\mathbf{s} \bmod n$. Therefore, the probability that an $\mathbf{s}$, different from $\mathcal{H}$'s secret, is in the set $\mathfrak{S}$ can be estimated as $\frac{(n/d)^m}{n^m} = 1/d^m$. Thus this probability becomes lower as m increases. Therefore, we can assume that the performance of this algorithm is good. The expected time complexity of Algorithm 1 is:

$$O\left(\tau\left(n, k, m, d\right)\left(\frac{n}{d}\right)^m\right)$$

If gaussian elimination is used as the Solve Equations algorithm, we require $m \geq 2n$, but this means that the expected size of $\mathfrak{R}^m$ will be greater than or equal to $(n/d)^{2n}$. We can see the complexity of this algorithm with concrete values. Let $n = 100$, $d = 10$ and $k = 16$. Then, we have $m \geq 2n = 200$. Gaussian elimination takes time $O(n^3)$ giving a total approximate time 2^{687}, which is surely infeasible.

Since the weights of $\mathbf{x}$ and $\mathbf{y}$ are restricted, we might still be able to use other methods with a smaller value of m. To this end, given m, we find the number of possible solutions of the following equation:

$$C'\mathbf{s} = \hat{\mathbf{r}} \bmod n \tag{2}$$

where $\hat{\mathbf{r}} \in \mathfrak{R}^m$. Since $\mathsf{wt}(\mathbf{x}) = 1$ and $\mathsf{wt}(\mathbf{y}) = k$, with $m = 0$ there are a total of $n\binom{n+k-1}{k}$ possible choices for $\mathbf{s}$. With $m = 1$, we expect a $1/n$ fraction of these choices to satisfy the above equation. Continuing on this way, we see that the expected number of possible choices for $\mathbf{s}$ are:

$$\frac{n\binom{n+k-1}{k}}{n^m} = \frac{\binom{n+k-1}{k}}{n^{m-1}}$$

Since the expected size of $\mathfrak{R}^m$ is $(n/d)^m$, we see that the combined expected number of solutions are:

$$\frac{n\binom{n+k-1}{k}}{d^m}$$

Equating the above expression to 1, we get:

$$m = \log_d n + \log_d \binom{n+k-1}{k} \tag{3}$$

Thus $m = O(\log_d n + \log_d \binom{n+k-1}{k})$ is required on average to find a unique value of $\mathbf{s}$ in Equation 2. By using the concrete values as above, we find that the resulting value of m from Equation 3 is approximately 29. Thus in theory, we can have an algorithm that solves the problem with $m = 29$. However, this value of m implies that $(n/d)^m \approx 2^{96}$. Thus whether or not a Solve Equations algorithm that works for smaller values of m can be found, the overall time complexity of Algorithm 1 is still very high. Thus it is not possible to improve this complexity without a different approach. Informally speaking, the main reason for this interesting result is that in our protocol, the computations are also done on the location indices and not just the digits corresponding to these locations as in previously proposed protocols. Since the digits are generated uniformly at random, the final answer is not linearly dependent on the location indices. Next, we present a time-space tradeoff algorithm that utilizes fewer challenge-response pairs and has better time complexity.

5.3 Time-Memory Tradeoff

In [1, §6], Hopper and Blum sketched a meet-in-the-middle algorithm which, on k-out-of-n protocols, has average-case time complexity of:

$$O\left(n^{k\left(1-\frac{\ln d}{2\ln Q}\right)}\right) \tag{4}$$

Here Q is an intermediate result, which in our protocol corresponds to the range of the intermediate locations during the computation of the protocol. Thus in our protocol, $Q = n$. Our protocol and the protocols from [1] as well as many other protocols in literature loosely fall in this category of protocols (the only difference in our protocol is that we have a starting location that is computed differently from the k remaining locations). For the k-weight HB protocol, the average-case time complexity of this attack is: $O(\binom{n}{k/2})$ [1, §3.1, pp. 58]. This is true due to two reasons. First the protocol uses the addition operation, which is commutative, and the user has to choose k *unique* locations as a secret. Therefore, the number of possible secrets are $\binom{n}{k}$ instead of n^k. Secondly, Q equals d in their protocol. Similarly, for the sum of k mins protocol the average-case time complexity of this attack is $O(\binom{n(n-1)/2}{k/2})$[1, §3.2, pp. 59]. However, since the size of the secret is exactly twice than in the k-weight HB protocol, the comparative time complexity is $O(\binom{n(n-1)/2}{k/4})$.

This attack is essentially a time-memory tradeoff. The time-memory tradeoff attack that we present here employs a deterministic *baby-step giant-step* algorithm by Coppersmith, summarised in [16, pp. 109] and detailed in [14, §2.1]. On k-out-of-n protocols, the resulting attack has average-case time complexity of:

$$O\left(\frac{n^{k\left(1-\frac{\ln d}{2\ln Q}\right)}n^{\frac{\ln d}{\ln Q}}}{2^{\frac{k}{2}\frac{\ln d}{\ln Q}}}\right)$$

which is better than the former if $2^{k/2} < n$ or $k < 2\log_2 n$. The space complexities of the two attacks are the same. While the time complexity is comparable to the previously mentioned meet-in-the-middle attack for generic k-out-of-n protocols, our attack however, performs much better on the two protocols in [1]. The average-case time complexity of our algorithm on the k-weight HB protocol is $O(\binom{n/2}{k/2})$ and on the sum of k mins protocol is $O(\binom{n(n-1)/4}{k/4})$. This is substantially smaller than the previous result.

The original application of Coppersmith's algorithm is to solve the restricted hamming weight discrete logarithm problem [15]. But since this algorithm essentially utilizes the knowledge of the restricted hamming weight, we can modify it to solve our problem. We notice that there are several other deterministic algorithms that perform asymptotically better than Coppersmith's algorithm like the one proposed by Stinson of time complexity $O\left(k^{3/2}(\ln n)\binom{n/2}{k/2}\right)$ [14]. However, for the choice of parameters used in this paper, the performance is comparable to Coppersmith's algorithm if not worse. There are also some probabilistic variants, but which cannot be applied here since the discrete logarithm is always unique and this is not necessarily the case with the candidate locations in our problem for small values of m. For larger values of m time-memory tradeoff algorithms become infeasible. We now describe the attack on our protocol and derive its time complexity. The derivations of the other results mentioned above are similar.

For simplicity, we assume n and k to be even integers. For arbitrary n and k, the attack can be carried out with minor differences [14, §5]. For $0 \leq i \leq$

$n-1$, define $\mathbf{b}_i$ to be a vector of length n such that $\mathbf{b}_i[l+1] = 1$ whenever, $l \equiv i + j \bmod n$, for $0 \le j \le n/2 - 1$, and 0 otherwise. Clearly, for all i, $\mathbf{b}_i$ is a binary vector with $\mathsf{wt}(\mathbf{b}_i) = n/2$. Let $\mathfrak{B} = \{\mathbf{b}_i : 0 \le i \le n/2 - 1\}$. Now, let $\mathfrak{Y}$ be the set of all n-element vectors. From [14, §2.1], we see that for all $\mathbf{y} \in \mathfrak{Y}$ with $\mathsf{wt}(\mathbf{y}) = k$, there exists a $\mathbf{b} \in \mathfrak{B}$, such that:

$$\mathbf{y} \cdot \mathbf{b} = \frac{k}{2}$$

For any $\mathbf{y}_1, \mathbf{y}_2 \in \mathfrak{Y}$, $\mathbf{y}_2$ is called the *sub* of $\mathbf{y}_1$, denoted $\mathbf{y}_2 \prec \mathbf{y}_1$, if $\mathbf{y}_1[l] = 0 \Rightarrow \mathbf{y}_2[l] = 0$ for $1 \le l \le n$. Let $\mathbf{1}$ denote the binary vector of weight n. Let $\mathbf{y}_{=k/2}$ denote a vector whose weight is $k/2$. We divide $\mathbf{s}$ into two parts: $\mathbf{s}_1 = [\,\mathbf{x}\ \mathbf{y}_{=k/2}\,]^T$ and $\mathbf{s}_2 = [\,\mathbf{0}\ \mathbf{y}_{=k/2}\,]^T$. We assume there to be a hash table, initially empty, which will be used as a data structure in this attack.

Algorithm 2

```
Input: The set $\mathfrak{B}$ and $m$ challenge-response pairs.
 1: Initialize an empty set $\mathfrak{S}$.
 2: for $0 \le i \le n/2 - 1$ do
 3:     for each possible vector $\mathbf{s}_1 = [\,\mathbf{x}\ \mathbf{y}_{=k/2}\,]^T$ such that $\mathbf{y}_{=k/2} \prec \mathbf{b}_i$ do
 4:         Compute the string $q \leftarrow \mathbf{c}_1 \cdot \mathbf{s}_1 \bmod n || \cdots || \mathbf{c}_m \cdot \mathbf{s}_1 \bmod n$.
 5:         Insert this $m$-digit string $q$ along with $\mathbf{s}_1$ in the hash table.
 6:     for each vector $\mathbf{s}_2 = [\,\mathbf{0}\ \mathbf{y}_{=k/2}\,]^T$ such that $\mathbf{y}_{=k/2} \prec \mathbf{1} - \mathbf{b}_i$ do
 7:         for $1 \le i \le m$ do
 8:             Initialize an empty set $\mathfrak{Q}_i$.
 9:             For $1 \le j \le n$, if $r_i \equiv (j + \mathbf{c}_i \cdot \mathbf{s}_2) \bmod n$, update $\mathfrak{Q}_i \leftarrow \{j\} \cup \mathfrak{Q}_i$.
10:         Insert each string $q \in \mathfrak{Q}_1 \times \cdots \times \mathfrak{Q}_m$ in the hash table along with
            $\mathbf{s}_2$.
11: For each collision in the hash table, construct $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$ and update $\mathfrak{S} \leftarrow$
    $\{\mathbf{s}\} \cup \mathfrak{S}$.
12: Output $\mathfrak{S}$.
```

Once Algorithm 2 is executed, we need another algorithm to uniquely determine the vector in $\mathfrak{S}$ that satisfies $m \ge \log_d(n\binom{n+k-1}{k})$ challenge-response pairs.

Algorithm 3

```
Input: The set $\mathfrak{S}$ and $m \ge \log_d(n\binom{n+k-1}{k})$ challenge-response pairs.
 1: for each $\mathbf{s} \in \mathfrak{S}$ do
 2:     If $\mathbf{c}_i \cdot \mathbf{s} \equiv r_i \bmod n$ for $1 \le i \le m$, output $\mathbf{s}$ and halt.
```

The memory requirement of Algorithm 2 is $O(n\binom{n+k/2-1}{k/2})$. Neglecting logarithmic terms, the combined running time of Algorithm 2 and 3 is:

$$O\left(\frac{n^{m+1}}{d^m}\binom{n/2+k/2-1}{k/2} + \frac{\binom{n+k-1}{k}}{d^m}\right)$$

Following a similar procedure to that of [1, §6], we see that to minimize this quantity the optimum value of m for Algorithm 2 is:

$$m = \frac{\ln\left(\frac{\binom{n+k-1}{k}\ln d}{\binom{n/2+k/2-1}{k/2}\ln(n/d)}\right)}{\ln n} - 1$$

And this value of m gives the running time:

$$O\left(\binom{n+k-1}{k}^{1-\ln d/\ln n}\binom{n/2+k/2-1}{k/2}^{\ln d/\ln n}\right)$$

In contrast, if we use the meet-in-the-middle algorithm from [1], we get a running time in which $n/2$ is replaced by n in the second term above; hence considerably larger in the second term. Table 1 shows various choices of the parameters n and k and the resulting combined time and space complexity of Algorithms 2 and 3. We assume $d = 10$ and the time and space complexities are represented by the symbols τ and μ respectively.

Table 1. The time complexity τ and space complexity μ of the time-memory tradeoff attack with the optimum value of m against n and k

n	k	τ	μ	n	k	τ	μ	n	k	τ	μ
100	10	2^{33}	2^{33}	120	10	2^{36}	2^{35}	150	10	2^{39}	2^{37}
	20	2^{55}	2^{52}		20	2^{60}	2^{55}		20	2^{65}	2^{58}
	30	2^{72}	2^{67}		30	2^{79}	2^{71}		30	2^{86}	2^{76}
200	10	2^{43}	2^{39}	300	10	2^{48}	2^{43}	500	10	2^{55}	2^{47}
	20	2^{72}	2^{63}		20	2^{83}	2^{69}		20	2^{96}	2^{77}
	30	2^{97}	2^{83}		30	2^{112}	2^{92}		30	2^{132}	2^{104}

5.4 Comparative Time Complexities

The main motivation behind our protocol was to increase Q relative to d in Equation 4 without compromising too much on usability. If Q roughly equals the square of d, then we can choose smaller values of k, as the time complexity of the attack will increase. This is not straightforwardly possible in the k-weight HB protocol and the sum of k mins protocol. For instance, if $Q = 100 = 10^2 = d^2$, these protocols will require k additions of 2 digit numbers in a single round. This is prohibitively difficult for most humans since the additions have to be performed mentally. Our protocol achieves this by shifting the computations to the locations rather than the values of those locations. At each step, the user only has to add a 2 or 3 digit number to a single digit number. As a result, usability is preserved while the time-memory tradeoff attacks perform worse in our case. Table 2 shows a direct comparison of the three protocols in terms of the time-complexity of our attack. The time complexity of the meet-in-the-middle attack from [1] is labeled "Old", whereas our attack is labeled "New". As can be seen, our attack is more efficient than the previous attack by a few orders of magnitude. Time-memory tradeoff attacks is one way to attack our protocol. The next section looks at a different way to attack the protocol.

Table 2. The time complexities of the time-memory tradeoff attacks on the k-weight HB protocol, the sum of k mins protocol and our protocol

		k-weight HB		Sum of k mins		Our Protocol	
n	k	Old	New	Old	New	Old	New
100	8	2^{22}	2^{18}	2^{24}	2^{22}	2^{30}	2^{28}
	12	2^{30}	2^{24}	2^{34}	2^{31}	2^{41}	2^{38}
	16	2^{37}	2^{29}	2^{45}	2^{41}	2^{51}	2^{47}
200	8	2^{26}	2^{22}	2^{28}	2^{26}	2^{37}	2^{36}
	12	2^{36}	2^{30}	2^{40}	2^{37}	2^{52}	2^{49}
	16	2^{46}	2^{37}	2^{53}	2^{49}	2^{65}	2^{61}
300	8	2^{28}	2^{24}	2^{30}	2^{28}	2^{42}	2^{40}
	12	2^{40}	2^{34}	2^{44}	2^{41}	2^{58}	2^{56}
	16	2^{50}	2^{42}	2^{57}	2^{53}	2^{73}	2^{70}

5.5 Significance of the Jump Constant a

Let $\hat{r}$ denote a location. Clearly it is an integer modulo n. We first attempt to find the probability distribution of obtaining $\hat{r}$ as a sum of the values of k locations, ignoring the starting location and hence the jump constant a. To this end, let $p(k, \hat{r})$ be the probability that $\hat{r}$ is the final location after the sum of the values of k locations as in our protocol. In other words, it denotes the probability that $\hat{r}$ is the sum of k integers (not necessarily unique): $s_1, \ldots, s_k \in \mathbb{Z}_d$. Clearly, $p(1, \hat{r}) = 1/d$ for $0 \leq \hat{r} \leq d-1$ and $p(1, \hat{r}) = 0$ for $d \leq \hat{r} \leq n-1$. For any subsequent k, we see that the probability $p(k, \hat{r})$ can be obtained by:

$$p(k, \hat{r}) = \sum_{i=0}^{n-1} p(k-1, \hat{r} - i \bmod n) p(1, i)$$

This is similar to [1, §3.2]. By a dynamic programming algorithm of time complexity $O(kn^2)$ we can compute these probabilities. Now, let $q(a, k, \hat{r})$ denote the probability of obtaining the location $\hat{r}$ as the sum of k integers and the starting location, thus including the jump constant a. Then, we can see that:

$$q(a, k, \hat{r}) = \frac{1}{d} \sum_{i=0}^{d-1} p(k, \hat{r} + ia \bmod n)$$

Let U denote the uniform distribution over $\mathbb{Z}_n$. Let Q denote the distribution of the $q(a, k, \hat{r})$'s. $\Delta(Q, U) = \frac{1}{2} \sum_{i=0}^{n-1} |q(a, k, i) - \frac{1}{n}|$ is defined as the statistical distance between the two probability distributions. We can then see that this

Table 3. The statistical distance $\Delta(Q,U)$ against n and k. Greater value of n requires larger value of k to make $\Delta(Q,U)$ small

n	k	$\Delta(Q,U)$	n	k	$\Delta(Q,U)$	n	k	$\Delta(Q,U)$
100	10	3.5×10^{-16}	120	10	4.9×10^{-8}	150	10	1.0×10^{-4}
	20	7.0×10^{-16}		20	3.3×10^{-15}		20	1.6×10^{-8}
	30	9.9×10^{-16}		30	1.0×10^{-15}		30	2.5×10^{-12}
200	10	7.2×10^{-3}	300	10	9.7×10^{-2}	500	10	3.3×10^{-1}
	20	8.2×10^{-5}		20	1.5×10^{-2}		20	1.7×10^{-1}
	30	9.3×10^{-7}		30	2.3×10^{-3}		30	8.8×10^{-2}

distance is minimum if $a = \frac{n}{d}$. Table 3 shows $\Delta(Q,U)$ with different values of n and k (a is chosen such that $n = ab = ad$). Based on these results, we see that if the statistical distance is small, an adversary can distinguish from the uniform distribution after observing $\frac{1}{\Delta(Q,U)}$ challenge-response pairs on average. The adversary can then (possibly) use some statistical methods to guess the starting location and can then guess the answer by choosing a location in the region which is more probable to contain the answer. Notice that each session in our protocol consists of m rounds. Therefore, in light of the discussion above, we mandate the use of our protocol for $\frac{1}{m\Delta(Q,U)}$ sessions only, before secret renewal.

6 Usability

To demonstrate comparable usability, we use similar parameters as used in the experiment in [1]. We use $n = 200$, $k = 15$ and $m = 6$. With these parameters, the time and space complexity of our time-memory tradeoff attack is proportional to 2^{61} and 2^{54}, respectively. The statistical distance $\Delta(Q,U)$ is 4.9×10^{-4}, which means that the quantity $\frac{(\Delta(Q,U))^{-1}}{m} \approx 340$. Thus, with these parameters our scheme can be used securely for at least 340 authentication sessions.

With $n = 200$, $k = 15$ and $m = 7$, the experiment done for the k-weight HB protocol by the authors in [1] gave an average time of 166 seconds. Notice that there are $m = 7$ rounds instead of 6. This is important to add noise into the answer. The user sends the wrong answer to one of the challenges. To compare with our protocol, we can see that apart from the starting location, the user has to add two numbers for each secret location. One of these numbers is in the range $[0, 199]$ and the other is in the range $[0, 9]$. Thus arguably, adding a single digit number to a number in the range $[0, 199]$ will take approximately the same time, as we are well versed with doing such computations in our heads. For the starting location, we see that the user can move vertically (in a circular way) according to the digit corresponding to the starting location. The user reaches to a new location this way. The result of the remaining $k = 15$ locations can then be divided by 20 to get a quotient and a remainder. The quotient is the number of vertical steps and the remainder is the number of horizontal steps to be taken.

The user can then follow these steps and output the digit corresponding to the location thus reached. Thus while this last step takes more time than the other steps, it can surely be done within half the time required for the computation of the $k = 15$ other secret locations. Now, one round of k-weight HB protocol takes $166/7 \approx 23.7$ seconds on average. This implies that according to our argument, the computation of the last part takes ≈ 12 seconds. Thus, conjecturing that the calculations for the remaining $k = 15$ locations amounts to time 166, we can see that for $m = 6$ rounds, this amounts to a total time of ≈ 213 seconds.

From this discussion, we can say that for low values of α and β, our protocol is approximately $(\alpha, \beta, 213)$-human executable. While it takes slightly more time than the k-weight HB protocol, it is more usable as the user does not have to send a wrong answer with probability 1/7, which is not possible for most humans. Furthermore, for these parameter choices, the time complexity of the time-memory tradeoff attack is proportional to 2^{37} in the case of k-weight HB protocol. In our case, the complexity is 2^{61}. The comparative time-complexity of the attack on the sum of k mins protocol with similar parameters is 2^{49}. Again, lower than the time-complexity for our protocol. To increase usability, one can further reduce m from 6 to 4 and get a time of approximately 143 seconds. Some of the common authentication mechanisms, such as PIN number authentication, use 4-digit numbers for security. We acknowledge the absence of actual experiments on users.

Finally, to handle human errors we can require the user to only answer correctly in most of the rounds. For instance, if $m = 6$, then the server accepts the user if 5 or more of the answers are correct.

6.1 Suggested Parameters

Table 4 shows choices of parameter values for different security requirements and the resulting parameterized security against different attacks. In the table, m stands for the number of iterations (rounds) in one authentication session. R stands for the success probability of the random guess attack. B stands for the complexity of the brute force attack. τ/μ shows the time/space complexity of the time-space tradeoff algorithm. Finally, *Sessions*, represents the number of authentication sessions a particular secret can be used. It is obtained as $\frac{(\Delta(Q,U))^{-1}}{m}$. Notice that, space complexity can be a severe limitation as well ($2^{63}/8 \approx 10^{18}$, i.e. about 1 eta byte). For low and medium level security, the restriction on the number of sessions can be relaxed.

Notice that in comparison with some other protocols found in literature, the number of sessions is quite high. For instance, the cognitive authentication scheme of Weinshall [10] can only be used for approximately 40 sessions even when the size of the user's secret is as large as 150 [11, §4.1]. For more practical sizes of the secret, the number of allowable sessions is even lower. For these reasons, we have restricted our comparison to the two protocols in [1].

Table 4. Suggested Parameters

Security	n	k	m	R	B	τ/μ	$\Delta(Q, U)$	Sessions
Low	200	12	4	10^{-4}	2^{71}	$2^{49}/2^{44}$	2.9×10^{-3}	85
Medium	200	16	4	10^{-4}	2^{87}	$2^{61}/2^{54}$	4.9×10^{-4}	500
High	200	20	6	10^{-6}	2^{101}	$2^{72}/2^{63}$	8.2×10^{-5}	$2,000$
Paranoid	200	24	6	10^{-6}	2^{114}	$2^{83}/2^{71}$	1.4×10^{-5}	$12,000$

7 Conclusion

Recently, many human identification protocols secure against passive adversaries have been proposed in the literature. However, they can only be used for a few authentication sessions before secret renewal. Furthermore, many of them lack a detailed security analysis. Some of them have ignored the impact of time-memory tradeoff attacks. We attempted to construct a protocol with security against time-memory tradeoff attacks in mind. The resulting protocol offers reasonable usability and good security. We acknowledge that the protocol can not be used frequently, as authentication seems to require about 2-3 minutes time. However, it can be used under certain circumstances such as when the user is using an insecure computer. An interesting question is whether improvements can be made to find a solution that achieves better security with progressively smaller values of parameters such as the size of the secret. Another area of interest is to find other variants of time-memory tradeoff attacks that can be applied to human identification protocols.

Acknowledgements. Hassan Jameel Asghar was supported by Macquarie University Research Excellence Scholarship (MQRES). Josef Pieprzyk was supported by the Australian Research Council under Grant DP0987734. The work of H. Wang is supported in part by the Australian Research Council under ARC Discovery Project DP0665035, the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03 and the Singapore Ministry of Education under Research Grant T206B2204.

References

1. Hopper, N.J., Blum, M.: Secure Human Identification Protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
2. Jameel, H., Shaikh, R.A., Lee, H., Lee, S.: Human Identification Through Image Evaluation Using Secret Predicates. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 67–84. Springer, Heidelberg (2006)
3. Jameel, H., Shaikh, R., Hung, L., Wei, Y., Raazi, S., Canh, N., Lee, S., Lee, H., Son, Y., Fernandes, M.: Image-feature based human identification protocols on limited display devices. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 211–224. Springer, Heidelberg (2009)

4. Matsumoto, T., Imai, H.: Human Identification through Insecure Channel. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 409–421. Springer, Heidelberg (1991)
5. Jermyn, I., Mayer, A., Monrose, F., Reiter, M., Rubin, A.: The design and analysis of graphical passwords. In: 8th USENIX Security Symposium (1999)
6. Wang, C.H., Hwang, T., Tsai, J.J.: On the Matsumoto and Imai's Human Identification Scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 382–392. Springer, Heidelberg (1995)
7. Matsumoto, T.: Human-computer cryptography: An attempt. In: 3rd ACM Conference on Computer and Communications Security, pp. 68–75. ACM Press, New York (1996)
8. Li, X.-Y., Teng, S.-H.: Practical Human-Machine Identification over Insecure Channels. Journal of Combinatorial Optimization 3, 347–361 (1999)
9. Li, S., Shum, H.-Y.: Secure Human-computer Identification against Peeping Attacks (SecHCI): A Survey. Unpublished report, available at Elsevier's Computer Science Preprint Server (2002)
10. Weinshall, D.: Cognitive Authentication Schemes Safe Against Spyware (Short Paper). In: 2006 IEEE Symposium on Security and Privacy, pp. 295–300 (2006)
11. Golle, P., Wagner, D.: Cryptanalysis of a Cognitive Authentication Scheme. Cryptology ePrint Archive, Report 2006, /258, http://eprint.iacr.org/
12. Bai, X., Gu, W., Chellappan, S., Wang, X., Xuan, D., Ma, B.: PAS: Predicate-Based Authentication Services Against Powerful Passive Adversaries. acsac. In: 2008 Annual Computer Security Applications Conference, pp. 433–442 (2008)
13. Li, S., Shum, H.-Y.: Secure human-computer identification (interface) systems against peeping attacks:SecHCI. IACR's Cryptology ePrint Archive: Report 2005/268 (August 2005)
14. Stinson, D.: Some Baby-Step Giant-Step Algorithms for the Low Hamming Weight Discrete Logarithm Problem. Math. Comp. 71, 379–391 (2002)
15. Agnew, G., Mullin, R., Onyschuk, I., Vanstone, S.: An Implementation for a Fast Public-Key Cryptosystem. J. Cryptography 3 (1991)
16. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
17. Li, S., Asghar, H.J., Pieprzyk, J., Sadeghi, A.-R., Schmitz, R., Wang, H.: On the Security of PAS (Predicate-Based Authentication Service). In: ACSAC '09: Proceedings of the 2009 Annual Computer Security Applications Conference, pp. 209–218 (2009)

Secure Sketch for Multiple Secrets

Chengfang Fang[1], Qiming Li[2], and Ee-Chien Chang[1,⋆]

[1] School of Computing, National University of Singapore
{c.fang,changec}@comp.nus.edu.sg
[2] Institute for Infocomm Research, Singapore
Qiming.Li@ieee.org

Abstract. Secure sketches are useful in extending cryptographic schemes to biometric data since they allow recovery of fuzzy secrets under inevitable noise. In practice, secrets derived from biometric data are seldom used alone, but typically employed in a multi-factor or a multimodality setting where multiple secrets with different roles and limitations are used together. To handle multiple secrets, we can generate a sketch for each secret independently and simply concatenate them. Alternatively, we can "mix" the secrets and individual sketches, for example, by taking the first secret as the key to encrypt the sketches of all other secrets. Hence, it is interesting to investigate how the secrets are to be mixed so as to cater for different requirements of individual secrets. We found that, by appropriate mixing, entropy loss on more important secrets (e.g., biometrics) can be "diverted" to less important ones (e.g., password or PIN), thus providing more protection to the former. On the other hand, we found that mixing may not be advisable if the amount of randomness invested in sketch construction is large, or the sketch contains high redundancy, or all secrets are of the same importance. Our analysis provides useful insights and guidelines in the applications of secure sketches in biometric systems.

Keywords: Multi-factor authentication, biometric security, sketch construction.

1 Introduction

Biometrics is potentially useful in building secure and easy-to-use security systems, since it is tightly bound to identities, cannot be easily forgotten or lost. However, these features can also make user credentials based on biometric measures hard to revoke, since once the biometric data of a user is compromised, it would be very difficult to replace it, if possible at all. A key challenge in protecting biometric data as user credentials is that they are fuzzy, in the sense that it is not possible to obtain exactly the same data in two measurements. This renders traditional cryptographic techniques used to protect passwords and keys inapplicable.

⋆ Chang is supported by Grant R-252-000-413-232/422/592 from Temasek Defence Systems Institute (TDSI).

Secure sketches [6] are a recently proposed cryptographic primitive that can be used, in conjunction with other cryptographic techniques, to extend classical cryptographic techniques to fuzzy secrets, including biometric data. The key idea is that, given a secret x, we can compute some auxiliary data p, which is called a *sketch*. The sketch p will be able to correct errors from a noisy version of x and recover the original data x that was enrolled. From there, typical cryptographic schemes such as one-way hash functions can then be applied on x. In particular, an *extractor* can be further applied on the data to obtain a nearly-uniform key of certain length given that the min-entropy of the original data is known. Such a generic method of obtaining a consistent key from fuzzy data is referred to as a *fuzzy extractor*.

The work by Dodis et al. [6] on secure sketches and fuzzy extractors provides a theoretical framework that allows us to analyze the security measured by the *entropy loss* of the sketch, which gives a measure of the amount of information a sketch reveals about the underlying secret. There are also a number of schemes (e.g., [11,10,6,3]) with provable upper bounds on the entropy loss. However, no matter how small the entropy loss is, without additional protections, for most biometric representations, it is inevitable that some important information is revealed.

Biometric data is often employed together with other types of secrets as in a multi-factor setting, or in a multimodal setting where there are multiple sources of biometric data, partly due to the fact that human biometrics is usually of limited entropy. In the context of secure sketches, it is possible to treat these secrets independently: The sketches are generated independently and the final sketch is simply the concatenation of all sketches. The security analysis can also be easily carried out by investigating each secret separately. However, secrets may differ in terms of their entropies and fuzziness. More importantly, they may differ in their roles and constraints in their usage. For example, the likelihood of being lost, stolen or forgotten and the ease of revocation and replacement, would be different for different secrets. Furthermore, when exposed, biometric data, like fingerprints, can be used to infer some sensitive illness information [25] of the person. The straightforward method of combining the secrets independently treats each secret equally, thus may not be able to cater for individual security requirements.

One way to address this issue is to mix different secrets together. By mixing the secrets, we may be able to provide more protection to more important secrets at the expense of reduced protection of others. However, if mixing is not done appropriately, it could reveal more information compared to the straightforward method without mixing. Therefore, a detailed investigation is required.

Let us give a simple example here. Suppose the credential of a user consists of a fingerprint and a password, both of which are known to have relatively low entropies. Intuitively, to provide more protection to the important fingerprint, one could use the password to encrypt the fingerprint's sketch. Now, a few questions to address are: How to quantify the additional protection provided? Does the method really provide additional protection, i.e. are there situations where more information is leaked by inappropriate mixing?

In this paper, we propose and analyze a cascaded mixing approach which is essentially the same as described above: use the less important secret to mix with the sketch of the more important secret. As the leftover entropies of password might be low, it is feasible for an adversary to carry attack by enumerating all likely passwords. Hence, we do not rely on the assumption that the mixing function is computationally one-way. Instead, we focus on information-theoretic aspect of the mixing. To address the question on how to quantify the security, note that if we treat the two secrets as a single secret and investigate the combined leftover entropy $\widetilde{\mathbf{H}}_\infty((X,K)|Q)$, where X, K and Q are random variables of the biometrics data, password and the final sketch respectively, the simple method of concatenating the two secrets could already be optimal. Hence, to capture the additional protection, we investigate the individual leftover entropy $\widetilde{\mathbf{H}}_\infty(X|Q)$ and $\widetilde{\mathbf{H}}_\infty(K|Q)$.

We show that, if the sketch construction is deterministic, cascaded mixing can divert the information leakage towards K. Such additional protection is desirable. In the above example, consider a scenario where an adversary happens to obtain some prior knowledge of the more important secret X. Without mixing, the sketch may provide additional information for the adversary to obtain the secret with high probability. By proper mixing, the adversary cannot obtain information of X from the mixed sketch Q, instead, he obtains some information of K.

Consider the second question on whether there are scenarios where mixing is not advisable. We make two observations. Firstly, we found that when there are high redundancies in the sketch, more entropy could be lost compared with the straightforward method of handling the secrets independently. More precisely, the leftover entropy $\widetilde{\mathbf{H}}_\infty(X,K|Q)$ may be less than $\widetilde{\mathbf{H}}_\infty(X,K|P)$, where P is simply the concatenation of the sketches for X and K. This observation is useful as a number of sketch constructions (e.g., [5]) would produce sketches that contain high redundancies but are difficult to compress. In the second observation, we give counter example to show that, when the randomness invested during sketch construction cannot be decoupled from the sketch, there are scenarios where the mixing is an redundant step as it does not provide more protection to the more important secret, i.e. it essentially provides the same protection as the simple concatenation method. Hence given two choices of sketch constructions where one is deterministic and the other is probabilistic, it is advisable to employ the deterministic method to achieve the protection provided by mixing.

Contributions and Organization

We observe that, in some biometric applications, different secrets have different requirements and some secrets require more protection than others. We argue that the straightforward method of constructing the sketch independently is not satisfactory as it does not address such differences.

We propose a cascaded mixing approach to mix the secrets whereby more important secrets are mixed first (Section 4.1). We analyze the approach and show that, if the sketch construction does not involve randomness, the information leakage on the more important secrets will be "diverted" to the less important secrets (Section 5.1, Theorem 2, 3).

We provide counter-examples to demonstrate that, if the sketch construction involves randomness, there are scenarios where mixing function is unable to further protect the more important secret (Section 6.1) and in some cases it leak information of the less important secret (Section 6.2). We also give an intuitive explanation.

Based on our analysis, we provide guidelines in constructing sketches for multiple secrets (Section 7).

2 Related Work

The fuzzy commitment [11] and the fuzzy vault [10] schemes are among the first error-tolerant cryptographic techniques. More recently, Dodis et al. [6] give a general framework of secure sketches and fuzzy extractors, where the security is measured by the entropy loss of the secret given the sketch. They give specific schemes that meet theoretical bounds for Hamming distance, set difference and edit distance respectively. Another distance measure, point-set difference, motivated from a popular representation for fingerprint features, is investigated in a number of studies [5,3,4]. A different approach [14,24,23] focuses on information leakage defined using Shannon entropy on continuous data with known distributions.

There are also a number of investigations on the limitations of secure sketches under different security models. Boyen [1] studies the re-usability of sketches where the concern is whether multiple sketches of the same biometric data reveal sensitive information. This security model is further extended and studied by Boyen et al. [2] and Simoens et al. [20], where the latter work focuses more on privacy issues. Kholmatov et al. [12] and Hong et al. [9] demonstrate such limitations by giving correlation attacks on known schemes.

The idea of using a secret to protect other secrets is not new. Souter et al. [21] propose integrating biometric patterns and encryption keys by hiding the cryptographic keys in the enrollment template via a secret bit-replacement algorithm. Some other methods use password protected smartcards to store user templates [15,19]. Ho et al. [8] propose a dual-factor scheme where a user needs to read out a one-time password generated from a token, and both the password and the voice features are used for authentication. Sutcu et al. [22] study secure sketch for face features and give an example of how the sketch scheme can be used together with a smartcard to achieve better security.

Using only passwords as an additional factor is more challenging than using smartcards, since the entropy of typical user chosen passwords is relatively low [17,7,13]. Monrose [16] presents an authentication system based on Shamir's secret sharing scheme to harden keystroke patterns with passwords. Nandakuma et al. [18] propose a scheme for hardening a fingerprint minutiae-based fuzzy vault using passwords, so as to prevent cross-matching attacks.

3 Formulations and Background

Table [1] summarizes the notation we are going to use in this paper.

Table 1. Table of notations used

X:	Fuzzy secret distributed over space $\mathcal{M}$.
$\mathtt{D}$:	Distance function defined with $\mathcal{M}$.
$\mathbf{H}_\infty(A)$:	Min-entropy of random variable A.
$\widetilde{\mathbf{H}}_\infty(A\|B)$:	Average min-entropy of A given B.
Enc:	Encoder of a known sketch scheme.
P:	The sketch of X, $P = \mathsf{Enc}(X, R)$.
R:	Recoverable random string used in an encoder.
K:	A non-fuzzy secret or a key.
f:	A mixing function.
S:	Recoverable randomness used in f.
Q:	Output of a mixing function, $Q = f(P, K, S)$.
L_A:	The length of variable A, e.g. L_P is the length of sketch P

3.1 Min-Entropy and Entropy Loss

We follow Dodis et al. [6] and use the following definitions of min-entropy and entropy loss.

The *min-entropy* $\mathbf{H}_\infty(A)$ of a discrete random variable A is $\mathbf{H}_\infty(A) = -\log(\max_a \Pr[A = a])$. For two discrete random variables A and B, the *average min-entropy* of A given B is defined as $\widetilde{\mathbf{H}}_\infty(A|B) = -\log(\mathbb{E}_{b\leftarrow B}[2^{-\mathbf{H}_\infty(A|B=b)}])$

The *entropy loss* of A given B is defined as the difference between the min-entropy of A and the average min-entropy of A given B. In other words, the entropy loss $\mathcal{L}(A, B) = \mathbf{H}_\infty(A) - \widetilde{\mathbf{H}}_\infty(A|B)$. Note that for any n-bit string B, it holds that $\widetilde{\mathbf{H}}_\infty(A|B) \geq \mathbf{H}_\infty(A) - n$, which means we can bound $\mathcal{L}(A, B)$ from above by n regardless of the distributions of A and B.

3.2 Secure Sketches and Fuzzy Extractors

Assuming the original secret x is a point in a discrete domain $\mathcal{M}$ with distance function $\mathtt{D}$, a secure sketch scheme consists of two efficient algorithms: An encoder Enc, which computes a sketch p on the given x, and a decoder Dec, which computes an x' given a p and y such that $x' = \mathsf{Dec}(p, y) = x$ if $\mathtt{D}(x, y) \leq t$ for some threshold t.

More formally, let $\mathcal{M}$ be a metric space with distance function $\mathtt{D}$, we have the following definition[1].

[1] Our definition here looks slightly different from that given by Dodis et al. [6] in that we make the randomness invested during encoding more explicit.

Definition 1 ([6]). *An $(\mathcal{M}, t, \gamma)$-sketch scheme consists of two deterministic polynomial-time algorithms* $\mathsf{Enc} : \mathcal{M} \times \{0,1\}^\gamma \to \{0,1\}^*$ *and* $\mathsf{Dec} : \mathcal{M} \times \{0,1\}^* \to \mathcal{M}$ *such that for all $x, y \in \mathcal{M}$ and $r \in \{0,1\}^\gamma$, it holds that* $\mathsf{Dec}(y, \mathsf{Enc}(x, r)) = x$ *when* $\mathtt{D}(x, y) \leq t$. *We call $p = \mathsf{Enc}(x, r)$ the sketch of x. Also, we say that the randomness r is recoverable if for any x and r', if $\mathsf{Enc}(x, r') = \mathsf{Enc}(x, r)$, we have $r = r'$.*

A fuzzy extractor can be built on top of a secure sketch by applying an *extractor* Ext on a random secret, as shown by Dodis et al. [6]. Given a random variable X with sufficient min-entropy, an extractor[2] is able to compute a *nearly uniform* key of a length that is slightly less than the min-entropy of X. Hence, given a secret x, we can use an extractor to obtain a key k from it. When a y that is close to x with respect to $\mathtt{D}$ and t is presented, the original x can be reconstructed and hence the same key can be obtained by applying the same extractor again on the reconstructed x. In this way, fuzzy secrets can be used just the same way as consistent secrets, except that now an additional sketch p has to be stored and used to reconstruct the original secret.

A well adopted approach measures the security of such a scheme by the amount of information revealed by the sketch about the original secret. Formally, for discrete metric space $\mathcal{M}$ with distance function $\mathtt{D}$, the entropy loss of an $(\mathcal{M}, t, \gamma)$-sketch scheme with encoder Enc is defined as follows.

Definition 2. *The entropy loss of an $(\mathcal{M}, t, \gamma)$-sketch scheme is $\mathbf{H}_\infty(X) - \widetilde{\mathbf{H}}_\infty(X \mid P)$ for random variable X on $\mathcal{M}$ and the sketch $P = \mathsf{Enc}(X, R)$.*

Essentially, if a sketch scheme has an entropy loss bounded from above by $\mathcal{L}$, it means that $\mathbf{H}_\infty(X) - \widetilde{\mathbf{H}}_\infty(X|P) \leq \mathcal{L}$ for any distribution of X. It is possible to design an extractor Ext such that $K = \mathsf{Ext}(X)$ is nearly uniform even when P is known, and the length of K can be at least $\widetilde{\mathbf{H}}_\infty(X|P) - \delta$ for a small δ determined by how close the distribution of K is to the uniform distribution[6]. Hence, if an attacker tries to guess the extracted key, the success probability cannot be much better than $2^{-\widetilde{\mathbf{H}}_\infty(X|P)+\delta}$.

Furthermore, let R be the randomness invested by the encoder Enc during the computation of the sketch P, it is not difficult to show (as mentioned in [6]) that when R is recoverable from X and P, we have

$$\mathbf{H}_\infty(X) - \widetilde{\mathbf{H}}_\infty(X|P) \leq L_P - \mathbf{H}_\infty(R) \tag{1}$$

That is, the entropy loss is bounded from above by the difference between the length of P and $\mathbf{H}_\infty(R)$, which is just the length of R if it is uniform. Furthermore, this upper bound is independent of X, hence it holds for any distribution of X.

[2] For example, pair-wise independent hash functions.

The inequality (1) is useful in deriving a bound on the entropy loss, since typically the size of P and $\mathbf{H}_\infty(R)$ can be easily obtained regardless of the distribution of X. This approach is useful in many scenarios where it is difficult to model the distribution of X, for example, when X represents the features of a fingerprint.

4 Secure Sketch for Two Secrets

In some applications the credential of a user consists of two independent secrets. The sources of these secrets can be different. For example, they may be in different metric spaces with different distance functions and thresholds.

A straightforward extension of sketch construction to two secrets is to simply apply two sketch schemes, for the two secrets x_1 and x_2 independently. The final sketch for the two secrets is the concatenation of the sketches p_1 and p_2 computed from x_1 and x_2 respectively. That is, the sketch $p = p_1 \| p_2$, where $\|$ represents concatenation. Furthermore, the final key can be obtained by concatenating the keys k_1 and k_2 extracted from x_1 and x_2 respectively.

Suppose the entropy loss of the first secret given the sketch is at most $\mathcal{L}_1$, and that of the second secret is at most $\mathcal{L}_2$, then it is clear that the overall entropy loss is at most $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$, since the secrets are independent.

As we have mentioned, this straightforward approach is not able to differentiate secrets with different characteristics, and give equal protection to both secrets.

4.1 A Cascaded Mixing Approach

Instead of treating the two secrets independently, it may be desirable to combine different types of secrets to achieve additional security goals. Here we give an alternative sketch construction. Figure 1 illustrates our proposed method.

For secrets x_1 and x_2, we first compute sketches p_1 and p_2 as in the concatenating approach, and extract keys k_1 and k_2 respectively then we encrypt p_1 using k_2 as the key. That is, we compute $q_1 = f(p_1, k_2, s)$, where f is a deterministic function and s is an auxiliary random string. The final sketch q is $q = q_1 \| p_2$.

Let us call f the *mixing function* which serves as an encryption with k_2 as the key. As the leftover entropy of k_2 given p_2 could be low, we should not rely on the computational difficulty in inverting f to protect p_1. Thus, it is important to analyze how much information about the two secrets x_1 and x_2 is revealed.

Let us consider the mixing function $f : \{0,1\}^{L_P} \times \{0,1\}^{L_K} \times \{0,1\}^{L_S} \to \{0,1\}^{L_Q}$ and random variables Q, P, K and S such that $Q = f(P, K, S)$. We require f to have certain properties. First, as an encryption function, f must be invertible.

Definition 3 (Invertibility). *We say that a mixing function f is invertible if there is a function g such that for all $p \in \{0,1\}^{L_P}$, $k \in \{0,1\}^{L_K}$ and $s \in \{0,1\}^{L_S}$, $g(f(p,k,s),k) = p$.*

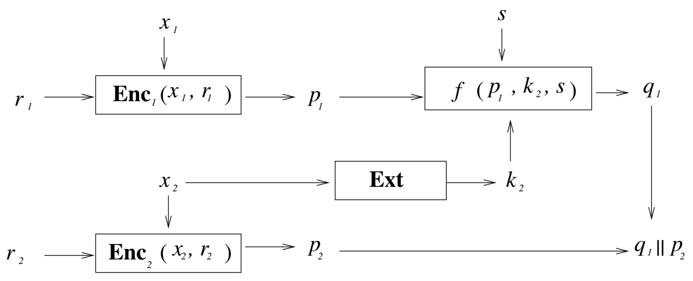


Fig. 1. Construction of cascaded mixing approach

In addition, in our analysis we consider mixing functions with the following properties on recoverability of the randomness invested.

Definition 4 (Recoverable Randomness). *For a mixing function f, the randomness S is called recoverable if for any $p \in \{0,1\}^{L_P}$, $k \in \{0,1\}^{L_K}$ and $s, s' \in \{0,1\}^{L_S}$, if $f(p,k,s) = f(p,k,s')$, we have $s = s'$.*

Definition 5 (β-Recoverable Key). *For a mixing function f, the key K is called β-recoverable if for any $p \in \{0,1\}^{L_P}$, $q \in \{0,1\}^{L_Q}$, the cardinality of the set $\mathcal{K}_{p,q} = \{k \in \{0,1\}^{L_K} | \exists s \in \{0,1\}^{\beta}, f(p,k,s) = q\}$ is at most 2^{β}.*

It is easy to construct mixing function achieving both invertability and recoverability. For example, we can obtain one from a block cipher $f(p,k,r) = r\|E_k(p\|r)$. Note that the recoverability properties are not necessary for the recovery of the secrets, but will become handy in the security analysis.

When a user presents y_1 and y_2 that are close to x_1 and x_2 respectively, x_2 is first reconstructed using y_2 and p_2, and a key k_2 is extracted from x_2, which in turn is used to retrieve p_1 if f is invertible. After that, x_1 is reconstructed using y_1 and p_1. An extractor can be further applied on $x_1\|x_2$ to extract a key.

Intuitively, this alternative approach gives more protection to the first secret x_1, since it would require the attacker to guess x_2 using p_2 first, only when the attacker is successful can the attacker gain information on x_1 from q_1 by computing p_1 from q_1 and x_2.

5 Analysis

We now study the case of two secrets and a scheme that follows the cascaded sketch construction (Section 4.1) Let $x \in \mathcal{M}$ be a fuzzy secret (say, a fingerprint), and let $k \in \{0,1\}^{L_K}$ be an independent secret key that is not fuzzy. Consider a $(\mathcal{M}, t, L_R)$-sketch scheme with encoder Enc, and let the sketch $p = \mathsf{Enc}(x, r)$. Figure 2 illustrates the process.

It is clear that when the key K is uniform and no shorter than the sketch, we can easily hide the sketch p completely (e.g., by using a one-time pad). However,

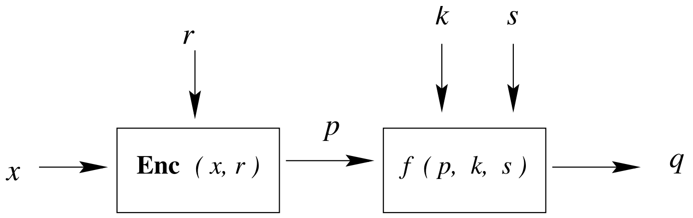


Fig. 2. Computation of mixed sketch

in practical scenarios (e.g., user chosen PIN/password as the key), K can be shorter than p, and the analysis of security may become challenging. In fact, we will show that, for shorter K, mixing is not always a better strategy than the straightforward method of treating the secrets independently. We will also show the conditions under which mixing is desirable.

5.1 Security of the Cascaded Mixing Approach

Analysis of overall remaining entropy $\widetilde{\mathbf{H}}_\infty(X, K|Q)$

First, let us investigate the remaining entropy when we treat (X, K) as a single secret, i.e. the remaining entropy $\widetilde{\mathbf{H}}_\infty(X, K|Q)$.

Lemma 1. *Given random variables X, K, R, S and mixing function f as described above, We have*$\widetilde{\mathbf{H}}_\infty(X, K|Q) \geq \mathbf{H}_\infty(X) + \mathbf{H}_\infty(K) + \mathbf{H}_\infty(R) - L_P$ *.*

Proof: Since S is recoverable, we can consider Enc and f together as the encoding algorithm for the final sketch Q, R and S together as the recoverable randomness, and the inequality (1) in Section 3 applies. Note that $L_Q = L_P + L_S$, and we have

$$\begin{aligned}\widetilde{\mathbf{H}}_\infty(X, K|Q) &\geq \mathbf{H}_\infty(X, K) + \mathbf{H}_\infty(R) + \mathbf{H}_\infty(S) - L_Q \\ &= \mathbf{H}_\infty(X) + \mathbf{H}_\infty(K) + \mathbf{H}_\infty(R) - L_P.\end{aligned}$$

Hence the lemma holds as claimed. □

Lemma 1 gives a lower bound of the remaining entropy of X and K. In general, if both secrets are fuzzy, we can similar obtain the bound:

$$\widetilde{\mathbf{H}}_\infty(X_1, X_2|Q) \geq \mathbf{H}_\infty(X_1) + \mathbf{H}_\infty(X_2) + \mathbf{H}_\infty(R_1) + \mathbf{H}_\infty(R_2) - L_{P_1} - L_{P_2}.$$

where X_1 and X_2 are the secrets, R_1, R_2, are the randomness invested in constructing the sketch P_1, P_2 for the two respective secrets. Note that this bound is the same when we use the straightforward concatenation approach.

Analysis of individual secret $\widetilde{\mathbf{H}}_\infty(X|Q)$ and $\widetilde{\mathbf{H}}_\infty(K|Q)$

Now, let us look at the remaining entropy of individual secret, i.e. $\widetilde{\mathbf{H}}_\infty(X|Q)$ and $\widetilde{\mathbf{H}}_\infty(K|Q)$.

If the sketch is not uniformly distributed, then given the mixed q, it is possible that $(K|Q=q)$ is not uniform. That is, Q will leak some information about K. Indeed, an adversary, given q, may enumerate all possible k's and the correspond sketch p to determine the most likely k. Nevertheless, leakage of K is acceptable as long as it can provide more protection to X. Next theorem gives a lower bound on the remaining entropy of X given the mixed sketch Q.

Theorem 2. *Given three independent random variables X, K and R distributed over $\mathcal{M}$, $\{0,1\}^{L_K}$ and $\{0,1\}^{L_R}$ respectively and an $(\mathcal{M}, t, L_R)$-sketch scheme with encoder* Enc*, Let P be the sketch of X, i.e., $P = \mathsf{Enc}(X, R)$, where R is recoverable, and let $f : \{0,1\}^{L_P} \times \{0,1\}^{L_K} \to \{0,1\}^{L_Q}$ be an mixing function and $Q = f(P, K, S)$, where S is a L_S bits of recoverable randomness. If f is invertible and the key K is L_S-recoverable. Then*

$$\widetilde{\mathbf{H}}_\infty(X|Q) \geq \mathbf{H}_\infty(X) + \mathbf{H}_\infty(K) - L_Q. \tag{2}$$

We would like to refer the reader to Appendix A for the proof of the above theorem.

The theorem holds for any distributions of X and K, and for uniformly distributed K, the theorem implies that $\widetilde{\mathbf{H}}_\infty(X|Q) \geq \mathbf{H}_\infty(X) + L_K - L_Q$. Let us compare the remaining entropy if we use the simple concatenation method, which is as follows,

$$\widetilde{\mathbf{H}}_\infty(X|P) \geq \mathbf{H}_\infty(X) + L_R - L_P \tag{3}$$

Now, coming back to the question that whether it is beneficial to use a cascading function when the secret k is short compared with p. Clearly, from Theorem 2 and inequality (3), we can see that when $\mathbf{H}_\infty(K) - L_Q \geq L_R - L_P$, or equivalently, $\mathbf{H}_\infty(K) \geq L_R + L_S$, the R.H.S in (2) is larger then the R.H.S in (3), i.e. the entropy bound when using a mixing function is no worse than not using it. In particular, consider a deterministic sketch scheme (i.e. L_R =0), and a length preserving mixing function (thus $L_P = L_Q$), the difference in the right hand side of the inequality (2) and (3) is $\mathbf{H}_\infty(K)$. In other words, the bound on leftover entropy of X given Q can be increased by $\mathbf{H}_\infty(K)$. Viewing from another direction, information loss on X is "diverted" to K.

Now, we consider only the non-fuzzy secret k and analyze the entropy loss.

Theorem 3. *Given an $(\mathcal{M}, t, L_R)$-sketch scheme with encoder* Enc*, and let X, K, R, P, Q, f, S be as defined in Theorem 2, we have*

$$\widetilde{\mathbf{H}}_\infty(K|Q) \geq \mathbf{H}_\infty(K) + \mathbf{H}_\infty(R) - L_P. \tag{4}$$

Proof: Since $Q = f(P, K, S)$, we can regard Q as a sketch of K where the cascading function f is an encoder, and $P = \mathsf{Enc}(X, R)$ and S are the "randomness" invested in computing Q, which are recoverable. Clearly, we can apply the general bound (1) on K and Q, and since R is recoverable, we have

$$\mathbf{H}_\infty(X) + \mathbf{H}_\infty(P) \geq \widetilde{\mathbf{H}}_\infty(X, P) \geq \mathbf{H}_\infty(X) + \mathbf{H}_\infty(R)$$

which means that $\mathbf{H}_\infty(P) \geq \mathbf{H}_\infty(R)$, hence the inequality holds as desired. □

It is worth to note that the bound in Theorem 3 is tight in the sense that there exists random variables and functions such that the equality in (4) holds. We will see an example of such case in Section 6.2. Therefore, if L_P is large but the min-entropy $\mathbf{H}_\infty(P)$ is low, the quantity $\mathbf{H}_\infty(K)+\mathbf{H}_\infty(P)-L_P$ may be reduced to 0 or even less than 0, in which case Q may reveal all information about K.

6 Examples of Improper Mixing

In this section we give examples to illustrate the scenarios where mixing function may not be beneficial: (1) in scenarios where the sketch construction employs randomness, mixing function may not always provide protection on X. (2) when the sketch contains high redundancy from the adversary point of view, mixing function may reveal information of K.

6.1 Randomness Invested in Sketch

This section gives a simple example to illustrate the idea that mixing function may not always provide protection on X, if the sketch construction contains randomness. Hence, as a general guideline, when choosing a sketch scheme to be used in the cascaded mixing framework, it is better to select one that requires no randomness.

Consider a non-fuzzy K in $\{0,1\}^{L_K}$, and a fuzzy X in $\{1\dots 2^{L_X}\}$ with the distance function

$$d(x_1,x_2)=\begin{cases}0, & \text{if } x_2=x_1\\ 1, & \text{if } x_2=x_1+1 \mod 2^{L_X}\\ \infty, & \text{otherwise}\end{cases}$$

for any $x_1,x_2\in\{1\dots 2^{L_X}\}$ and the noise threshold is 1. Hence, a noisy copy of an x could be either x or $(x+1) \mod 2^{L_X}$.

Consider the following two sketch constructions: a deterministic construction $\mathsf{Enc}_1(X)=X \mod 2$, and a probabilistic construction $\mathsf{Enc}_2(X,R)=X+R \mod 2^{L_X}$, where R is a uniform random even number in $\{1\dots 2^{L_X}\}$. Without mixing, sketches output from both constructions reveal at most one bit of X.

Given a one bit secret K, let the mixing function $f(P,K,S)$ be as following: it first generates with seed S a set $\mathbf{S}=\langle \mathbf{s}_1,\mathbf{s}_2\rangle$ of random strings of length L_P, then it output $P+\mathbf{k}_K \mod 2^{L_P}$.

Consider the case when Enc_1 is used, the mixing function is one-time pad encryption, by Theorem 2, there will be no entropy loss on X i.e. $\mathbf{H}_\infty(X)-\widetilde{\mathbf{H}}_\infty(X|Q)=0$. However, when Enc_2 is used, there could be cases where $\mathbf{s}_i$ has same parity, for example, $\mathbf{S}=\langle 0,2\rangle$. In that case, the information of the sketch is not protected and $\mathbf{H}_\infty(X)-\widetilde{\mathbf{H}}_\infty(X|Q)=1$ and there is no gain nor loss in mixing the secrets compare to the straightforward method. In other words, the secret K is unable to provide additional protection as desired.

Note that, by Lemma 1, the overall entropies $\widetilde{\mathbf{H}}_\infty(X,K|Q)$ are the same in the aforementioned two cases, as well as in the straightforward method of not mixing the secrets.

Hence, when given two choices of sketch constructions where one is deterministic and the other is probabilistic, it is advisable to employ the deterministic method to achieve the protection provided by mixing function.

6.2 Redundancy in Sketch

When the sketch has redundancy, that is, the entropy of the sketch is smaller than the length of the sketch, information on k will be leaked from the mixed sketch. There are a few known sketch constructions where the "support" of the sketch (i.e. the number of sketches which non-zero probability of occurrences) is significantly smaller than 2^{L_P} where L_P is the length of the sketch and thus their sketches contain redundancy. One example is the chaff-based method [5] proposed to protect the biometric fingerprint. Here, a fingerprint is the secret x and can be represented as a set of 2D points. The chaff-based method gives its sketch which is the original x union with a set of random 2D points, constrained by the requirement that no two points are close to each other (w.r.t Euclidean distance). It is not easy to derive a compact description of the sketch whose support has size close to 2^{L_P}. Now, suppose that the sketch is mixed with a short k. Given a mixed sketch q, it could be highly likely that among all possible K's in inverting q, only one give a point set that satisfies the constrain. Thus, immediately, the secret k and the sketch is revealed, and the remaining entropy of the combined $\widetilde{\mathbf{H}}_\infty(X, K|Q) = \widetilde{\mathbf{H}}_\infty(X|P)$. Hence, by mixing, not only there is no further protection of x, the k is revealed.

We also conducted experiment to illustrate that, even when the description of sketch is compact, i.e. its support equals 2^{L_P}, the chaff-based sketch still contains significant redundancy that leads to lost of information on k.

Consider the chaff-based method for 1D points, which is easy to derive a compact description. We simulated the chaff-based method in $\mathbb{Z}_{24}$ with a minimum distance 3. There are in total 605 possible sketches, and we randomly generated 10^5 sketches. Figure 3 shows the numbers of occurrences for all 605 sketches with x-axis descendingly sorted by the number of occurrence (and we call the position of a sketch in this descending list the *rank* of it).

Suppose the sketch is then protected by a 5 bits key k, and a mixing function f such that the inverts are always valid sketches. We then simulate an adversary who try to guess k when given $q = f(k, p, s)$, where k and s are randomly chosen from their domain and p is chosen according to the distribution approximated by Figure 3. We simulated 10^5 guesses and the adversary can succeed with probability slightly more than 0.052, instead of $1/(2^5) = 0.03125$ as in random guessing.

7 Further Discussions

7.1 The Case of Two Fuzzy Secrets

When both secrets are fuzzy and may not be uniform, we show that the bounds of Lemma 1, Theorem 2 and 3 can be obtained with slight modifications.

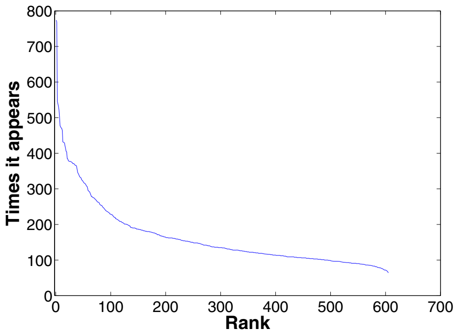


Fig. 3. Histogram of sketch occurrences

Suppose there are two independent secrets $x_1 \in \mathcal{M}_1$ and $x_2 \in \mathcal{M}_2$, and two sketch construction schemes with encoder Enc_1 and Enc_2 respectively. We assume that the first secret x_1 is more important than x_2. In this case, we can use the following steps to construct the sketch for the two secrets.

1. Compute $p_1 = \mathsf{Enc}_1(x_1, r_1)$ and $p_2 = \mathsf{Enc}_2(x_2, r_2)$.
2. Extract a key k_2 from x_2 using an extractor Ext.
3. Compute $q_1 = f(p_1, k_2, S)$ using a mixing function f.
4. Output the final sketch $q = q_1 \| p_2$.

It is possible to design Ext such that K_2 and P_2 are independent, and $\mathbf{H}_\infty(K_2)$ is only slightly smaller than $\widetilde{\mathbf{H}}_\infty(X_2|P_2)$ [6]. Let δ be a small extractor-dependent value such that $\mathbf{H}_\infty(K_2) \geq \widetilde{\mathbf{H}}_\infty(X_2|P_2) - \delta$.

The bound in Theorem 2 still applies on x_1 and k_2. Consider random variables X_1 and K_2, corresponding sketches P_1 and P_2, mixed sketch Q_1, and final sketch Q, it's not difficult to show that $\widetilde{\mathbf{H}}_\infty(X_1|Q) \geq \mathbf{H}_\infty(X_1) + \mathbf{H}_\infty(X_2) + \mathbf{H}_\infty(R_2) - L_{P_2} - \delta - L_Q$ where R_2 is the recoverable randomness used in computing P_2. In this case, the small δ can be considered as the overhead of using the extractor Ext.

As a comparison, if we treat the two secrets independently, and consider $P = P_1 \| P_2$, we have $\widetilde{\mathbf{H}}_\infty(X_1|P) = \widetilde{\mathbf{H}}_\infty(X_1|P_1) \geq \mathbf{H}_\infty(X_1) + L_{R_1} - L_{P_1}$.

Similar to the example, we can conclude that if $\mathbf{H}_\infty(K_2) \geq L_{R_1} + L_S$, we can obtain a better bound on the entropies when we choose to mix k_2 with p_1. Otherwise, doing so may reveal more information about X_1.

The entropy loss on the second secret X_2 can be obtained using the bound in Theorem 3. It's not difficult to show that $\widetilde{\mathbf{H}}_\infty(X_2|Q) \geq \mathbf{H}_\infty(X_2) + \mathbf{H}_\infty(R_2) + \mathbf{H}_\infty(R_1) - L_{P_1} - L_{P_2} - \delta$

The overall entropy loss in Lemma 1 applies to the general case. That is,

$$\widetilde{\mathbf{H}}_\infty(X_1, X_2|Q) \geq \mathbf{H}_\infty(X_1) + \mathbf{H}_\infty(X_2) + \mathbf{H}_\infty(R_1) + \mathbf{H}_\infty(R_2) - L_{P_1} - L_{P_2}.$$

7.2 Cascaded Structure for Multiple Secrets

In some systems, it may be desirable to use more than two secrets. For example, in a multi-factor system, a user credential may include a fingerprint, a smartcard and a PIN, or two fingerprints and a password. Unlike the two secret case, there are many different cascaded strategies to mix the secrets.

Given secrets $x_1, x_2, \cdots, x_s$ and the corresponding sketches $p_1, p_2, \cdots, p_s$, the following are the main strategies to mix them, assuming we have mixing functions $f_1, \cdots, f_{s-1}$.

1. (Fanning) Apply mixing functions f_i on x_1 and p_{i+1} for all $1 \leq 1 \leq s-1$.
2. (Chaining) Apply mixing function f_i on x_i and p_{i+1} for all $1 \leq 1 \leq s-1$.
3. (Hybrid) Use a combination of fanning, chaining and independent encoding. For example, we can mix x_1 with p_2 and p_3, and further mix x_2 with p_4, but x_5 is encoded independently.

With the fanning approach, the entropy loss would be mostly diverted to the first secret, which may be the most easily revocable and replaceable secret. However, this approach requires that the first secret has sufficiently high entropy, since otherwise it may be relatively easy to obtain the first secret from the mixed sketch. In practice, this approach can be used when a long revocable key is available, such as key stored in a smartcard.

On the other hand, using the chaining approach only requires that the entropy of the i-th secret is sufficient to mix with the $(i+1)$-th sketch. In this case, the secrets should be mixed in the order of their "importance", which could be, for example, the ease of revocation and replacement, or the likelihood of being lost or stolen. Note that in this approach, it is crucial to determine the exact order of importance of the secrets.

If no single secret is of sufficient entropy, and the order of importance among secrets is not always clear, a hybrid approach may become more appropriate. As a special case, when all secrets are short and no secret is more important than others, it would not be advisable to use the mixing approach and a straightforward method can be better.

7.3 Guidelines for Applying Mixing Functions on Two Secrets

To summarize, we give some guidelines for the application of cascaded mixing functions to two secrets. The same principles apply to multiple secrets.

1. If the importance of the secrets cannot be determined or is the same for both secrets, mixing is not recommended.

2. For the more important secret, if there are two secure sketch schemes that differ only in the amount of randomness used in the construction; choose the one that uses less randomness.

3. If the randomness invested cannot be decoupled from the sketch, cascaded mixing is not advisable unless the length of consistent key is longer than the length of the sketch.

8 Conclusions

In this paper, we investigate the security of secure sketches and fuzzy extractors that use more than one secret, motivated by the fact that user credentials based on biometric data are seldom used alone, but often combined with other secrets. Since the leftover entropy of each secret is not high and exhaustive search is feasible, we focus on information theoretic results and measure security using min-entropies.

In many practical applications that involve multiple secrets, the secrets may have different characteristics such as their revocability, ease of replacement, and likelihood of being lost or stolen. Hence, they often require different level of protections. However, such differentiation cannot be expressed easily in existing frameworks.

To cater for different security requirements for different secrets, we propose to analyze the security separately for different secrets, and we propose a cascaded mixing approach that combines the secrets when computing the final sketch. We show that under certain conditions, the proposed method provides more protections to more important secrets at the expense of increasing the risk of reduced security on the less important ones.

We show that there are scenarios where the cascaded mixing approach may not be advisable. These include cases where the sketch construction uses a lot of randomness, or the sketch contains a lot of redundancies, or it is difficult to determine the importance of secrets. We illustrate these subtleties with some examples.

We start with the case of two secrets and extend our discussions to the case of more secrets. We also give general guidelines as how these secrets should be mixed in practice.

References

1. Boyen, X.: Reusable cryptographic fuzzy extractors. In: Proceedings ACM Conf. on Computer and Communications Security, October 2004, pp. 82–91 (2004)
2. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005)
3. Chang, E.-C., Li, Q.: Hiding secret points amidst chaff. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 59–72. Springer, Heidelberg (2006)
4. Chang, E.C., Shen, R., Teo, F.W.: Finding the original point set hidden among chaff. In: ACM Symposium on Information, computer and communications security, p. 188 (2006)
5. Clancy, T.C., Kiyavash, N., Lin, D.J.: Secure smartcard-based fingerprint authentication. In: ACM Workshop on Biometric Methods and Applications, pp. 45–52 (2003)
6. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
7. Florencio, D., Herley, C.: A large-scale study of web password habits. In: Proceedings of the 16th international conference on World Wide Web, pp. 657–666. ACM, New York (2007)

8. Ho, P., Armington, J.: A dual-factor authentication system featuring speaker verification and token technology. In: Audio- and Video-Based Biometric Person Authentication, pp. 128–136 (2003)
9. Hong, S., Jeon, W., Kim, S., Won, D., Park, C.: The vulnerabilities analysis of fuzzy vault using password. In: Second International Conference on Future Generation Communication and Networking, FGCN'08, pp. 76–83 (2008)
10. Juels, A., Sudan, M.: A fuzzy vault scheme. In: IEEE Intl. Symp. on Information Theory, pp. 408–421 (2002)
11. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: Proceedings ACM Conf. on Computer and Communications Security, pp. 28–36 (1999)
12. Kholmatov, A., Yanikoglu, B.: Realization of correlation attack against the fuzzy vault scheme. Security, Forensics, Steganography, and Watermarking of Multimedia Contents (January 2008)
13. Klein, D.V.: Foiling the cracker: A survey of, and improvements to, password security. In: 2nd USENIX Security Workshop, pp. 5–14 (1990)
14. Linnartz, J.-P.M.G., Tuyls, P.: New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Kittler, J., Nixon, M.S. (eds.) AVBPA 2003. LNCS, vol. 2688, pp. 393–402. Springer, Heidelberg (2003)
15. Lisimaque, G.: Biometrics and smart cards. In: Proceedings of Conference of the Biometric Consortium (1999)
16. Monrose, F., Reiter, M., Wetzel, S.: Password hardening based on keystroke dynamics. In: Proceedings ACM Conf. Computer and Communications Security, pp. 73–82 (1999)
17. Morris, R., Thompson, K.: Password security: A case history. Communications of the ACM, 594–597 (1979)
18. Nandakumar, K., Nagar, A., Jain, A.K.: Hardening fingerprint fuzzy vault using password. In: Advances in Biometrics International Conference, August 2007, pp. 927–937 (2007)
19. Sanchez-Reillo, R.: Including biometric authentication in a smart card operating system. In: Bigun, J., Smeraldi, F. (eds.) AVBPA 2001. LNCS, vol. 2091, pp. 342–347. Springer, Heidelberg (2001)
20. Simoens, K., Tuyls, P., Preneel, B.: Privacy weaknesses in biometric sketches. In: IEEE Symposium on Security and Privacy, vol. 16. IEEE Computer Society, Los Alamitos (2009)
21. Soutar, C., Roberge, D., Stoianov, A., Gilroy, R., Kumar, B.V.K.V.: Biometric encryption. In: ICSA Guide to Cryptography (1999)
22. Sutcu, Y., Li, Q., Memon, N.: Protecting biometric templates with sketch: Theory and practice. IEEE Transactions on Information Forensics and Security, 503–512 (September 2007)
23. Tuyls, P., Akkermans, A.H.M., Kevenaar, T.A.M., Schrijen, G.J., Bazen, A.M., Veldhuis, R.N.J.: Practical biometric authentication with template protection. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 436–446. Springer, Heidelberg (2005)
24. Tuyls, P., Goseling, J.: Capacity and examples of template-protecting biometric authentication systems. In: Maltoni, D., Jain, A.K. (eds.) BioAW 2004. LNCS, vol. 3087, pp. 158–170. Springer, Heidelberg (2004)
25. Yousefi-Nooraie, R., Mortaz-Hedjri, S.: Dermatoglyphic asymmetry and hair whorl patterns in schizophrenic and bipolar patients. Psychiatry Research, 247–250 (2008)

Appendix A: Proof of Theorem 2

Proof: First, let $\mathcal{K}_{x,q} \subset \{0,1\}^{L_K}$ be the set of secret $k \in \{0,1\}^{L_K}$ such that there exists an $r \in \{0,1\}^{L_R}$ and $s \in \{0,1\}^{L_S}$ so that q can be computed from x, r, k and s. That is,

$$\mathcal{K}_{x,q} = \{k \in \{0,1\}^{L_K} | \exists r, s, f(\mathsf{Enc}(x,r), k, s) = q\}.$$

Since the key of the mixing function f is L_S-recoverable, it is clear that the cardinality $|\mathcal{K}_{x,q}|$ is no more than the number of all possible r's multiplied by 2^{L_S}, where $L_S = L_Q - L_P$. That is, $|\mathcal{K}_{x,q}| \leq 2^{L_R+L_S}$ for any x and q. Now, consider

$$\begin{aligned} A &= 2^{-\widetilde{\mathbf{H}}_\infty(X|Q)-L_R-L_S} \\ &= \sum_q \Pr[Q=q] \max_x \Pr[X=x|Q=q] 2^{-L_R-L_S} \\ &= \sum_q \max_x \Pr[X=x, Q=q] 2^{-L_R-L_S}. \end{aligned}$$

On the other hand, we have

$$B = 2^{-\widetilde{\mathbf{H}}_\infty(X,K|Q)} = \sum_q \max_{x,k} \Pr[X=x, K=k|Q=q].$$

For any $q_0 \in \{0,1\}^{L_Q}$, let us consider

$$\begin{aligned} &\max_x \Pr[X=x, Q=q_0] 2^{-L_R-L_S} \\ &= \max_x \sum_k \Pr[X=x, Q=q_0, K=k] 2^{-L_R-L_S} \\ &\leq \max_x \left(\max_k \Pr[X=x, Q=q_0, K=k] 2^{L_R+L_S} \right) 2^{-L_R-L_S} \\ &= \max_{x,k} \Pr[X=x, Q=q_0, K=k] \end{aligned}$$

The inequality holds because for any x, there will be at most $|\mathcal{K}_{x,q_0}| \leq 2^{L_R+L_S}$ non-zero terms in the summation, hence the sum will be at most $2^{L_R+L_S}$ times the largest term in the summation. As a result, we have

$$A \leq \sum_q \max_{x,k} \Pr[X=x, Q=q, K=k] = B.$$

This is equivalent to

$$\widetilde{\mathbf{H}}_\infty(X|Q) + L_R + L_S \geq \widetilde{\mathbf{H}}_\infty(X,K|Q).$$

By applying the bound on overall entropy loss (Lemma 1), and considering that the recoverable randomness includes the L_R bit R and L_S bit S, we have

$$\widetilde{\mathbf{H}}_\infty(X|Q) \geq \widetilde{\mathbf{H}}_\infty(X,K|Q) - L_R - L_S \geq \mathbf{H}_\infty(X) + \mathbf{H}_\infty(K) - L_Q$$

Therefore the theorem holds as claimed. □

A Message Recognition Protocol Based on Standard Assumptions

Atefeh Mashatan and Serge Vaudenay

The Security and Cryptography Laboratory (LASEC), EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch/

Abstract. We look at the problem of designing Message Recognition Protocols (MRP) and note that all proposals available in the literature have relied on security proofs which hold in the random oracle model or are based on non-standard assumptions.

Incorporating random coins, we propose a new MRP using a pseudorandom function F and prove its security based on new assumptions. Then, we show that these new assumptions are equivalent to the standard notions of preimage resistance, second preimage resistance, and existential unforgeability given that F is a pseudorandom function.

Keywords: Cryptographic Protocols, Authentication, Recognition, Pseudorandom Functions, Pervasive Networks, Ad Hoc Networks.

1 Introduction

Message recognition is a notion that has recently been developed for small devices in ad hoc networks. In particular, the devices have low computational power, low communication bandwidth and low energy resources. Moreover, they are placed in an environment where no pre-established authentic information exists and without the presence of a trusted third party. Although chips in embedded systems are becoming more and more powerful, researchers have always been looking for lower complexity algorithms in the past 30 years.

In 2003, Weimerskirch and Westhoff [WW03], realized that achieving message authentication is not possible under such restrictive assumptions. The notion of *recognition* was later formalized by Hamell et al. [HWGW05] and motivated by Lucks et al. [LZWW08] with the following example. Alice and Bob are two complete strangers. They meet in a party. Therefore, they have the chance to briefly meet in person, before they depart. A few days later, Bob receives some message from a person who claims to be Alice. Now, obviously Bob would like to recognize the source of the message and make sure the message is sent from the same person who introduced herself as Alice in the party, and not from a malicious person, Eve, who is trying to deceive Bob.

As a real life application, one can think of Alice having a contactless smart badge who wants to buy several movies from a shop owned by Bob. For her first movie, she walks into the shop and pays using her contactless smart badge.

She would like to download the movie to her computer when she goes home. Moreover, she would like to buy and download more movies from home without having to walking to the shop in person. That is, Alice would like Bob to recognize her after the first in person encounter. Many other settings can be considered when Alice and Bob do not have public keys and do not have the required time to generate appropriate keys or agree on domain parameters when they meet for the first time.

More formally, we have two small devices, one sender and one receiver, who share no secret information and are in a setting where they can send authentic, but not confidential, information for a short period of time. Later, the sender wants the receiver to recognize the message it sends. The adversarial goal is to make the receiver accept a message as sent by the sender whereas the sender never sent that message. A message recognition protocol (MRP) is secure if the sender or the receiver detect the active adversary. A passive adversary is not considered harmful in this setting.

There have been many recent MRP proposals in the literature, see for example [GMS09, LZWW08, MS08, Mit03]. However, one can go back to 1998 to trace the first protocol [ABC$^+$98] which was designed to fulfill the notion of recognition, although such a term was not used then. Clearly, a digital signature scheme would suffice: Alice gives her public-key to Bob when they first meet and, later, signs her messages. However, in message recognition protocols, one is looking for a cheaper primitive, such as message authentication codes, as opposed to having to compute modular exponentiations. In the case of this paper, the message recognition protocol requires one MAC computation and one pseudorandom function computation. We are unaware of any public-key based solution that could compete with such efficiency.

1.1 Literature Review

There has been considerable recent interest in designing security protocols for devices who do not share a secret key which are placed in an environment that does not provide a public-key infrastructure. One proposal is the use of a narrow-band authenticated channel, along with a broadband insecure channel, in order to achieve *message authentication* in such an environment has been investigated in several recent papers, see for example [GN04, MS09, SA99, Vau05]. In such solutions, the narrow-band channel is available all the time and can be used at least once for every message. In this paper however, we are focusing on a more restricted case where the narrow-band channel is only available once at the beginning at the initialization step. To distinguish between the two, the literature refers to the more restricted case as *recognition*, as opposed to *authentication*.

We now briefly go over the already existing message recognition protocols and mention their advantages and disadvantages compared to one another.

'Guy Fawkes' protocol, designed by Anderson et al. [ABC$^+$98], seems to be the first in line in proposing a protocol achieving recognition, but not authentication. The first variant requires a time-stamping authority and the second variant uses digital signatures for authentication. Hence, the practicality of either variants for restricted devices in restricted environments is under question.

'Remote User Authentication' protocol, proposed by Mitchell [Mit03], uses a message authentication code to authenticate the sender. Hence, does not need a trusted third party. However, it requires computing and sending $2t$ MAC values and sending r secret keys for every message. The suggested parameters are $t \geq 35$ and $r \approx t/2$. Therefore, in terms of computation and communication, it is costly and not suitable for low power and low communication bandwidth devices.

'Zero Common-Knowledge' (ZCK) protocol, by Weimerskirch et al. [WW03], seemed to be the first to admit all the required properties. Implemented by Hammell et al. [HWGW05], the ZCK protocol proved to be practical for devices with restrictive properties such as low computational power, low code space, low communication bandwidth, low energy resources. However, Lucks et al. [LZWW08] found an attack against ZKC which pointed out a flaw in its security proof.

'Jane Doe' protocol, designed by Lucks et al. [LZWW08], uses the idea of using values of a hash chain as keys for MACs to authenticate messages. The Jane Doe protocol exhibits all the preferred properties for small devices placed in a hostile environment. Its security proof is based on the assumption that preimage resistance, second preimage resistance, and their hash chain equivalents, hold for a hash function. Moreover, it makes use of a message authentication code that exhibits existential unforgeability and its hash chain equivalent. The hash chain equivalent properties are non-standard ones. On the other hand, although provably secure, the Jane Doe protocol has a recoverability problem: with one move Eve can bring Alice and Bob out of their synchronized states for the life time of these devices. As a result, they will never be able to communicate again.

Goldberg et al. proposed a self-recoverable MRP [GMS09] to overcome Jane Doe's shortcoming in synchronization. They modified the assumptions of the Jane protocol a little bit, however, they still need to assume the non-standard hash chain assumptions.

Mashatan and Stinson proposed a message recognition protocol [MS08] which does not make use of a hash chain. As a result, they claim that the security assumptions they need become *closer* to the standard notions of preimage resistance and second preimage resistance. However, the assumptions are not exactly standard yet, and it comes with a communication cost of sending about twice as long messages in each flow. Since these protocols are considered in the context of small devices, power consumption is an issue. One should avoid unnecessary communication in order to minimize the power consumption.

Hence, the problem of designing a message recognition protocol based on standard assumptions which exhibits low computational power, low code space, low communication bandwidth, and low energy resources is yet unsolved. This is what we try to achieve in this paper. Alongside of other papers in this area, we do not target any particular device and only specify that low computational power, low code space, low communication bandwidth, and low energy resources are the constraints that our devices are dealing with.

1.2 Our Contribution

We propose an MRP and *for the first time* prove its security in the standard model and based on the existence of pseudorandom functions. The essential idea in our protocol consist in adding random coins in every step of the Jane Doe protocol or its self-recoverable variant due to Goldberg et al. [GMS09].

The MRP presented in this paper is based on the same design principle of the protocols by Lucks et al. [LZWW08] and Goldberg et al. [GMS09] which instructs Alice to send a message m along with a commitment d of m to Bob. Then, Bob is to make Alice recognize him followed by Alice revealing the key in which the commitment was computed with. We use the same design principle, but we use different primitives, e.g., a pseudorandom function. Moreover, the only source of the randomness in the latter two proposals is the root of a hash chain, whereas we insert randomness per key while building the hash chain. Furthermore, using appropriate primitives along with more randomness, we end up *not* requiring the non-standard security assumptions that both Lucks et al. [LZWW08] and Goldberg et al. [GMS09] need to assume. Instead, we prove the security of our MRP based on standard assumptions. Note that the logic of our protocol is similar to the protocol due to Goldberg et al. [GMS09], as opposed to the original Jane Doe protocol, to obtain self-recoverability.

We make use of two primitives, a function $F : \{0,1\}^{s+k} \to \{0,1\}^s$ and a message authentication code MAC : $\{0,1\}^s \times \{0,1\}^* \to \{0,1\}^c$. We define new notions of security for our primitives, F and MAC, namely degree-i preimage resistance and degree-i second preimage resistance for F, and F-degree-i existential unforgeability for MAC. Next, we show that these new notions are equivalent to preimage resistance, second preimage resistance, and existential unforgeability under the assumption that F is a pseudorandom function.

In each protocol instance, Alice and Bob are only required to exchange one F output during their encounter (when they meet in the party). This output can be as short as 80 bits.

As in the Jane Doe protocol, one instance of our MRP provides the message recognition primitive from Alice to Bob. This is not considered as a limitation since one uses two separate protocol instances, one in each direction, to achieve message recognition in both directions. Moreover, the total number of messages to be recognized is required to be preset, both in the Jane Doe protocol and in ours, but we propose the last message to be recognized to simulate a new 'key exchange during a party' which enables Alice and Bob to execute the protocol for another set of messages (see Section 3.2). This is possible whenever message recognition is in place in both directions.

The rest of the paper is organized as follows. In Section 2, we list and analyze the properties we require for the function F and the message authentication code MAC and reduce them to the standard notions. Section 3 is dedicated to our MRP and proves its security based on the assumptions analyzed earlier.

2 Our Security Assumptions and Pseudorandom Functions

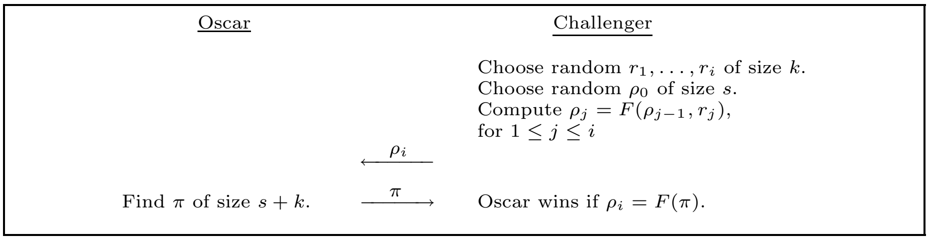
Suppose we have a function $F : \{0,1\}^{s+k} \to \{0,1\}^s$ and a message authentication code MAC : $\{0,1\}^s \times \{0,1\}^* \to \{0,1\}^c$. We now present three non-standard security notions for F and MAC. Later, we show that these notions are equivalent to standard notions based on the assumption that pseudorandom functions exist.

By F *is pseudorandom* we mean that *the F_r family defined by $F_r(\rho) = F(\rho, r)$ is a pseudorandom function family.* In other words, an oracle initialized with a random r and implementing F_r would be indistinguishable from another implementing a random function after a polynomial number of queries. Note that we only require indistinguishability after a single query, and not multiple queries.

Definition 1. *For randomly chosen secrets $r_1, \ldots, r_i$, each having k bits, and randomly chosen secret ρ_0, of size s, let secret $\rho_1, \ldots, \rho_{i-1}$ and known ρ_i be such that $\rho_j = F(\rho_{j-1}, r_j)$, where $1 \le j \le i$. The function F is a* **degree-i preimage resistant (i-PR)** *function if it is infeasible to find π, of size $s + k$, such that $\rho_i = F(\pi)$.*

We note that the notion of degree-i preimage resistance is similar to the well known notion of *one-way on iterates* first introduced by Levin [Lev85]. Variations of this notion was used later by other authors, see for example [GKL93]. We use our variation of 'one-way on iterates' and give it the new name 'degree-i preimage resistance' to be consistent with the later security notions of this paper and also the literature on message recognition protocols.

The notion of degree-i preimage resistance is illustrated in Figure 1 as a game between a player Oscar and a challenger.



Choose random $r_1, \ldots, r_i$ of size k.
Choose random ρ_0 of size s.
Compute $\rho_j = F(\rho_{j-1}, r_j)$,
for $1 \le j \le i$

$\xleftarrow{\rho_i}$

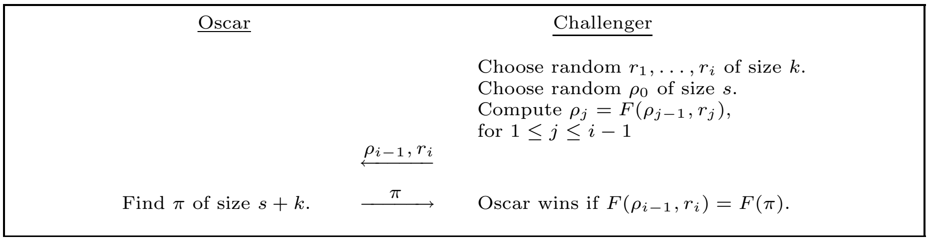
Find π of size $s + k$. $\xrightarrow{\pi}$ Oscar wins if $\rho_i = F(\pi)$.

Fig. 1. Degree-i Preimage Resistant Game

Note that we obtain the classical notion of preimage resistance when $i = 0$. Moreover, for $k = 0$, we obtain the depth-i preimage resistance considered by Lucks et al. [LZWW08]. In other words, we are considering extra randomness for each round, whereas they rely on the randomness of the root element of the hash chain for the entire life time of the protocol.

Definition 2. *For randomly chosen secrets $r_1, \ldots, r_{i-1}$, each having k bits, and randomly chosen secret ρ_0, of size s, let secret $\rho_1, \ldots, \rho_{i-2}$ and known ρ_{i-1} be such that $\rho_j = F(\rho_{j-1}, r_j)$, where $1 \leq j \leq i-1$. The function F is a* **degree-i second preimage resistant (i-SPR)** *function if, given a random r_i of size k, it is infeasible to find π, of size $s+k$, such that $F(\rho_{i-1}, r_i) = F(\pi)$.*

Figure 2 depicts this notion as a game between a player and a challenger.


Oscar

Challenger

Choose random $r_1, \ldots, r_i$ of size k.
Choose random ρ_0 of size s.
Compute $\rho_j = F(\rho_{j-1}, r_j)$,
for $1 \leq j \leq i-1$

$\xleftarrow{\rho_{i-1}, r_i}$

Find π of size $s+k$. $\xrightarrow{\pi}$ Oscar wins if $F(\rho_{i-1}, r_i) = F(\pi)$.

Fig. 2. Degree-i Second Preimage Resistant Game

Again, note that for $i = 1$, we obtain the classical notion of second preimage resistance. Furthermore, if we consider the case of $k = 0$, the case when the only source of randomness is ρ_0, we obtain the depth-i second preimage resistance of Lucks et al. [LZWW08].

Definition 3. *For randomly chosen secrets $r_1, \ldots, r_i$, each having k bits, and randomly chosen secret ρ_0, of size s, let secret $\rho_1, \ldots, \rho_{i-1}$ and known ρ_i be such that $\rho_j = F(\rho_{j-1}, r_j)$, where $1 \leq j \leq i$. A message authentication code MAC is* **F-degree-i existentially unforgeable (i-EU)** *if, knowing ρ_i, it is infeasible to mount an existential forgery against $MAC_{\rho_{i-1}}$ in an adaptive chosen message attack scenario.*

2.1 Pseudorandom Functions Satisfy i-PR

We now show that if F is a pseudorandom function, then the notion of degree-i preimage resistance for F is equivalent to the notion of preimage resistance.

Theorem 1. *Consider a pseudorandom function $F : \{0,1\}^{s+k} \rightarrow \{0,1\}^s$ and let i be polynomial in s and k. Then, the function F is preimage resistant if and only if it is degree-i preimage resistant.*

We actually show a stronger result: if the distribution of $F(\rho, \pi)$, for $(\rho, \pi) \in_R \{0,1\}^{s+k}$, is computationally indistinguishable from the uniform distribution using a single sample, then F is preimage resistant if and only if it is degree-i preimage resistant.

Note that for $k = 0$, as in the case of properties introduced by Lucks et al. [LZWW08], this property can only be true if almost all elements of $\{0,1\}^s$ have a single preimage under F. For a random function F, the probability of every value to have no preimage is roughly e^{-1}. Hence, this property is almost never achieved. And, this argument justifies the introduction of random values r_i.

Proof. Define the success probability of a polynomially bounded player Oscar in the iPR game to be

$$Succ^i_{\mathrm{PR}} := \Pr(F(\pi) = \rho_i),$$

where the probability is taken over all random choices of Oscar and the Challenger. In other words, F is an i-PR if and only if $Succ^i_{\mathrm{PR}}$ is negligible. We are going to first find an upperbound for $|Succ^i_{\mathrm{PR}} - Succ^{i-1}_{\mathrm{PR}}|$ and use triangle inequality to find an upper bound for $|Succ^i_{\mathrm{PR}} - Succ^1_{\mathrm{PR}}|$.

Moreover, for a variable x of size s, define the degree-i distribution to be

$$\mathcal{D}^i_{\mathrm{PR}}(x) := \Pr_{r_0,r_1,\ldots,r_i}[\rho_i = x].$$

Note that $\mathcal{D}^0_{\mathrm{PR}}$ is just the uniform distribution.

Consider a player, Charlie, who wants to distinguish a ρ following either $\mathcal{D}^i_{\mathrm{PR}}$ or $\mathcal{D}^{i-1}_{\mathrm{PR}}$. As illustrated in Figure 3, Charlie can use Oscar as a black-box. His advantage is

$$Adv^i_{\mathrm{PR}} := |Succ^i_{\mathrm{PR}} - Succ^{i-1}_{\mathrm{PR}}|.$$

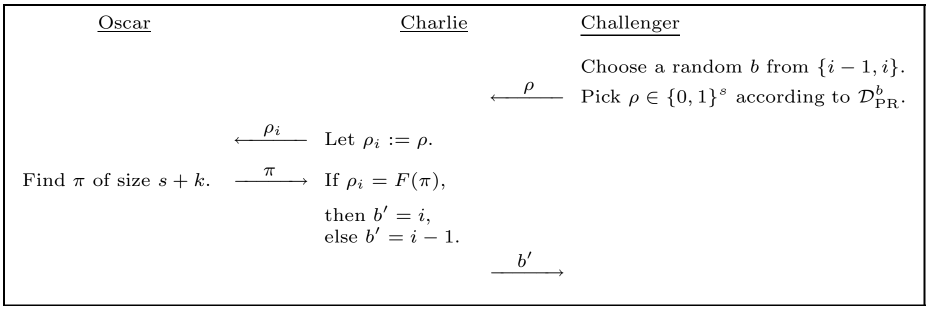

Oscar
Charlie
Challenger
Choose a random b from $\{i-1, i\}$.
Pick $\rho \in \{0,1\}^s$ according to $\mathcal{D}^b_{\mathrm{PR}}$.
ρ
ρ_i
Let $\rho_i := \rho$.
Find π of size $s+k$.
π
If $\rho_i = F(\pi)$,
then $b' = i$,
else $b' = i-1$.
b'

Fig. 3. Degree-i Distinguishing Game

On the other hand, Charlie can be transformed into a distinguisher Dave between $\mathcal{D}^1_{\mathrm{PR}}$ and $\mathcal{D}^0_{\mathrm{PR}}$. To see this, consider a player Dave who, given ρ, uses Charlie as a black-box. This game is illustrated in Figure 4.

The advantage of Dave is equal to Adv^i_{PR}. Therefore, $Adv^i_{\mathrm{PR}} = |Succ^i_{\mathrm{PR}} - Succ^{i-1}_{\mathrm{PR}}|$ must be negligible. Now applying the triangle inequality $i-1$ times, we obtain that $|Succ^i_{\mathrm{PR}} - Succ^1_{\mathrm{PR}}|$ is negligible.

Note that $Succ^1_{\mathrm{PR}}$ is simply the success probability of the adversary in winning the standard notion of preimage resistance for our function F. On the other hand, if the function F is pseudorandom, then the advantage of any polynomial time distinguisher between $\mathcal{D}^1_{\mathrm{PR}}$ and $\mathcal{D}^0_{\mathrm{PR}}$ must be negligible. Therefore, we have shown that F is i-PR if and only if it is PR. $\square$

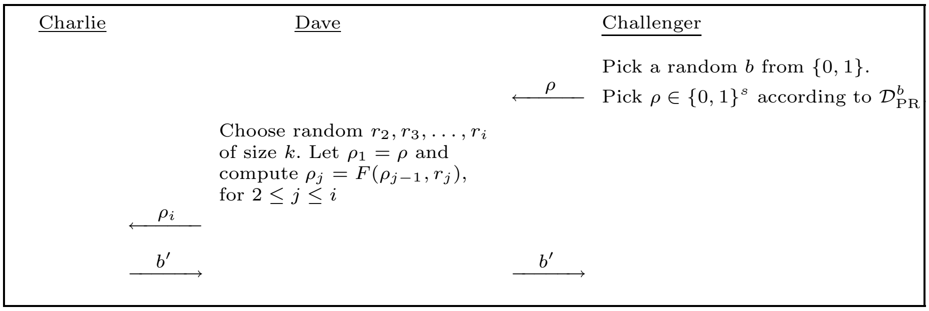

Charlie
Dave
Challenger
Pick a random b from $\{0,1\}$.
Pick $\rho \in \{0,1\}^s$ according to $\mathcal{D}^b_{\mathrm{PR}}$
ρ
Choose random $r_2, r_3, \ldots, r_i$ of size k. Let $\rho_1 = \rho$ and compute $\rho_j = F(\rho_{j-1}, r_j)$, for $2 \leq j \leq i$
ρ_i
b'
b'

Fig. 4. Reducing Degree-1 Distinguishing Game to Degree-i Distinguishing Game

2.2 Pseudorandom Functions Satisfy i-SPR

Similarly to the previous section, we show that the notions of degree-i second preimage resistance and second preimage resistance are equivalent for a pseudorandom function F.

Theorem 2. *Consider a pseudorandom function $F : \{0,1\}^{s+k} \to \{0,1\}^s$ and let i be polynomial in s and k. Then, the function F is second preimage resistant if and only if it is degree-i second preimage resistant.*

Again, we prove a slightly stronger statement than the Theorem. We prove that if the distribution of $F(\rho,\pi)$, for $(\rho,\pi) \in_R \{0,1\}^{s+k}$, is computationally indistinguishable from the uniform distribution using a single sample, then F is second preimage resistant if and only if it is degree-i second preimage resistant.

Proof. We define the success probability of a computationally bounded player Oscar in the iSPR game to be

$$Succ^i_{\mathrm{SPR}} := \Pr(F(\pi) = F(\rho_{i-1}, r_i)),$$

where the probability is taken over all random choices of Oscar and Challenger.

To show that F is iSPR, we need to show that $Succ^i_{\mathrm{SPR}}$ is negligible. We first find an upperbound for $|Succ^i_{\mathrm{SPR}} - Succ^{i-1}_{\mathrm{SPR}}|$ and, then, using the triangle inequality find an upperbound for $|Succ^i_{\mathrm{SPR}} - Succ^1_{\mathrm{SPR}}|$.

Now consider Charlie who wants to distinguish ρ following either $\mathcal{D}^{i-1}_{\mathrm{PR}}$ or $\mathcal{D}^{i-2}_{\mathrm{PR}}$. Figure 5 is depicting Charlie when he is using Oscar as a black-box to distinguish between a random value and ρ_{i-1}. This reduction implies that $|Succ^i_{\mathrm{SPR}} - Succ^{i-1}_{\mathrm{SPR}}|$ is the advantage for distinguishing $\mathcal{D}^{i-1}_{\mathrm{PR}}$ from $\mathcal{D}^{i-2}_{\mathrm{PR}}$.

We conclude like in the proof of Theorem 1. □

2.3 Existential Unforgeable MACs Are i-EU

In this section, given a pseudorandom functions F and a secure message authentication code MAC, we show that MAC is also degree-i existentially unforgeable.

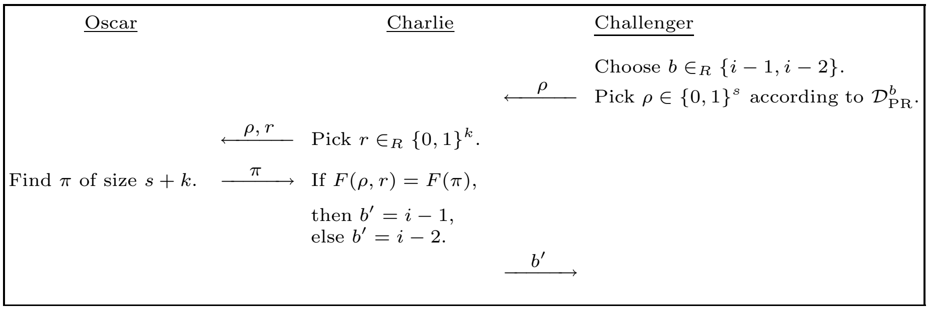

Oscar
Charlie
Challenger
Choose $b \in_R \{i-1, i-2\}$.
Pick $\rho \in \{0,1\}^s$ according to $\mathcal{D}^b_{\mathrm{PR}}$.
ρ
ρ, r
Pick $r \in_R \{0,1\}^k$.
Find π of size $s+k$.
π
If $F(\rho, r) = F(\pi)$,
then $b' = i-1$,
else $b' = i-2$.
b'

Fig. 5. Degree-i Distinguishing Game

Theorem 3. *Consider a message authentication code* MAC : $\{0,1\}^s \times \{0,1\}^* \to \{0,1\}^c$ *and a function* $F : \{0,1\}^{s+k} \to \{0,1\}^s$*, where* $k \geq 2s$*. If* i *is polynomial in* s *and* k*, and* F *is a pseudorandom function, then, the notions of existential unforgeability and* F*-degree-*i *existential unforgeability are equivalent.*

Proof. For a variable x, of size s, we define the following distribution

$$\mathcal{D}^i_\rho(x) = \Pr_r[F(\rho, r) = x].$$

We need to show that for all ρ, $\mathcal{D}^i_\rho$ is indistinguishable from the uniform distribution, using a single sample. This comes from $(F(., r))_{r \in \{0,1\}^k}$ being a pseudorandom function. As in the analysis in the previous proofs, i-EU is equivalent to 1-EU. We now show that 1-EU is equivalent to EU.

Let Oscar be a player who finds standard existential forgeries, and Charlie be a player who is trying to distinguish between $\mathcal{D}^1_\rho$ and the uniform distribution. Now, Charlie uses Oscar as a black box, as illustrated in Figure 6. Note that with $b = 1$, Oscar is a 1-EU adversary. On the other hand, with $b = 0$, Oscar is a regular EU adversary (who is given a useless x). Let $Succ_{b=0}$(Oscar) be the success probability of Oscar when $b = 0$ and $Succ_{b=1}$(Oscar) be his success probability when $b = 1$.

For every ρ, the advantage of Charlie in distinguishing between $\mathcal{D}^1_\rho$ and the uniform distribution is negligible. Hence, on average, the advantage is negligible too. As a result, $|Succ_{b=0}(\text{Oscar}) - Succ_{b=1}(\text{Oscar})|$ is negligible. Thus, we obtain that $Succ_{b=1}$(Oscar) is negligible if and only if $Succ_{b=0}$(Oscar) is negligible, that is if MAC is existentially unforgeable. □

2.4 Separation between PR and i-PR

We show that there is a separation between preimage resistance and degree-i preimage resistance. This implies that considering both assumptions is necessary. Let $\ell : \{0,1\}^s \to \{1, 2, \ldots, s\}$ be defined as $\ell(x)$ = the number of leading zeros of x. Consider a preimage resistant hash function H and define

$$F(x, r) := \mathrm{trunc}_s(0^{\ell(x)+1} 1 \| H(x \| r)),$$

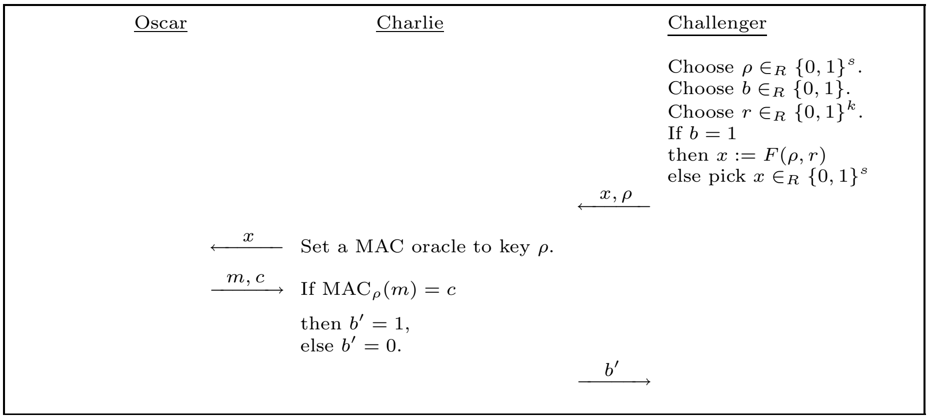

Oscar
Charlie
Challenger
Choose $\rho \in_R \{0,1\}^s$.
Choose $b \in_R \{0,1\}$.
Choose $r \in_R \{0,1\}^k$.
If $b = 1$
then $x := F(\rho, r)$
else pick $x \in_R \{0,1\}^s$
x, ρ
x
Set a MAC oracle to key ρ.
m, c
If $\mathrm{MAC}_\rho(m) = c$
then $b' = 1$,
else $b' = 0$.
b'

Fig. 6. Reducing 1-EU to EU

where trunc_s outputs the first s bits of the input. Since H is preimage resistant, F is also preimage resistant. However, F is not degree-s preimage resistant. One can make similar constructions for degree-i second preimage resistance.

3 A Message Recognition Protocol Based on Pseudorandom Functions

Consider a pseudorandom function $F : \{0,1\}^{s+k} \to \{0,1\}^s$ and a message authentication code $\mathrm{MAC} : \{0,1\}^s \times \{0,1\}^* \to \{0,1\}^c$ with typical parameters $s \geq 80$, $k \geq 2s$, and $c \geq 30$. Moreover, let the maximum number of messages to be authenticated be fixed to be n.

Alice randomly chooses $a_0, a_1, \ldots, a_{n-1}$ of size k and α_0 of size s, and forms a chain of the form $\alpha_i = F(\alpha_{i-1}, a_{i-1})$, $i = 1, \ldots, n$. Analogously, Bob chooses random $b_0, b_1, \ldots, b_{n-1}$ of size k and β_0 of size s, and forms his chain of the form $\beta_i = F(\beta_{i-1}, b_{i-1})$, $i = 1, \ldots, n$.

They start with index n and go downward in the α and β chains, revealing elements of hash chains and the random keys in a descending order. In each session i, Alice and Bob, respectively, use the random a_i and b_i as keys for the MAC values they compute. On the other hand, they use α_i and β_i in session $i + 1$ to commit to a_i and b_i of session i.

The protocol starts with an initialization phase, illustrated in Figure 7, in which Alice and Bob exchange α_n and β_n over an authenticated channel. Eve is passive at this stage, hence, the channel is denoted by $\Rightarrow$.

We first present a high level description of our protocol, depicted in Figure 8 based on the logic of the Jane Doe protocol. Although this high level presentation does not include the details of our proposal, it helps in signifying the differences and it gives a better big picture on how the hash chaining technique is modified in order to obtain a security proof based on standard assumptions.

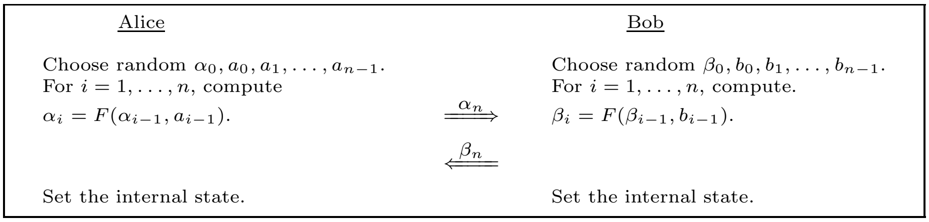

Alice
Bob
Choose random $\alpha_0, a_0, a_1, \ldots, a_{n-1}$.
For $i = 1, \ldots, n$, compute
$\alpha_i = F(\alpha_{i-1}, a_{i-1})$.
$\xRightarrow{\alpha_n}$
Choose random $\beta_0, b_0, b_1, \ldots, b_{n-1}$.
For $i = 1, \ldots, n$, compute.
$\beta_i = F(\beta_{i-1}, b_{i-1})$.
$\xLeftarrow{\beta_n}$
Set the internal state.
Set the internal state.

Fig. 7. Initialization Phase

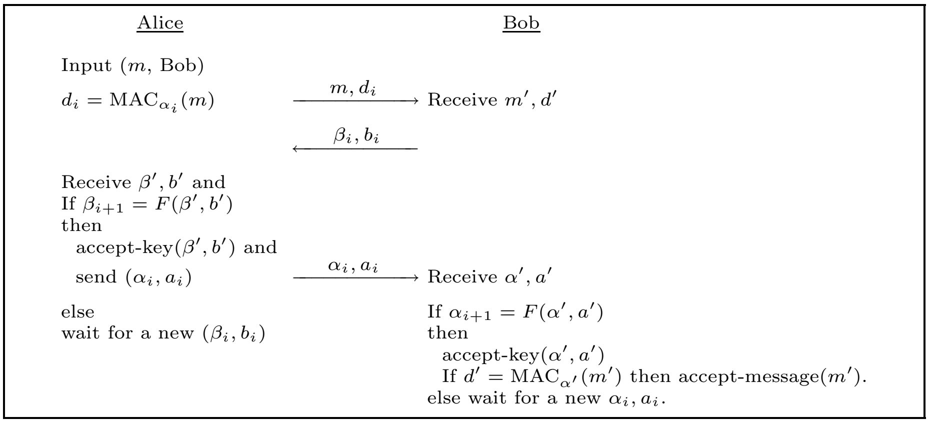

Alice
Bob
Input (m, Bob)
$d_i = \mathrm{MAC}_{\alpha_i}(m)$
$\xrightarrow{m, d_i}$
Receive m', d'
$\xleftarrow{\beta_i, b_i}$
Receive β', b' and
If $\beta_{i+1} = F(\beta', b')$
then
accept-key(β', b') and
send (α_i, a_i)
$\xrightarrow{\alpha_i, a_i}$
Receive α', a'
else
wait for a new (β_i, b_i)
If $\alpha_{i+1} = F(\alpha', a')$
then
accept-key(α', a')
If $d' = \mathrm{MAC}_{\alpha'}(m')$ then accept-message(m').
else wait for a new α_i, a_i.

Fig. 8. High Level Description of our Message Recognition Protocol

Alice uses α_i as the key for the MAC value, but also to make Bob recognize her. Bob uses β_i, so Alice will recognize him. However, b_is are not used to send a message. Note that the role of Alice, as the claimant, and the role of Bob, as the verifier are not reversible. In other words, if Bob wishes to authenticate messages to Alice, they should fix another pair of random keys. Indeed, if the same α_i and β_i are used, a man-in-the-middle attack is possible.

In order to present the details of the logic of our proposed protocol, we adapt the approach of Goldberg et al. [GMS09] to obtain a self-recoverable MRP. Note that our building blocks are different from theirs and, more importantly, our security assumptions are different. The point in adapting the logic of the protocol is to ensure self-recoverability of the protocol.

The internal state of Alice includes (along with each variable's initial value):
- $i_A := n - 1$: the position of Alice in her chain.
- $i_{accA} := n$: the last index of Bob's chain that was accepted by Alice.
- $\beta_A := \beta_n$: the last value of Bob's chain that was accepted by Alice.
- $b_A := Null$: the last value for Bob's randomness accepted by Alice.
- $M := Null$: the input message to be authenticated in the current session.
- a one-bit flag, to distinguish the program states **A0** and **A1**.

Similarly, Bob's internal state is as follows:
- $i_B := n - 1$: the position of Bob in his chain.
- $i_{accB} := n$: the last index of Alice's chain that was accepted by Bob.
- $\alpha_B := \alpha_n$: the last value of Alice's chain that was accepted by Bob.
- $a_B := Null$: the last value for Bob's randomness accepted by Alice.
- $e' := Null$: the MAC value received in the current session, supposedly from Alice.
- $M' := Null$: the message received in the current session, supposedly from Alice.
- a one-trit flag, to distinguish the program states **B0**, **B1**, and **B2**.

We write commit-message(M, i_A) to indicate that Alice is committing herself to sending the message M to Bob in session i_A. We let T be the maximum amount of time Alice waits to receive a response from Bob, and vice versa. Alice and Bob start in program states **A0** and **B0**.

A0 is executed as follows:

```
    If $i_A \leq 0$ then **Abort**.
    Receive input $(M)$.
    Compute $e_{i_A} := \mathrm{MAC}_{\alpha_{i_A}}(i_A \| M)$.
    Send $[e_{i_A}, M]$ to Bob and **goto A1**.
```

B0 is executed as follows:

```
    If $i_B \leq 0$ then **Abort**.
    Wait to receive $[e', M']$, then **goto B1**.
```

B1 has the following description:

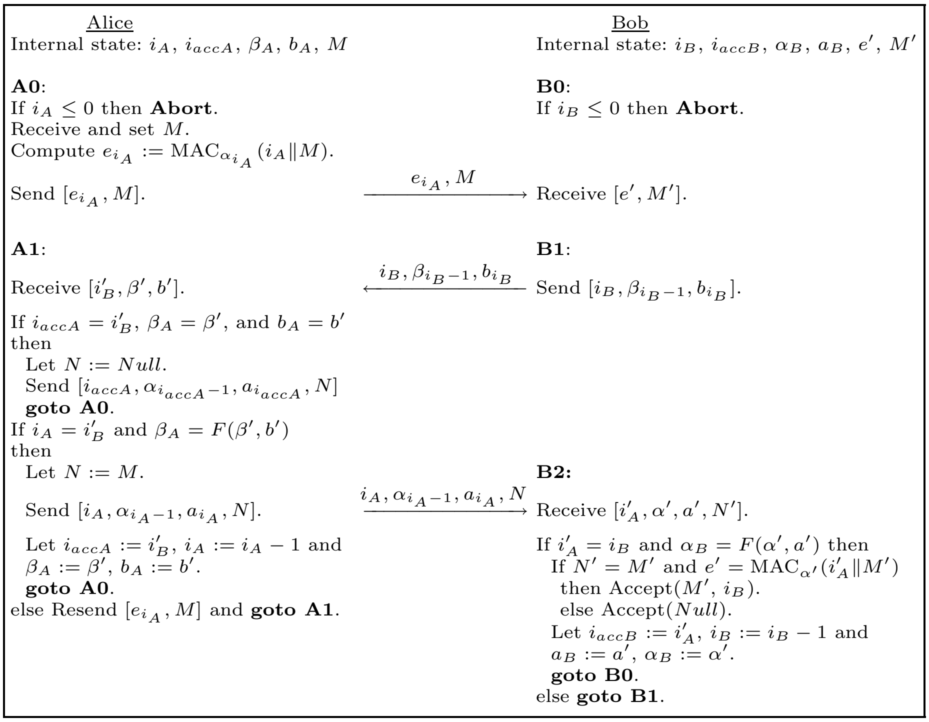
```
    Send $[i_B, \beta_{i_B - 1}, b_{i_B}]$ to Alice and **goto B2**.
```

A1 is performed in the following manner:

```
    Wait at most time $T$ to receive $[i'_B, \beta', b']$.
    If $[i'_B, \beta', b']$ is received, then
        If $i_{accA} = i'_B$, $\beta_A = \beta'$, and $b_A = b'$ (Bob has not received the last flow
        of the previous session) then
            Let $N := Null$.
            Send $[i_{accA}, \alpha_{i_{accA}-1}, a_{i_{accA}}, N]$ and **goto A0**.
        If $i_A = i'_B$ and $\beta_A = F(\beta', b')$ then (Alice and Bob seem to be synchro-
        nized.)
            Let $N := M$.
            Send $[i_A, \alpha_{i_A - 1}, a_{i_A}, N]$ to Bob.
            Let $i_{accA} := i'_B$, $i_A := i_A - 1$, $\beta_A := \beta'$, $b_A := b'$. (Alice updates her
            state.)
            **goto A0**.
        else Resend $[e_{i_A}, M]$ to Bob and **goto A1**.
    If timeout then
    Resend $[e_{i_A}, M]$ to Bob and **goto A1**.
```

B2 is performed as follows:

```
Wait at most time $T$ to receive $[i'_A, \alpha', a', N']$.
If $[i'_A, \alpha', a', N']$ is received, then
    If $i'_A = i_B$ and $\alpha_B = F(\alpha', a')$ then (Alice and Bob seem to be synchronized.)
        If $N' = M'$ and $e' = \mathrm{MAC}_{\alpha'}(i'_A \| M')$ then
            Accept($M'$, $i_B$).
        else Accept($Null$).
        Let $i_{accB} := i'_A$, $i_B := i_B - 1$, $\alpha_B := \alpha'$, $a_B := a'$. (Bob updates his state.)
        **goto B0**.
    else **goto B1**.
If timeout, then **goto B1**.
```



Alice

Internal state: i_A, i_{accA}, β_A, b_A, M

```
**A0**:
If $i_A \leq 0$ then **Abort**.
Receive and set $M$.
Compute $e_{i_A} := \mathrm{MAC}_{\alpha_{i_A}}(i_A \| M)$.
Send $[e_{i_A}, M]$.        $\xrightarrow{e_{i_A}, M}$

**A1**:
Receive $[i'_B, \beta', b']$.        $\xleftarrow{i_B, \beta_{i_B - 1}, b_{i_B}}$
If $i_{accA} = i'_B$, $\beta_A = \beta'$, and $b_A = b'$ then
  Let $N := Null$.
  Send $[i_{accA}, \alpha_{i_{accA}-1}, a_{i_{accA}}, N]$
  **goto A0**.
If $i_A = i'_B$ and $\beta_A = F(\beta', b')$ then
  Let $N := M$.
  Send $[i_A, \alpha_{i_A - 1}, a_{i_A}, N]$.        $\xrightarrow{i_A, \alpha_{i_A - 1}, a_{i_A}, N}$
  Let $i_{accA} := i'_B$, $i_A := i_A - 1$ and $\beta_A := \beta'$, $b_A := b'$.
  **goto A0**.
else Resend $[e_{i_A}, M]$ and **goto A1**.
```

Bob

Internal state: i_B, i_{accB}, α_B, a_B, e', M'

```
**B0**:
If $i_B \leq 0$ then **Abort**.
Receive $[e', M']$.

**B1**:
Send $[i_B, \beta_{i_B - 1}, b_{i_B}]$.

**B2:**
Receive $[i'_A, \alpha', a', N']$.
If $i'_A = i_B$ and $\alpha_B = F(\alpha', a')$ then
  If $N' = M'$ and $e' = \mathrm{MAC}_{\alpha'}(i'_A \| M')$
   then Accept($M'$, $i_B$).
   else Accept($Null$).
  Let $i_{accB} := i'_A$, $i_B := i_B - 1$ and $a_B := a'$, $\alpha_B := \alpha'$.
  **goto B0**.
else **goto B1**.
```

Fig. 9. Proposed Message Recognition Protocol

Figure 9 illustrates the common case of our protocol. It is worth mentioning that since we are basing our protocol on the logic of the message recognition protocol of Goldberg et al. [GMS09], the security analysis, specially the reductions, are similar to those presented in their paper. For example, our protocol directly inherits the self-recoverability property of their protocol. Hence, we do not discuss self-recoverability. On the other hand, we stress that we are using different assumptions and primitives to start with. Hence, we obtain a different protocol and it deserves its own security analysis.

3.1 Security Result

In Appendix A, we prove the following theorem which is analogous to the Security and Self-recoverability Theorem of Goldberg et al. [GMS09].

Theorem 4. *A successful adversary against the protocol of Section 3 who efficiently deceives Bob into accepting (M',i), where M' is not Null and Alice did not send M' in session i, implies an efficient algorithm that finds degree-i preimages or degree-i second preimages, or creates degree-i existential forgeries. Moreover, the adversary cannot stop Alice and Bob from successfully executing the protocol unless she is actively disrupting the communication for the lifetime of Alice and Bob.*

The above theorem together with the theorems of Section 2, namely Theorems 1, 2, and 3, imply the following theorem.

Theorem 5 (Final result). *Consider a pseudorandom function $F : \{0,1\}^{s+k} \to \{0,1\}^s$, $k \geq 2s$, and a message authentication code* MAC $: \{0,1\}^s \times \{0,1\}^* \to \{0,1\}^c$*. Moreover, let i be polynomial in s and k. If F is preimage resistant and second preimage resistant, and if MAC is existentially unforgeable, then there is no efficient adversary against the ith session of the protocol of Section 3.*

The condition on i being a polynomial in s and k is unavoidable to get negligible advantage for the distinguishers in Theorems 1, 2, and 3. On the other hand, note that $i \leq n$ to begin with and, hence, this assumption is reasonable.

3.2 Discussion

Our MRP shares some similarities with protocols based on hash chains (e.g., TESLA [HPJ02], OTP (S/Key) [HMNS98], Lamport authentication [Lam81]). These protocols can adapt to different data structures such as hash trees (see for example, Merkle signatures [Mer89] and Merkle tree traversal [Szy04]) and can be turned into dynamic ones (consult Naor and Yung [NY89]).

Once our MRP is in place in both directions, using two separate pairs of chains, we can use the last two iterations to authenticate some new α_n and β_n to be able to continue. Note that using the same pair of hash chains in both directions results in man-in-the-middle type of attacks.

4 Conclusion

Incorporating random coins in every step of a hash chain, we proposed the first MRP which is provably secure in the standard model and based on standard assumptions. Our protocol uses two primitives, a pseudorandom function $F : \{0,1\}^{s+k} \to \{0,1\}^s$ and an existential unforgeable message authentication code $\mathrm{MAC} : \{0,1\}^s \times \{0,1\}^* \to \{0,1\}^c$.

We defined new notions of security for our primitives, F and MAC, namely degree-i preimage resistance and degree-i second preimage resistance for F, and F-degree-i existential unforgeability for MAC. Then, we showed that these new properties are equivalent to the standard notions of preimage resistance, second preimage resistance, and existential unforgeability under the assumption that F is a pseudorandom function.

References

[ABC+98] Anderson, R., Bergadano, F., Crispo, B., Lee, J.-H., Manifavas, C., Needham, R.: A new family of authentication protocols. In: ACMOSR: ACM Operating Systems Review, vol. 32, pp. 9–20 (1998)

[Geh98] Gehrmann, C.: Multiround unconditionally secure authentication. Designs, Codes, and Cryptography 15(1), 67–86 (1998)

[GKL93] Goldreich, O., Krawczyk, H., Luby, M.: On the existence of pseudorandom generators. SIAM J. Comput. 22(6), 1163–1175 (1993)

[GMS09] Goldberg, I., Mashatan, A., Stinson, D.R.: A new message recognition protocol with self-recoverability for ad hoc pervasive networks. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 219–237. Springer, Heidelberg (2009)

[GN04] Gehrmann, C., Nyberg, K.: Security in personal area networks.In: IEE Security for Mobility, London, pp. 191–230 (2004)

[HMNS98] Haller, N., Metz, C., Nesser, P., Straw, M.: A One-Time Password System. RFC 2289 (February 1998)

[HPJ02] Hu, Y.-C., Perrig, A., Johnson, D.B.: Ariadne: a secure on-demand routing protocol for ad hoc networks. In: Akyildiz, I.F., Lin, J.Y.-B., Jain, R., Bharghavan, V., Campbell, A.T. (eds.) MOBICOM, pp. 12–23. ACM, New York (2002)

[HWGW05] Hammell, J., Weimerskirch, A., Girao, J., Westhoff, D.: Recognition in a low-power environment. In: ICDCSW '05: Proceedings of the Second International Workshop on Wireless Ad Hoc Networking (WWAN), Washington, DC, USA, pp. 933–938. IEEE Computer Society, Los Alamitos (2005)

[Lam81] Lamport, L.: Password authentification with insecure communication. ACM Commun. 24(11), 770–772 (1981)

[Lev85] Levin, L.A.: One-way functions and pseudorandom generators. In: STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing, pp. 363–365. ACM, New York (1985)

[LZWW08] Lucks, S., Zenner, E., Weimerskirch, A., Westhoff, D.: Concrete security for entity recognition: The Jane Doe protocol. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 158–171. Springer, Heidelberg (2008)

[Mer89] Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)

[Mit03] Mitchell, C.J.: Remote user authentication using public information. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 360–369. Springer, Heidelberg (2003)

[MS08] Mashatan, A., Stinson, D.R.: A new message recognition protocol for ad hoc pervasive networks. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 378–394. Springer, Heidelberg (2008)

[MS09] Mashatan, A., Stinson, D.R.: Interactive two-channel message authentication based on interactive-collision resistant hash functions. Int. J. Inf. Secur. 8(1), 49–60 (2009)

[NY89] Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC, pp. 33–43. ACM, New York (1989)

[SA99] Stajano, F., Anderson, R.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Malcolm, J.A., Christianson, B., Crispo, B., Roe, M. (eds.) Security Protocols 1999. LNCS, vol. 1796. Springer, Heidelberg (2000)

[Szy04] Szydlo, M.: Merkle tree traversal in log space and time. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 541–554. Springer, Heidelberg (2004)

[Vau05] Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 309–326. Springer, Heidelberg (2005)

[WW03] Weimerskirch, A., Westhoff, D.: Zero common-knowledge authentication for pervasive networks. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 73–87. Springer, Heidelberg (2004)

A Proof of Theorem 4

We list all possible attacks against our protocol and show that they are all infeasible when the assumptions listed in Section 2 hold. This discussion is similar to what is presented by Goldberg et al. [GMS09].

Eve remains passive for a period of time and, at some point, she gets active to carry out her attack. Since Eve was a passive observer before the attack, it holds that $i_A = i_B$ before she starts her attack. Let $i := i_A = i_B$ and observe that at the start of session i we have $i_{accA} = i_{accB} = i+1$, $a_B = a_{i+1}$, $b_A = b_{i+1}$, $\alpha_B = \alpha_i$, $\beta_A = \beta_i$.

We adapt the approach of Gehrmann [Geh98] in listing all possible attacks by considering different orderings of the flows. Gehrmann labels a flow by **A** when the recipient is Alice. Analogously, a flow is labelled as **B**, if the recipient is Bob. One distinguishes between the attacks that are started in one session and are completed in the same session versus the attacks that are started in one session and completed in a later session, named multi-session attacks.

We first analyze single-session attacks. Recall that Eve has been passive before the session the attack takes place and was only observing the activities. Hence, the attack is started and completed in session i. Gehrmann [Geh98] has showed that there are only $\binom{4}{2} = 6$ different single-session attacks for a three flow protocol. In his notation, these attacks are labelled as AABB, ABAB, BBAA, ABBA, BABA, and BAAB. We will reduce the AABB and BBAA attack scenarios to degree-i preimage resistant or degree-i second preimage resistant games. Moreover, we reduce the ABAB attack scenario to degree-i existential forgeries, degree-i preimage resistant, or degree-i second preimage resistant games.

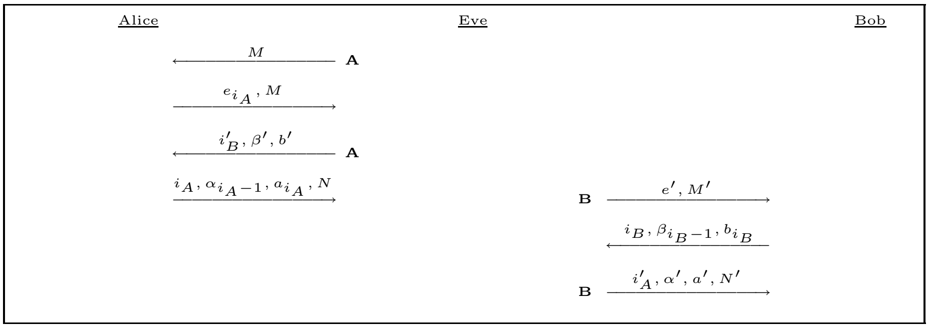


Fig. 10. AABB Attack

AABB Attack. This attack scenario is depicted in Figure 10. Recall that up until the start of this session, Alice and Bob were synchronized having $i_A = i_B = i$. Hence, Eve has to set $i'_A := i$, otherwise Bob will detect Eve, and $i'_B := i$, otherwise Alice will detect Eve. Moreover, recall that $\beta_A = \beta_i$ and $\alpha_B = \alpha_i$. Alice sends $i_A, \alpha_{i_A-1}, a_{i_A}, N$ to Eve if and only if i'_B, β', b' are verified. It implies that unless $\beta_A = F(\beta', b')$, Alice will not cooperate with Eve. Hence, having seen β_i and not having seen β_{i-1}, b_i, Eve has to find β' and b' such that $\beta_i = F(\beta', b')$. That is, Eve has to win the degree-i preimage resistant game of Definition 1.

BBAA Attack. This scenario is shown in Figure 11. For Eve not to be detected by Bob, she has to find i'_A, α', a', and N' such that they get verified by Bob. Note that Eve has $\alpha_B = \alpha_i$ from the previous session. This implies that, not having seen α_{i-1}, a_i, Eve has to find α' and a' such that $\alpha_i = F(\alpha', a')$. Again, if Eve successfully finds such α' and a', then she can win the degree-i preimage resistant game of Definition 1.

ABAB Attack. Figure 12 illustrates this scenario. Eve first receives $\beta_{i_B-1} = \beta_{i-1}$ and $b_{i_B} = b_i$. Then, she has the choice between setting $(\beta', b') = (\beta_{i-1}, b_i)$ or $(\beta', b') \neq (\beta_{i-1}, b_i)$. Let us assume that she sets $(\beta', b') \neq (\beta_{i-1}, b_i)$. In order for Eve not to get detected by Alice, Eve must find β' and b' such that $F(\beta', b') = F(\beta_{i-1}, b_i)$. This implies that she has to win the degree-i second preimage resistant game of Definition 2.

Now, assume that $(\beta', b') = (\beta_{i-1}, b_i)$. Alice will verify β' and b' and, then, send $i_A, \alpha_{i_A-1}, a_{i_A}, N$. Again, Eve has the choice between setting $(\alpha', a') = (\alpha_{i-1}, a_i)$ or $(\alpha', a') \neq (\alpha_{i-1}, a_i)$. In order to set $(\alpha', a') \neq (\alpha_{i-1}, a_i)$ and not get detected by Bob, she has to win the degree-i second preimage resistant game of Definition 2. Let us now assume that she chooses $(\alpha', a') = (\alpha_{i-1}, a_i)$. For Eve not to get detected by Bob, she has to first set $N' := M'$. Moreover, *not knowing*

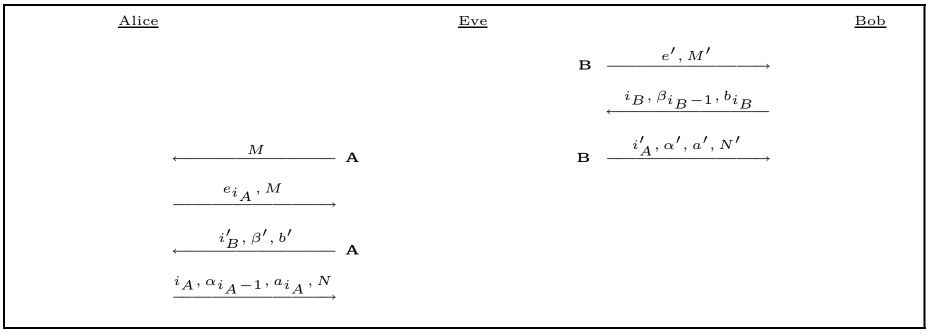


Fig. 11. BBAA Attack

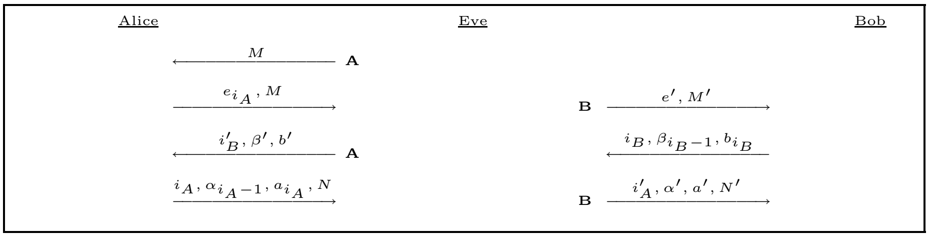


Fig. 12. ABAB Attack

a', Eve must have set $e' := \mathrm{MAC}_{\alpha'}(i'_A \| M')$, for M' to be verified by Bob. Hence, Eve has to perform a degree-i existential forgery, introduced in Definition 3.

It can be shown that the remaining three attack scenarios are reduced to the former three scenarios. In particular, one can reduce the BABA attack to the ABBA attack. Next, the ABBA attack is reduced to the ABAB attack. Last, but not least, one reduces the BAAB attack to the ABAB attack. It remains to take care of multi-session attacks against our protocol. Analogous to the analysis presented by Goldberg et.al [GMS09], one can show that multi-session attacks and show that they reduce to single-session attacks. The reductions for our protocol are analogous to the reductions presented by Goldberg et.al [GMS09], hence, we do not repeat them here.

Affiliation-Hiding Key Exchange with Untrusted Group Authorities

Mark Manulis[1], Bertram Poettering[1], and Gene Tsudik[2]

[1] Cryptographic Protocols Group, TU Darmstadt & CASED, Germany
mark@manulis.eu, bertram.poettering@cased.de
[2] Computer Science Department, UC Irvine, USA
gts@ics.uci.edu

Abstract. Privacy-preserving techniques are increasingly important in our highly computerized society where privacy is both precious and elusive. Affiliation-Hiding Authenticated Key Exchange (AH-AKE) protocols offer an appealing service: authenticated key agreement coupled with privacy of group memberships of protocol participants. This type of service is essential in privacy-conscious p2p systems, mobile ad hoc networks and social networking applications. Prior work has succeeded in constructing a number of secure and efficient AH-AKE protocols which all assume full trust in the Group Authority (GA) — the entity that sets up the group as well as registers and (optionally) revokes members. In this paper, we argue that, for many anticipated application scenarios, the trusted GA model should be relaxed to allow for certain types of malicious behavior. We examine the consequences of malicious GAs and explore the design of stronger AH-AKE protocols that withstand GA attacks. Our results demonstrate that such protocols are both feasible and practical.

1 Introduction

Privacy-Preserving Authentication. Secret Handshakes (SH) [2,8,23,22,21,1,16,17] and Affiliation-Hiding Authenticated Key Exchange (AH-AKE) protocols [14,15] offer privacy-preserving authentication among members of the same group. A user joins a group and obtains its membership credential by registering with the Group Authority (GA). In some schemes, the GA can later revoke these credentials [2,8,23,15,17]. Possession of valid credentials by both participants is a requirement for a successful handshake. Privacy of the authentication process is defined as the requirement to hide the affiliations (group memberships) of protocol participants from outsiders, as well as from each other, unless their respective affiliations match. Since the authentication process is usually followed by a communication session, AH-AKE protocols [14,15] combine secret handshakes with session key establishment. Although session keys are also provided by many secret handshake schemes, the distinguishing feature of AH-AKE protocols is ensuring that session key leakage does not reveal any information about affiliations of session participants. Additionally, AH-AKE protocols guarantee the usual security requirements, i.e., authenticated key exchange security with forward secrecy [7].

Both secret handshakes and AH-AKE protocols can be linkable or unlinkable, depending on whether sessions of the same group member can be related. Linkable secret handshakes and AH-AKE protocols [2, 8, 14, 15] are useful if participants wish to be recognized across different sessions, while keeping their affiliations hidden. This property is typically realized via reusable pseudonyms obtained by members during the registration process. Unlinkable secret handshakes and AH-AKE protocols [23,16,1,17] prevent any correlation among multiple sessions.

Untrusted Group Authorities. Current secret handshakes and AH-AKE protocols assume that group authorities are trusted. This assumption is apparent in the security models of [16,15,17] where corruptions of GAs are not considered. We believe that there are two main reasons for the trusted GA assumption: (a) anticipated applications of secret handshakes were in the domain of homeland security (e.g. intelligence agencies, police) where GA trustworthiness is not questioned, and (b) the GA's role has been considered as being similar to that of a certification authority (CA) in classical PKI-based scenarios where such corruptions are counter-intuitive as they would allow the CA to certify new users at will. However, we consider a more general context of open and commercial applications, such as p2p systems or social networks that allow for the creation of multiple groups, each administrated by its own GA.

In such settings, unconditional trust in the GA is problematic because, unlike a CA only trusted with security, a GA is trusted with both security and privacy. We argue that it is reasonable to trust a GA not to register members frivolously or fraudulently, whereas trusting a GA to maintain and respect privacy of members is a matter that should be treated separately. We now informally discuss the impact of untrusted GAs on security of AH-AKE protocols.

In general, a malicious GA might attempt any of the following: generate public group parameters in a rogue way, create phantom group members, misbehave during the registration process of honest members, and mount active attacks on sessions involving honest members. The resulting challenge is: Which security requirements of AH-AKE protocols can be preserved in a meaningful way? For example, if a malicious GA creates a phantom member and participates on its behalf in an AH-AKE protocol with another party, then this GA would be able to check whether that someone is a member of its group. Moreover, if the protocol succeeds, the GA would also compute the corresponding session key. However, this is an unavoidable consequence of trusting the GA with the security of registration. In fact, it is conceivable that all registration processes are logged and are auditable by some higher authority, thus dis-incentivizing the GA from registering phantom members.

Therefore, we claim that it is meaningful to restrict GA misbehavior to attacks on affiliation-hiding and key secrecy for sessions involving honest members. To this end, our goal is to explore approaches to preserve both of these properties even in the face of a misbehaving GA. In linkable AH-AKE protocols, hiding affiliations of participants appears to be especially challenging due to the use of pseudonyms created during the registration phase. Indeed, none of the linkable AH-AKE protocols we are aware of can attain this goal. There are also some protocols, e.g. [2,8,1], where knowledge of GA secrets would immediately reveal session keys. Additionally, consideration of malicious GAs leads us to a new privacy goal, which we call *untraceability* — the infeasibility for a malicious GA to learn real identities of honest members through their AH-AKE sessions. This requirement is beneficial even for sessions executed with phantom members introduced by the GA. Intuitively, untraceability is related to the GA's ability to obtain information during the registration phase of an honest member that allows it to later link AH-AKE sessions of that member to the registration process, and thus, to the real identity. We observe that many current linkable protocols [2,8,22,15] do not provide this property. This may have serious impact on user privacy. Consider for example an anti-government social network that operates in an oppressive regime. The social network controls the GA and issues credentials to its members. Members can then identify each other and hold discrete meetings and conversations. However, if the government raids the social network and confiscates the GA, then all parameters and records become exposed and the government can thereafter trace and identify all members.

Contributions and Organization. In this paper, we address the challenge posed by untrusted GAs in linkable AH-AKE (LAH-AKE) protocols. This research direction is particularly interesting, since it is counter-intuitive to tolerate linkability of AH-AKE sessions and provide meaningful support for revocation, while offering untraceability against malicious GAs. Our work models untrusted GAs in LAH-AKE protocols and yields stronger LAH-AKE schemes with meaningful security guarantees. Specifically, we extend work by Jarecki, et al. [15], which constructed a LAH-AKE protocol in the trusted GA model.

In Section 2, we propose some updates to the syntax of LAH-AKE protocols and to their security model in order to accommodate corruptions of GAs. More precisely, we extend the adversary model to allow adversarial control of GAs, and define the aforementioned untraceability property. We also update former definitions of authenticated key exchange (AKE) security and linkable affiliation-hiding (LAH) security from [15] to take into account malicious GAs.

In Section 3, we show how the protocol from [15], with security proven under the RSA assumption on safe moduli [13,12] in the random oracle model (ROM) [4], can be fortified to provide new security properties without increasing the round complexity of the original protocol. Our modifications involve several "tricks". First, the initial generation of group parameters is extended with slightly modified (non-interactive) zero-knowledge proofs from [6] to prevent rogue values from being chosen by the GA. Second, we construct an anonymized registration process using the blind RSA signature scheme [9,3] to ensure privacy of obtained pseudonyms and membership credentials. This modification is essential to achieve untraceability. In order to prevent impersonation attacks in AH-AKE sessions we link pseudonyms to public keys of some existentially-unforgeable signature scheme. The idea is to let new members choose their corresponding private signature keys during the registration process and use them during each AH-AKE session to sign key confirmation messages. This allows us to ensure that each member proves ownership of the pseudonym used during protocol execution. By linking pseudonyms to public signature keys we also allow members to use the same pseudonym in different groups. Similar to [15], we can still support revocation of pseudonyms via revocation lists.

The security and efficiency analysis of our protocol is given in Section 4. We stress that security of our scheme no longer relies solely on the RSA problem (since GA knows the factors of the modulus n) but also on the Decisional Diffie-Hellman assumption [5] in subgroups of $\mathbb{Z}_n^*$ of maximal order.

Related Work. Early secret handshake schemes [2,8,22] provided group members with pseudonyms and secret credentials, as part of the registration process. Such schemes are linkable since the same pseudonym is used in multiple handshakes. Unlinkability can be trivially obtained by using one-time pseudonyms; however, this is clearly unscalable. In [2], a credential is a secret element of a bilinear group; [8] uses special CA-oblivious PKI-enabled encryption realized via Schnorr signatures; and [22, 15] use blinded verification of RSA signatures.[1] As the next step, [15] introduced an LAH-AKE protocol that offers both linkable affiliation-hiding and key exchange security with forward secrecy.

Unlinkable secret handshake schemes [1,16,21,17] support reusable credentials while precluding correlation of multiple sessions involving the same participant. However, [1] does not support revocation; [16] requires users to synchronize revocation epochs, and [21] is a complex scheme based on group signatures and broadcast encryption, which complicates revocation. Another flavor of unlinkability (*k-anonymity*) was explored in [23]. Based on some group signature-related techniques, Jarecki and Liu [17] recently proposed another construction with revocation and unlinkable reusable credentials. However, this scheme is hardly practical as it is based on pairings and has linear computation complexity in the number of revoked users.

The only current work on privacy-protection against group authorities is due to Kawai, et al. [18], who deviate from the classical setting and split the role of the group authority among two entities: the issue authority responsible for the registration of users and issue of certificates, and the tracing authority that can trace users from their handshakes. Of particular interest is the new notion of *co-traceability*, which is supposed to prevent authorities from identifying users upon their handshake sessions. The crucial assumption for this property in [18] is that the issue and tracing authorities do not collude. In contrast, our setting is more consistent with prior work, since we treat the group authority as a monolithic entity. Also, our model considers security of session keys and their impact on user privacy, whereas, [18] builds upon [8] where these issues are not modeled. [18] requires group signatures with message recovery and its operation is based on pairings, which significantly decreases the efficiency of the scheme (note that no performance analysis is given in [18]).

[1] Note that the protocol in [22] was shown to be insecure in [15].

2 Untrusted GA Model for Linkable AH-AKE Protocols

We now define and model the security of LAH-AKE protocols while considering malicious GA behavior.

2.1 Linkable Affiliation-Hiding Key Exchange Syntax

An $\mathsf{LAH\text{-}AKE}$ scheme is a four-tuple $\{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$ with components defined as follows:

$\mathsf{CreateGroup}(1^\kappa)$**.** This probabilistic algorithm sets up a new group G and is executed by the corresponding GA. On input of the security parameter 1^κ it generates a public/private group key pair $(G.\mathsf{pk}, G.\mathsf{sk})$, initializes the group's pseudonym revocation list $G.\mathsf{prl}$ to $\emptyset$ and outputs public group parameters $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl})$ and private key $G.\mathsf{sk}$.

$\mathsf{AddUser}(U, G)$**.** This protocol is executed between the prospective group member U and the GA of G. The algorithm on U's side is denoted $\mathsf{AddUserU}(U, G.\mathsf{par})$, the algorithm on GA's side by $\mathsf{AddUserG}(U, G.\mathsf{sk})$. Let π be a session of either the $\mathsf{AddUserU}$ or the $\mathsf{AddUserG}$ algorithm. The *state* of π is defined through the session variable $\pi.\mathsf{state}$ and can take $\mathsf{running}$, $\mathsf{accepted}$, or $\mathsf{rejected}$ values. For both algorithms initially $\pi.\mathsf{state} = \mathsf{running}$. Once $\mathsf{AddUserU}$ session π reaches $\pi.\mathsf{state} = \mathsf{accepted}$ its variable $\pi.\mathsf{result}$ contains a pair $(\mathsf{id}, \mathsf{id.cred})$ where id is a *pseudonym* and $\mathsf{id.cred}$ is a *membership credential* enabling U to authenticate as id in group G in future $\mathsf{Handshake}$ sessions. A user can have several registered pseudonyms in the same group, and the same pseudonym may be registered in different groups.

$\mathsf{Handshake}(\mathsf{params}_1, \mathsf{params}_2)$**.** This is a protocol (handshake) executed between two users U_1 and U_2 on inputs $\mathsf{params}_i = ((\mathsf{id}_i, \mathsf{id}_i.\mathsf{cred}), G_i.\mathsf{par}, r_i)$, $i \in \{1, 2\}$, with $G_i.\mathsf{par} = (G_i.\mathsf{pk}, G_i.\mathsf{prl})$, $r_1 = \mathsf{init}$ and $r_2 = \mathsf{resp}$. We assume that each U_i executes the corresponding interactive algorithm $\mathsf{Handshake}'(\mathsf{param}_i)$. Note that id_i is the pseudonym previously registered to group G_i using the $\mathsf{AddUser}$ algorithm. The protocol verifies that both users are affiliated to the same group (i.e. $G_1 = G_2$) and possess valid membership credentials. If so, the protocol accepts with an established shared session key. Otherwise, it rejects. Users keep track of the state of created $\mathsf{Handshake}$ protocols π through session variables that are initialized as follows: $\pi.\mathsf{state} \leftarrow \mathsf{running}$, $\pi.\mathsf{key} \leftarrow \perp$, $\pi.\mathsf{id} \leftarrow \mathsf{id}$ (where id is the pseudonym used) and $\pi.\mathsf{partner} \leftarrow \perp$. At some point, the protocol will complete and $\pi.\mathsf{state}$ is then updated to either $\mathsf{rejected}$ or $\mathsf{accepted}$. In the latter case $\pi.\mathsf{key}$ is set to the established session key (of length κ) and the pseudonym of the handshake partner is assigned to $\pi.\mathsf{partner}$. The $\mathsf{accepted}$ state cannot be reached if the protocol partner is revoked from the group ($\pi.\mathsf{partner} \in G.\mathsf{prl}$).

$\mathsf{Revoke}(G.\mathsf{sk}, G.\mathsf{prl}, \mathsf{id})$**.** This algorithm is executed by the GA of G and results in the update of G's pseudonym revocation list: $G.\mathsf{prl} \leftarrow G.\mathsf{prl} \cup \{\mathsf{id}\}$.

Definition 1 (Correctness of LAH-AKE). *Assume that two users, U_1 and U_2, register as members of groups G_1 and G_2, and obtain their credentials* $(\mathsf{id}_1, \mathsf{id}_1.\mathsf{cred})$ *and* $(\mathsf{id}_2, \mathsf{id}_2.\mathsf{cred})$*, respectively, through corresponding* AddUser *executions. Assume that U_1 and U_2 participate in a* Handshake *protocol and let π_1 and π_2 denote the corresponding sessions of U_1 and U_2. The LAH-AKE scheme is called* correct *if (a) π_1 and π_2 complete in the same state, which is* accepted *iff $G_1 = G_2$ and $\mathsf{id}_1 \notin G_2.\mathsf{prl}$ and $\mathsf{id}_2 \notin G_1.\mathsf{prl}$ and $r_1 \neq r_2$, and (b) if both sessions accept then* $(\pi_1.\mathsf{key}, \pi_1.\mathsf{partner}, \pi_1.\mathsf{id}) = (\pi_2.\mathsf{key}, \pi_2.\mathsf{id}, \pi_2.\mathsf{partner})$.

2.2 Security Model and Extended Goals

We now present our security model that takes into account malicious GAs. After describing adversarial queries we define three security properties: authenticated key exchange (AKE) security (with forward secrecy), linkable affiliation-hiding (LAH) security, and untraceability. Our model and definitions build upon [15], which considered the first two properties in the trusted GA model.

Adversary Model. The adversary $\mathcal{A}$ is modeled as a PPT machine that interacts with protocol participants via the set of the following basic queries. Unless explicitly noted, we assume that $\mathcal{A}$ always has access to up-to-date exhaustive (system-wide) lists of groups GLi and pseudonyms IDLi (these lists do not disclose the mapping between pseudonyms and groups).

CreateGroup()**.** This query sets up a new group G and publishes its public parameters $G.\mathsf{par}$. The group is added to GLi.

AddUserU$(U, G.\mathsf{par})$**.** This query models the actions of U initiating the AddUser protocol with given target group G. A new protocol session π is started. Optionally, a first protocol message M is output. G is also added to GLi if it is a new group; this allows $\mathcal{A}$ to create its own groups with arbitrary (possibly malicious) public parameters.

AddUserG(G, U)**.** This query differs from AddUserU in that it models GA's actions on the AddUser protocol. We require that G has been already established through the CreateGroup query.

Handshake$(\mathsf{id}, G.\mathsf{par}, r)$**.** This query lets pseudonym id start a new session π of the Handshake protocol. It receives as input the public parameters of the group G wherein the handshake shall take place (given that id has credentials for that group) and a role identifier $r \in \{\mathsf{init}, \mathsf{resp}\}$ that determines whether the session will act as protocol initiator or responder. Optionally, this query returns a first protocol message M.

Send(π, M)**.** Message M is delivered to session π. After processing M, the eventual output is given to $\mathcal{A}$. This query is ignored if π is not waiting for input. Note that π is either an AddUserU, an AddUserG or a Handshake protocol session. If π is an AddUserU session and accepts after processing M then id from $\pi.\mathsf{result}$ is added to IDLi.

Reveal(π)**.** This query is defined only if π is a handshake session. Then, if $\pi.\mathsf{state} \neq \mathsf{running}$ it returns $\pi.\mathsf{state}$ and $\pi.\mathsf{key}$; otherwise the query is ignored.

$\mathsf{Corrupt}(*)$**.** The input is either a pseudonym id or a group identifier G:
  $\mathsf{Corrupt}(\mathsf{id})$: If $\mathsf{id} \in \mathsf{IDLi}$ then, for any group G where id is registered, the corresponding credential $\mathsf{id.cred}$ is given to $\mathcal{A}$.
  $\mathsf{Corrupt}(G)$: For a group G created by $\mathsf{CreateGroup}()$ this query hands G's long term secret $G.\mathsf{sk}$ and control over G's revocation list $G.\mathsf{prl}$ over to $\mathcal{A}$.

$\mathsf{Revoke}(G, \mathsf{id})$**.** This query lets the GA of G include $\mathsf{id} \in \mathsf{IDLi}$ in its pseudonym revocation list $G.\mathsf{prl}$.

Definition 2 (Honest Generation of Pseudonyms and Groups). *A pseudonym* id *is called* honestly generated *if it was established through an* $\mathsf{AddUserU}$ *query. It is called* honest *if thereafter no* $\mathsf{Corrupt}(\mathsf{id})$ *query has been asked. Similarly,* group G *is called* honestly generated *if it was established through a* $\mathsf{CreateGroup}$ *query. It is called* honest *if thereafter no* $\mathsf{Corrupt}(G)$ *query has been asked.*

Definition 3 (Session IDs and Partnered Sessions). Session id $\pi.\mathsf{sid}$ *of a* $\mathsf{Handshake}$ *session* π *that was initiated by pseudonym* id *and is in state* $\mathsf{accepted}$ *is a value that uniquely identifies* π *in the set of all protocol sessions of* id*. Two* $\mathsf{Handshake}$ *sessions* π, π' *are called* partnered *if* $\pi.\mathsf{state} = \pi'.\mathsf{state} = \mathsf{accepted}$ *and* $(\pi.\mathsf{sid}, \pi.\mathsf{id}, \pi.\mathsf{partner}) = (\pi'.\mathsf{sid}, \pi'.\mathsf{partner}, \pi'.\mathsf{id})$.

Authenticated Key Exchange Security. AKE-security of LAH-AKE protocols is determined by analyzing the statistical distribution of resulting session keys: $\mathcal{A}$'s task is to distinguish a key established in a protocol run from a randomly generated value of the same length.

In order to formalize the corresponding indistinguishability game we first introduce two new flags $\pi.\mathsf{revealed}$ and $\pi.\mathsf{tested}$, that are initially set to false, and define the Reveal^* query (as a slightly modified version of Reveal) and the auxiliary Test query with secret parameter $b \in \{0, 1\}$.

$\mathsf{Reveal}^*(\pi)$**.** This query is answered as the regular $\mathsf{Reveal}(\pi)$ query (and $\pi.\mathsf{revealed}$ is set to true) unless $\pi.\mathsf{tested} = \mathsf{true}$ or $\pi'.\mathsf{tested} = \mathsf{true}$ for any session π' that is partnered with π. In the latter case the query is ignored.

$\mathsf{Test}(\pi)$**.** This query is ignored if π is not fresh (see Definition 4). Otherwise $\pi.\mathsf{tested}$ is set to true and a key is returned, obeying the following rule: Let $b \in \{0, 1\}$ denote a bit chosen in advance. In case $b = 1$ $\pi.\mathsf{key}$ is returned. In case $b = 0$ a random element drawn uniformly from $\{0, 1\}^\kappa$ is returned instead. The Test query may be invoked at most once.

The following notion of freshness is useful to exclude trivial attacks and simplify the definition of AKE-security.

Definition 4 (Session Freshness). *A session* π *that is invoked in response to* $\mathsf{Handshake}(\mathsf{id}, G.\mathsf{par}, r)$ *for an honestly generated* id *is called* fresh *if all of the following hold:*

(a) $\pi.\mathsf{state} = \mathsf{accepted}$ *and* $\pi.\mathsf{revealed} = \mathsf{false}$*;*

(b) for any existing session π' *that is partnered with* π*,* $\pi'.\mathsf{revealed} = \mathsf{false}$ *and* $\mathsf{Corrupt}(\pi'.\mathsf{id})$ *was not invoked prior to setting* $\pi'.\mathsf{state} = \mathsf{accepted}$*. Note that in this case* $\pi'.\mathsf{id} = \pi.\mathsf{partner}$*;*

(c) $\pi.\mathsf{partner}$ *was honestly generated and* $\mathsf{Corrupt}(\pi.\mathsf{partner})$ *was not invoked prior to setting* $\pi.\mathsf{state} = \mathsf{accepted}$ *OR* $\pi.\mathsf{partner}$ *was not honestly generated and prior to setting* $\pi.\mathsf{state} = \mathsf{accepted}$ *both* G *was honest and no* $\mathsf{AddUserG}(G, \cdot)$ *has been asked;*

We now provide some rationale: Conditions (a) and (b) prevent $\mathcal{A}$ from revealing the session key computed by π or its partnered session π' and also model forward secrecy by allowing $\mathcal{A}$ to corrupt the corresponding members after the computation of the session key. Condition (c) states meaningful requirements on $\pi.\mathsf{partner}$: on the one hand, it prevents the corruption of honestly generated $\pi.\mathsf{partner}$ during the execution of the handshake (otherwise $\mathcal{A}$ can trivially act in the session on behalf of $\pi.\mathsf{partner}$ and compute the key); on the other hand, it prevents the trivial attack where $\pi.\mathsf{partner}$ was introduced either by the malicious GA of G or as a consequence of the $\mathsf{AddUserG}$ query with which $\mathcal{A}$ could otherwise obtain regular credentials for G. Note that we allow $\mathcal{A}$ to corrupt $\pi.\mathsf{id}$ at any time (even before protocol acceptance) without considering π as not fresh. Essentially this also models resilience against Key Compromise Impersonation (KCI) attacks [19]. We are now ready to formally define AKE-security.

Definition 5 (AKE-Security with Forward Secrecy). *Let* $\mathsf{LAH\text{-}AKE} = \{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$*,* b *be a bit chosen at random, and* $\mathcal{Q} = \{\mathsf{CreateGroup}, \mathsf{AddUserU}, \mathsf{AddUserG}, \mathsf{Handshake}, \mathsf{Send}, \mathsf{Reveal}^*, \mathsf{Test}, \mathsf{Corrupt}, \mathsf{Revoke}\}$ *denote the set of queries available to* $\mathcal{A}$*. By* $\mathsf{Game}^{\mathsf{ake},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ *we denote the following game:*

- $\mathcal{A}^{\mathcal{Q}}(1^\kappa)$ *interacts with all participants using the queries in* $\mathcal{Q}$*;*
- *at some point* $\mathcal{A}^{\mathcal{Q}}$ *asks* $\mathsf{Test}(\pi^*)$ *to a session* π^* *which is* fresh*;*
- $\mathcal{A}^{\mathcal{Q}}$ *continues interacting via queries until it terminates and outputs bit* b'*, which is the output of the game.*

We define:
$$\mathsf{Adv}^{\mathsf{ake}}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa) := \left|2\Pr[\mathsf{Game}^{\mathsf{ake},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa) = b] - 1\right|$$

and denote with $\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{LAH\text{-}AKE}}(\kappa)$ *the maximum advantage over all PPT adversaries* $\mathcal{A}$*. We say that* $\mathsf{LAH\text{-}AKE}$ *is* AKE-secure *if this advantage is negligible.*

In contrast to [15] we do not assume that GAs remain uncorrupted. We do not even restrict $\mathcal{A}$ from setting up a group on its own (presumably by choosing some 'odd' parameters) or from letting honest users register in such groups. Thus, we allow $\mathcal{A}$ to invoke Test query to a session of an honest user that registered with some malicious GA (as long as that session satisfies the freshness conditions).

Linkable Affiliation-Hiding Security. In order to define linkable affiliation-hiding (LAH) security we adopt the simulation-based approach from [15]. The idea is to require that the real protocol execution remains indistinguishable from an idealized one performed by a simulator $\mathcal{SIM}$ that simulates handshake executions without knowing participants' affiliation. This indistinguishability can be defined through a game played between $\mathcal{A}$ and the challenger $\mathcal{C}^b$ initialized with a secret bit $b \in_R \{0,1\}$. $\mathcal{C}^1$ answers all queries of $\mathcal{A}$ honestly following the real protocol specification. $\mathcal{C}^0$ answers the queries of $\mathcal{A}$ with help of $\mathcal{SIM}$ as shown below.

We call a group G *trivially intrudable* if G was not setup honestly through a $\mathsf{CreateGroup}(\,)$ query or if an $\mathsf{AddUserG}(G,\cdot)$ query has been posed by $\mathcal{A}$ or if $\mathcal{A}$ corrupted some pseudonym id' generated in response to some $\mathsf{AddUserU}(\cdot, G.\mathsf{par})$ query. This means that for all trivially intrudable groups $\mathcal{A}$ can obtain valid membership credentials in a trivial way. Therefore, the idea is to let $\mathcal{SIM}$ simulate sessions on behalf of honest pseudonyms only if they belong to groups that are not trivially intrudable.

$\mathsf{CreateGroup}()$, $\mathsf{AddUserU}(U, G.\mathsf{par})$, $\mathsf{Revoke}(G, \mathsf{id})$. These queries are answered honestly without involving $\mathcal{SIM}$.

$\mathsf{AddUserG}(G, U)$,$\mathsf{Corrupt}()$. These queries are answered honestly without involving $\mathcal{SIM}$ unless there exists some still running handshake session π invoked for a group G which is not trivially intrudable. In this latter case $\mathsf{AddUserG}$ and $\mathsf{Corrupt}$ queries are ignored if their input is such that after processing these queries the group G would become trivially intrudable.

$\mathsf{Handshake}(\mathsf{id}, G.\mathsf{par}, r)$. We distinguish between two cases:

<u>CASE 1</u> if G is trivially intrudable then $\mathcal{C}^0$ correctly answers the query. We call the invoked session a CASE 1-session.

<u>CASE 2</u> if G is not trivially intrudable then $\mathcal{C}^0$ invokes $\mathcal{SIM}.\mathsf{Handshake}(\mathsf{id}, r)$ and relays its reply. Note that, in this case, $\mathcal{SIM}$ doesn't learn the group parameters $G.\mathsf{par}$ from this query. We call the invoked session a CASE 2-session.

$\mathsf{Send}(\pi, M)$. If π is an $\mathsf{AddUserU}$ or $\mathsf{AddUserG}$ session then $\mathcal{C}^0$ answers the query itself. If π is a CASE 1-session then $\mathcal{C}^0$ correctly answers the query, whereby, if after processing M session π accepts then $\mathcal{C}^0$ sets the corresponding $\pi.\mathsf{key}$ as follows: If there exists a session π' partnered to π then $\pi.\mathsf{key} \leftarrow \pi'.\mathsf{key}$ is set; Otherwise, if "π is fresh", i.e., if $\pi.\mathsf{partner}$ was honestly generated and $\mathsf{Corrupt}(\pi.\mathsf{partner})$ was not asked OR $\pi.\mathsf{partner}$ was not honestly generated and both G is honest and $\mathsf{AddUserG}(G,\cdot)$ was not asked, then pick $\pi.\mathsf{key} \in_R \{0,1\}^\kappa$; Otherwise, set $\pi.\mathsf{key}$ according to the protocol specification. If π is a CASE 2-session then $\mathcal{C}^0$ invokes $\mathcal{SIM}.\mathsf{Send}(\pi, M)$ and relays its reply.

We let $\mathcal{C}^0$ create sessions π and process corresponding $\mathsf{Send}(\pi,\cdot)$ queries correctly, without involving $\mathcal{SIM}$ (CASE 1-sessions), in all cases for which $\mathcal{A}$ would be able to break the secrecy of $\pi.\mathsf{key}$ in the real protocol execution (during the interaction with $\mathcal{C}^1$) anyway, by the means of some trivial attack. In cases where $\mathcal{A}$ is not supposed to break the secrecy of $\pi.\mathsf{key}$ in the real protocol (the conditions are equivalent to those for session freshness in Definition 4) $\pi.\mathsf{key}$ is chosen randomly.

Before describing how $\mathcal{C}^0$ answers the $\mathsf{Reveal}(\pi)$ queries of $\mathcal{A}$ we define the auxiliary notion of compatible sessions.

Definition 6 (Compatible Sessions). *Two protocol sessions π, π' initiated by* $\mathsf{Handshake}$ *queries are called* compatible *if the groups associated with π, π' are identical, and the concatenation of the messages received by π is a prefix of the concatenation of messages sent by π' and vice versa. If π is a session that received enough messages to complete and there exists a compatible session π' then* $\pi.\mathsf{id}$ *and* $\pi.\mathsf{partner}$ *are set to the pseudonyms of the initiators of π and π'.*

$\mathsf{Reveal}(\pi)$**.** If π is a Case 1-session and $\pi.\mathsf{state} = \mathsf{accepted}$ then $\mathcal{C}^0$ replies with $(\mathsf{accepted}, \pi.\mathsf{key})$, however if $\pi.\mathsf{state} = \mathsf{rejected}$ then $\mathcal{C}^0$ replies with $(\mathsf{rejected}, \perp)$.

If π is a Case 2-session then $\mathcal{C}^0$ first checks whether π received all messages to complete the protocol ($\mathcal{C}^0$ knows this since it observes the messages passed to $\mathcal{SIM}$). If not, $\mathcal{C}^0$ ignores the query. Otherwise, let $\mathsf{Handshake}(\mathsf{id}, G.\mathsf{par}, r)$ be the query which invoked π. $\mathcal{C}^0$ checks whether there exists a session π' which is *compatible* to π. If no such session π' exists then $\mathcal{C}^0$ replies with $(\mathsf{rejected}, \perp)$. Otherwise, $\mathcal{C}^0$ replies with $(\mathsf{accepted}, \pi.\mathsf{key})$ according to the following rules: If $\pi.\mathsf{key}$ is not set but $\pi'.\mathsf{key}$ is then $\pi.\mathsf{key} \leftarrow \pi'.\mathsf{key}$. If both $\pi.\mathsf{key}$ and $\pi'.\mathsf{key}$ are not set then $\pi.\mathsf{key} \in_R \{0,1\}^\kappa$ is chosen randomly.

Compatibility of sessions π and π' means that the corresponding members id and id' satisfy all requirements for the acceptance in the handshake protocol. In this case, it is clear that, by revealing the session key, $\mathcal{A}$ will learn that id and id' belong to the same group. As noticed in [15], this is unavoidable and, even in this case, $\mathcal{A}$ is not supposed to learn the affiliation of these members. Now we are ready to formally define LAH-security.

Definition 7 (LAH-Security). *Let* $\mathsf{LAH\text{-}AKE} = \{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$*, b a randomly chosen bit, and* $\mathcal{Q} = \{\mathsf{CreateGroup}, \mathsf{AddUserU}, \mathsf{AddUserG}, \mathsf{Handshake}, \mathsf{Send}, \mathsf{Corrupt}, \mathsf{Reveal}, \mathsf{Revoke}\}$*. Let* $\mathsf{Game}^{\mathsf{lah},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ *denote the interaction of $\mathcal{A}^{\mathcal{Q}}(1^\kappa)$ with the challenger $\mathcal{C}^b$ via queries until $\mathcal{A}^{\mathcal{Q}}$ outputs bit b' which is the output of the game.*

$$\textit{We define:} \qquad \mathsf{Adv}^{\mathsf{lah}}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa) := \left| 2\Pr[\mathsf{Game}^{\mathsf{lah},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa) = b] - 1 \right|$$

and denote with $\mathsf{Adv}^{\mathsf{lah}}_{\mathsf{LAH\text{-}AKE}}(\kappa)$ *the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that* $\mathsf{LAH\text{-}AKE}$ *is* LAH-secure *if this advantage is negligible.*

Untraceability. The idea behind untraceability is that, even in the presence of a malicious GA, any member remains untraceable throughout its AH-AKE sessions. As discussed in Section 1, this is a new (individual) privacy requirement, distinct from AKE- and LAH-security. We formalize it using the indistinguishability approach: we let $\mathcal{A}$ specify group parameters for a group G and pick two users U_0 and U_1 that are then enrolled into G by the challenger that obtains their respective pseudonyms id_0 and id_1. Untraceability means the inability of $\mathcal{A}$, given id_b for $b \in_R \{0,1\}$, to identify user U_b.

Definition 8 (Untraceability). *Let* $\mathsf{LAH\text{-}AKE} = \{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$*, b a randomly chosen bit, and* $\mathcal{Q} = \{\mathsf{CreateGroup}, \mathsf{AddUserU}, \mathsf{AddUserG}, \mathsf{Handshake}, \mathsf{Send}, \mathsf{Reveal}, \mathsf{Corrupt}, \mathsf{Revoke}\}$ *the set of queries available to $\mathcal{A}$. By* $\mathsf{Game}^{\mathsf{trace},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ *we denote the following interaction of $\mathcal{A}$ with participants, where, for obvious reasons, we prevent $\mathcal{A}$ from accessing the up-to-date pseudonym list* IDLi*:*

- $\mathcal{A}^{\mathcal{Q}}(1^\kappa)$ *interacts with all participants using the queries in $\mathcal{Q}$ and outputs a triple* $(G.\mathsf{par}, U_0, U_1)$ *where* $G.\mathsf{par}$ *are public parameters of a group G and U_0 and U_1 are two distinct users.*
- U_0 *and U_1 are admitted to G through the execution of* $\mathsf{AddUser}(U_0, G)$ *and* $\mathsf{AddUser}(U_1, G)$ *protocols such that the corresponding pseudonyms* id_0 *and* id_1 *are generated. Note that, during this process, the protocol sessions on behalf of G (*$\mathsf{AddUserG}$*) can be executed by $\mathcal{A}$, however, the game does not proceed until the corresponding protocol sessions executed on behalf of U_0 and U_1 (*$\mathsf{AddUserU}$*) accept.*
- $\mathcal{A}$ *is given* id_b *and continues to interact with all participants via queries until it terminates and outputs bit b', which is also the output of the game.*

We define:
$$\mathsf{Adv}^{\mathsf{trace}}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa) := \left|2\Pr[\mathsf{Game}^{\mathsf{trace},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa) = b] - 1\right|$$

and denote by $\mathsf{Adv}^{\mathsf{trace}}_{\mathsf{LAH\text{-}AKE}}(\kappa)$ *the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that* $\mathsf{LAH\text{-}AKE}$ *is* untraceable *if this advantage is negligible (in κ).*

In game $\mathsf{Game}^{\mathsf{trace},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ the corruption of id_b is not forbidden. Therefore, untraceable LAH-AKE schemes hide the real identity of group members even if their membership credentials are leaked.

Revocation and its Relationship to Untraceability and Affiliation-Hiding. Our model raises some concerns about the relationship between the revocation procedure and the untraceability property. We notice that in linkable AH-AKE and SH protocols such as [15,8], where GA learns the pseudonym id of a user U during the registration phase, revocation can be understood in two ways: The first approach is what we call *revocation of users*, i.e., GA may want to revoke some particular user U. The second approach is what we call *revocation of pseudonyms*, i.e., GA may want to revoke some pseudonym id. In the trusted GA model [15], i.e. without untraceability, there is no difference between these two approaches, since GA knows the mapping between U and its id, and can add id in both cases to $G.\mathsf{prl}$. In contrast, our model with untraceability ensures that, during the registration of U, GA does not get any information about the pseudonym id. Therefore, revocation of users is no longer possible. However, users participate in group applications via pseudonyms. Therefore, if some misbehavior is noticed, the responsible pseudonym can be identified and revoked. This type of revocation is still meaningful, since, if GA revokes some pseudonym id that is owned by some user U, then U cannot communicate in that group anymore, unless it obtains a new pseudonym. To do so, U would have to re-register to the same group, which might be forbidden by the admission policy of the GA.

3 LAH-AKE Protocol Secure against Malicious GAs

We now describe our $\mathsf{LAH\text{-}AKE}$ scheme that provides security against malicious GAs. Our construction is based on the scheme from [15]. The modifications apply to the generation of group parameters, the registration process, and the actual key exchange protocol. Revocation of pseudonyms is handled via revocation lists similar to [15].

3.1 Number-Theoretic Assumptions and Building Blocks

Definition 9 (RSA Assumption on Safe Moduli). *Let* $\mathsf{RSA\text{-}G}(\kappa')$ *be a probabilistic algorithm that outputs pairs* (n, e) *where (a)* $n = pq$ *for random* κ'*-bit (safe) primes* $p \neq q$*, (b)* $p = 2p'+1, q = 2q'+1$ *for primes* p', q'*, and (c)* $e \in \mathbb{Z}_{\varphi(n)}$ *is coprime to* $\varphi(n)$*. The RSA-success probability of a PPT solver* $\mathcal{A}$ *is defined as*

$$\mathsf{Succ}^{\mathsf{rsa}}_{\mathcal{A}}(\kappa') := \Pr\big[(n, e) \leftarrow \mathsf{RSA\text{-}G}(\kappa');\ z \leftarrow_R \mathbb{Z}_n^*;\ m \leftarrow \mathcal{A}(n, e, z) \text{ with } m^e = z\big].$$

The RSA assumption on safe moduli *states that the maximum RSA-success probability* $\mathsf{Succ}^{\mathsf{rsa}}(\kappa')$ *(defined over all PPT solvers* $\mathcal{A}$*) is negligible in* κ'*.*

Definition 10 (CDH Assumption in $QR(p)$). *Let* $QR(p)$ *denote the group of quadratic residues modulo a safe prime* $p = 2p'+1$ *of length* κ'*. The CDH-success probability of a PPT adversary* $\mathcal{A}$ *is defined as*

$$\mathsf{Succ}^{\mathsf{cdh}}_{\mathcal{A}}(\kappa') := \max_{\substack{\text{safe prime } p \\ |p|=\kappa', \langle g\rangle = QR(p)}} \Pr\big[x, y \leftarrow_R \mathbb{Z}_{p'}; h \leftarrow \mathcal{A}(p, g, g^x, g^y) \text{ with } h = g^{xy}\big].$$

The CDH assumption in $QR(p)$ *states that the maximum CDH-success probability* $\mathsf{Succ}^{\mathsf{cdh}}(\kappa')$ *(defined over all PPT solvers* $\mathcal{A}$*) is negligible in* κ'*.*

Our scheme uses the following additional building blocks. Let κ, κ' be security parameters. We use cryptographic hash functions modeled as random oracles: $H_1 : \{0,1\}^* \to \{0,1\}^{2\kappa}$ and, for any $n \in \mathbb{N}$ with $|n| = 2\kappa'$, a specific hash function $H_n : \{0,1\}^* \to \mathbb{Z}_n$. Note that H_n can be constructed as $H_n(x) := H(n \,\|\, x) \bmod n$ using some hash function $H : \{0,1\}^* \to \{0,1\}^{2\kappa'+\kappa}$. By $\Sigma := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ we denote a digital signature scheme which is assumed to be existentially unforgeable under chosen message attacks (EUF-CMA).

Camenisch and Michels [6] show how to prove in zero-knowledge (ZK) the correct generation of an RSA modulus $n = pq$ for some safe primes p and q, including the necessary primality tests and without revealing any further information about the factors. We use an extended version of these ZK proofs: In Appendix A we show how to additionally prove in ZK that an element $g \in \mathbb{Z}_n^*$ has maximum order in $\mathbb{Z}_n^*$.

3.2 New LAH-AKE Scheme

We now proceed with the description of algorithms and protocols.

Algorithm $\mathsf{CreateGroup}(1^\kappa)$. This algorithm generates parameters for a new group G as follows: it picks two κ'-bit primes p, q with $p = 2p' + 1$ and $q = 2q' + 1$ for prime numbers p' and q', sets $n = pq$, picks an exponent $e \in \mathbb{Z}_{\varphi(n)}$ which is coprime to $\varphi(n) = (p-1)(q-1) = 4p'q'$, and computes $d = e^{-1} \pmod{\varphi(n)}$. Note that n is a Blum integer, i.e., $p \equiv q \equiv 3 \pmod 4$.

As $\mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ the largest element order in $\mathbb{Z}_n^*$ is $\mathrm{lcm}(\varphi(p), \varphi(q)) = 2p'q' = \varphi(n)/2$, and hence $\mathbb{Z}_n^*$ is not cyclic. For elements $g \in \mathbb{Z}_n^*$ with $\mathrm{ord}(g) \geq p'q'$ and $g^{p'q'} \neq \pm 1$ it follows that $\mathrm{ord}(g) = 2p'q'$ and that $-1 \notin \langle g \rangle$. In this case, we have $\mathbb{Z}_n^* \cong \langle -1 \rangle \times \langle g \rangle$. Let the $\mathsf{CreateGroup}$ algorithm pick such an element g. Since about a half of the elements in $\mathbb{Z}_n^*$ has the desired properties [15], g can easily be found by just random sampling and testing.

Our security model treats GAs as untrusted parties. This even includes lack of trust in the honest generation of group parameters. Appendix A sketches a technique based on [6] that constructs a ZK proof for (n, g) showing that n is a safe RSA modulus and that $\mathbb{Z}_n^* = \langle -1 \rangle \times \langle g \rangle$. By $\Pi_{n,g}$ we denote its non-interactive version that can be obtained via the classical Fiat-Shamir transformation [11].

Finally, the algorithm sets $G.\mathsf{prl} = \emptyset$ and outputs $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl}, \Pi_{n,g})$ with $G.\mathsf{pk} = (n, e, g)$ and the private key $G.\mathsf{sk} = d$.

Protocol $\mathsf{AddUser}(U, G)$. Member admission is implemented using a protocol between U and GA, as specified in Figure 1. Communication between U and GA is assumed to be authentic, yet it does not need to be confidential as in [15], mainly because membership credentials in our registration process are not transported from GA to U but computed through interaction, which need not be private. Some details follow. It is assumed that U obtains public group parameters $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl}, \Pi_{n,g})$ prior to the protocol execution. Then, in a first step, U examines the validity of the group parameters (n, e, g) by checking the NIZK proof $\Pi_{n,g}$. Then, U generates a signature key pair $(pk, sk) \leftarrow \Sigma.\mathsf{KGen}(\kappa)$. The verification key pk is hereafter used by U as its pseudonym id in group G, i.e., we set $\mathsf{id} := pk$. Using the standard blind RSA signature scheme [9,3] U obtains an RSA signature $\sigma_{\mathsf{id}} = H_n(\mathsf{id})^d$ on $H_n(\mathsf{id})$, as depicted in Figure 1 (all computations are mod n). Note that the blinding factor r^e effectively hides id and $H_n(\mathsf{id})$ from G. The output of U is $(\mathsf{id}, \mathsf{id.cred})$ with $\mathsf{id.cred} = (\sigma_{\mathsf{id}}, sk)$.

Protocol $\mathsf{Handshake}((\mathsf{id}_A, \mathsf{id}_A.\mathsf{cred}, G_A, \mathsf{init}), (\mathsf{id}_B, \mathsf{id}_B.\mathsf{cred}, G_B, \mathsf{resp}))$
The handshake protocol is executed between two users, say A and B, holding pseudonyms id_A and id_B, public group keys $G_A.\mathsf{pk} = (n_A, e_A, g_A)$ and $G_B.\mathsf{pk} = (n_B, e_B, g_B)$, and credentials $\mathsf{id}_A.\mathsf{cred} = (\sigma_{\mathsf{id}_A}, sk_A)$ and $\mathsf{id}_B.\mathsf{cred} = (\sigma_{\mathsf{id}_B}, sk_B)$, respectively. The protocol is specified in Figure 2. We assume that all computations are performed mod n_A on the left and mod n_B on the right, except the assignments containing the padding function pad (line 5). The aim of

	User U		Authority G
1	VALIDATE (n, e, g)		
2	$(\mathsf{id}, sk) \leftarrow \Sigma.\mathsf{KGen}(\kappa)$		
3	$r \leftarrow_R \mathbb{Z}_n^*$		
4	$m_1 \leftarrow H_n(\mathsf{id})r^e$	$\xrightarrow{m_1}$	$m_2 \leftarrow (m_1)^d$
5			$\left[= H_n(\mathsf{id})^d r\right]$
6	$\sigma_{\mathsf{id}} \leftarrow m_2/r \left[= H_n(\mathsf{id})^d\right]$	$\xleftarrow{m_2}$	
7	ABORT IF $(\sigma_{\mathsf{id}})^e \neq H_n(\mathsf{id})$		
8	$\mathsf{id.cred} \leftarrow (\sigma_{\mathsf{id}}, sk)$		
9	OUTPUT $(\mathsf{id}, \mathsf{id.cred})$		

Fig. 1. Specification of $\mathsf{AddUser}(U, G)$

the latter is to hide the moduli n_A and n_B from observers. The technique dates back to [10]. A leakage could allow $\mathcal{A}$ to make conclusions about the players' affiliations. The padding function pad is a probabilistic mapping that transforms $\theta' \in \mathbb{Z}_n$ into an integer θ in the interval $[0, 2^{2\kappa'+\kappa} - 1]$ by choosing a random $k \leftarrow_R [0, \lfloor 2^{2\kappa'+\kappa}/n \rfloor - 1]$ and returning $\theta \leftarrow \theta' + kn$. Note that $\theta \equiv \theta' \bmod n$ and that $\left\{(-1)^b g^t \mid (b,t) \in \{0,1\} \times \mathbb{Z}_{n/2}\right\} = \mathbb{Z}_n$.

	User A (Initiator)		User B (Responder)
1	LET $(n_A, e_A, g_A) = G_A.\mathsf{pk}$		LET $(n_B, e_B, g_B) = G_B.\mathsf{pk}$
2	LET $(\sigma_{\mathsf{id}_A}, sk_A) = \mathsf{id}_A.\mathsf{cred}$		LET $(\sigma_{\mathsf{id}_B}, sk_B) = \mathsf{id}_B.\mathsf{cred}$
3	$(b_A, t_A) \leftarrow_R \mathbb{Z}_2 \times \mathbb{Z}_{n_A/2}$		$(b_B, t_B) \leftarrow_R \mathbb{Z}_2 \times \mathbb{Z}_{n_B/2}$
4	$\theta'_A \leftarrow (-1)^{b_A}(g_A)^{t_A}\sigma_{\mathsf{id}_A}$		$\theta'_B \leftarrow (-1)^{b_B}(g_B)^{t_B}\sigma_{\mathsf{id}_B}$
5	$\theta_A \leftarrow \mathsf{pad}(\theta'_A, n_A, \kappa, \kappa')$	$\xrightarrow{m_1 = (\mathsf{id}_A, \theta_A)}$	$\theta_B \leftarrow \mathsf{pad}(\theta'_B, n_B, \kappa, \kappa')$
6		$\xleftarrow{m_2 = (\mathsf{id}_B, \theta_B)}$	
7	$\mathsf{sid}_A \leftarrow m_1 \Vert m_2$		$\mathsf{sid}_B \leftarrow m_1 \Vert m_2$
8	$r_A \leftarrow \left((\theta_B)^{e_A} H_{n_A}(\mathsf{id}_B)^{-1}\right)^{2t_A}$		$r_B \leftarrow \left((\theta_A)^{e_B} H_{n_B}(\mathsf{id}_A)^{-1}\right)^{2t_B}$
9	$(K_A, c_A) \leftarrow H_1(r_A \Vert \mathsf{sid}_A)$		$(K_B, c_B) \leftarrow H_1(r_B \Vert \mathsf{sid}_B)$
10	$s_A \leftarrow \Sigma.\mathsf{Sign}(sk_A, c_A \Vert \mathsf{sid}_A \Vert \mathsf{init})$	$\xrightarrow{s_A}$	$s_B \leftarrow \Sigma.\mathsf{Sign}(sk_B, c_B \Vert \mathsf{sid}_B \Vert \mathsf{resp})$
11		$\xleftarrow{s_B}$	
12	$v_A \leftarrow \Sigma.\mathsf{Verify}(\mathsf{id}_B, s_B, c_A \Vert \mathsf{sid}_A \Vert \mathsf{resp})$		$v_B \leftarrow \Sigma.\mathsf{Verify}(\mathsf{id}_A, s_A, c_B \Vert \mathsf{sid}_B \Vert \mathsf{init})$
13	IF $\mathsf{id}_B \notin G_A.\mathsf{prl}$ AND $v_A = \mathsf{true}$ THEN		IF $\mathsf{id}_A \notin G_B.\mathsf{prl}$ AND $v_B = \mathsf{true}$ THEN
14	$(\mathsf{key}, \mathsf{partner}, \mathsf{state}) \leftarrow (K_A, \mathsf{id}_B, \mathsf{accepted})$		$(\mathsf{key}, \mathsf{partner}, \mathsf{state}) \leftarrow (K_B, \mathsf{id}_A, \mathsf{accepted})$
15	ELSE $(\mathsf{key}, \mathsf{partner}, \mathsf{state}) \leftarrow (\perp, \perp, \mathsf{rejected})$		ELSE $(\mathsf{key}, \mathsf{partner}, \mathsf{state}) \leftarrow (\perp, \perp, \mathsf{rejected})$

Fig. 2. Specification of $\mathsf{Handshake}((\mathsf{id}_A, \mathsf{id}_A.\mathsf{cred}, G_A, \mathsf{init}), (\mathsf{id}_B, \mathsf{id}_B.\mathsf{cred}, G_B, \mathsf{resp}))$

Remark 1. There is an important difference between our Handshake protocol and that from [15]: The key confirmation message sent in the second round of our protocol is a signature; its validity confirms not only the equality of the session key but also serves as a proof for the ownership of claimed id by the participant (i.e. through the knowledge of the corresponding secret key). This way we thwart active impersonation attacks where $\mathcal{A}$ exploits the blinding feature of the AddUser protocol and obtains credentials for ids of honest users. Note that the possibility of such pseudonym impersonation by insiders (group members) would violate AKE security as defined in Section 2.2 (see part (c) of Definition 4). In contrast, the handshake protocol in [15] computes key confirmation messages as hash values computed using the established session key.

Algorithm $\mathsf{Revoke}(G.\mathsf{sk}, G.\mathsf{prl}, \mathsf{id})$. Revocation of pseudonyms is handled by the GA of G by placing the pseudonym into the corresponding pseudonym revocation list $G.\mathsf{prl}$. We assume that this list is distributed using authenticated channels.

4 Security and Efficiency

Correctness of the described LAH-AKE protocol follows by inspection. Note that the intermediate values r_A and r_B have the form

$$\begin{aligned} r_A &= \left((\theta'_B)^{2e_A} H_n(\mathsf{id}_B)^{-2}\right)^{t_A} = \left((g_B)^{2e_A t_B}(\sigma_{\mathsf{id}_B})^{2e_A} H_n(\mathsf{id}_B)^{-2}\right)^{t_A} \\ &= \left((g_B)^{2e_A t_B} H_n(\mathsf{id}_B)^{2e_A d_A} H_n(\mathsf{id}_B)^{-2}\right)^{t_A} = (g_B)^{2e_A t_A t_B} \qquad (\mathrm{mod}\, n_A) \end{aligned}$$

and, analogously, $r_B = (g_A)^{2e_B t_B t_A}$. Presuming that $(n_A, e_A, g_A) = (n_B, e_B, g_B)$, i.e. that users A and B are members of the same group, r_A and r_B evaluate to the same value, and $K_A = K_B$ follows.

Remark 2. Protocol correctness requires that the $H_n(\mathsf{id})$ values are indeed invertible $\mathrm{mod}\, n$. In fact, this is not the case for $H_n(\mathsf{id}) \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$ and then the protocol will fail. However, this occurs with negligible probability. We stress that this remark also applies to the original protocol in [15].

We now state that our LAH-AKE construction satisfies the AKE- and LAH-security and untraceability goals defined in Section 2.2. The corresponding proofs[2] (with estimated attack probabilities) are provided in the full version of this paper.

Theorem 1 (AKE-Security). *Our* LAH-AKE *scheme is AKE-secure (Def. 5) in the random oracle model under the RSA (Def. 9) and CDH (Def. 10) assumptions if $\Pi_{n,g}$ is sound and zero-knowledge, and Σ is EUF-CMA secure.*

Theorem 2 (LAH-Security). *Our* LAH-AKE *scheme is LAH-secure (Def. 7) in the random oracle model under the RSA (Def. 9) and CDH (Def. 10) assumptions if $\Pi_{n,g}$ is sound and zero-knowledge, and Σ is EUF-CMA secure.*

Theorem 3 (Untraceability). *Our* LAH-AKE *scheme is untraceable (Def. 8) in the random oracle model if $\Pi_{n,g}$ is sound.*

[2] It might initially seem, that, due to the utilization of blind signatures in the AddUser protocol, security of LAH-AKE cannot be shown without relying on the hardness of some One-More RSA-Inversion problem [3]. However, careful examination of the constraints for session freshness in Definition 4 shows that the AddUserG query (i.e. the adversary's access to the blind signature oracle) is available only in cases where the corresponding GA may be corrupted anyway. Hence, the RSA assumption suffices to prove the protocol's security.

Efficiency. The cost of our handshake protocol is dominated by the computations of $\theta'_{A/B}$, $r_{A/B}$, generation of $s_{A/B}$ and verification of $s_{B/A}$. The first two involve exponentiations (of size $\log n = 2\kappa'$) and the cost of the last two depends on the balance between $\Sigma.\mathsf{Sign}$ and $\Sigma.\mathsf{Verify}$. Many current signature schemes involve either low verification and high generation costs (e.g, RSA) or vice versa (e.g., DSA). In any case, suffice it to say that, for each participant, the overall computation cost amounts to approximately 3 full-blown exponentiations. Considering the high degree of security offered by our scheme, the overhead is very low.

The NIZK proof $\Pi_{n,g}$ in the AddUser protocol is the most expensive operation. In fact, the verifier would have to compute about $24\kappa t \log n$ (multi-) exponentiations, where 2^{-t} is the error-probability for the primality tests (see [6], Sections 4.3 and 5.1). Note that [6] suggests two optimizations on the protocol: the first one in [6, Section 5.2] that effectively removes factor t from the above equation; and the second one [6, Section 2.2] that is applicable only to interactive ZK proofs and eliminates factor κ. Nevertheless, the complexity for verifying the correctness of the group parameters remains relatively high. However, this proof is necessary (in theory) for the security in our model. In *practice*, it is conceivable to completely omit the verification of $\Pi_{n,g}$, since the set of public parameters of a group is fixed once, upon the initialization. Therefore, its verification by a single trusted auditing authority would suffice. An appropriate (weaker) security model is easily derived from that given in Section 2.2 by modifying the AddUserU query such that only group parameters $G.\mathsf{par}$ are accepted that were established by a CreateGroup query before. Note that, in this relaxed model, the untraceability of our LAH-AKE scheme becomes *unconditional* (since there is no longer any need to assume soundness of $\Pi_{n,g}$ in Theorem 3).

5 Conclusion

AH-AKE protocols are powerful privacy-preserving authentication mechanisms usable in various collaborative and group-based applications, such as p2p systems, mobile ad-hoc groups, and social networks. Prior protocols require a high degree of trust in the GA. In this work, we considered untrusted GAs in linkable AH-AKE protocols. We developed a model containing meaningful definitions of security and designed a concrete protocol for mitigating GA misbehavior, guaranteeing the valuable privacy goals for the users.

Acknowledgement

The authors wish to thank Marc Joye for the discussion about the hardness of the CDH assumption in the safe RSA setting, and Andreas Peter for general discussions on number-theoretic assumptions used in this paper and relationships among them.

References

1. Ateniese, G., Kirsch, J., Blanton, M.: Secret Handshakes with Dynamic and Fuzzy Matching. In: Network and Distributed System Security Symposium, NDSS 2007 (2007)
2. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.-C.: Secret Handshakes from Pairing-Based Key Agreements. In: IEEE Symposium on Security and Privacy 2003, pp. 180–196. IEEE CS, Los Alamitos (2003)
3. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 309–328. Springer, Heidelberg (2002)
4. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: 1st ACM Conference on Computer and Communications Security (CCS 1993), pp. 62–73. ACM, New York (1993)
5. Boneh, D.: The Decision Diffie-Hellman Problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
6. Camenisch, J., Michels, M.: Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
7. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
8. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret Handshakes from CA-Oblivious Encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
9. Chaum, D.: Blind Signatures for Untraceable Payments. In: CRYPTO 1982, pp. 199–203. Plenum Press, New York (1983)
10. Desmedt, Y.: Securing Traceability of Ciphertexts — Towards a Secure Software Key Escrow System (Extended Abstract). In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 147–157. Springer, Heidelberg (1995)
11. Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
12. Gennaro, R., Rabin, T., Jarecki, S., Krawczyk, H.: Robust and Efficient Sharing of RSA Functions. J. Cryptology 20(3), 393 (2007)
13. Gennaro, R., Rabin, T., Krawczyk, H.: RSA-Based Undeniable Signatures. J. Cryptology 20(3), 394 (2007)
14. Jarecki, S., Kim, J., Tsudik, G.: Group Secret Handshakes or Affiliation-Hiding Authenticated Group Key Agreement. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 287–308. Springer, Heidelberg (2006)
15. Jarecki, S., Kim, J., Tsudik, G.: Beyond Secret Handshakes: Affiliation-Hiding Authenticated Key Exchange. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 352–369. Springer, Heidelberg (2008)
16. Jarecki, S., Liu, X.: Unlinkable Secret Handshakes and Key-Private Group Key Management Schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 270–287. Springer, Heidelberg (2007)
17. Jarecki, S., Liu, X.: Private Mutual Authentication and Conditional Oblivious Transfer. In: Halevi, S. (ed.) Advances in Cryptology — CRYPTO 2009. LNCS, vol. 5677, pp. 90–107. Springer, Heidelberg (2009)

18. Kawai, Y., Yoneyama, K., Ohta, K.: Secret Handshake: Strong Anonymity Definition and Construction. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 219–229. Springer, Heidelberg (2009)
19. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
20. 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)
21. Tsudik, G., Xu, S.: A Flexible Framework for Secret Handshakes. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 295–315. Springer, Heidelberg (2006)
22. Vergnaud, D.: RSA-Based Secret Handshakes. In: Ytrehus, Ø. (ed.) WCC 2005. LNCS, vol. 3969, pp. 252–274. Springer, Heidelberg (2006)
23. Xu, S., Yung, M.: k-Anonymous Secret Handshakes with Reusable Credentials. In: 11th ACM Conference on Computer and Communications Security (CCS 2004), pp. 158–167. ACM, New York (2004)

A About the Construction of $\Pi_{n,g}$

Camenisch and Michels [6] describe a cryptographic protocol where a prover P holding the factorization $n = pq$ of a public safe RSA modulus n proves in Zero-Knowledge to a verifier V that n is indeed a safe RSA modulus. This is done by sending randomized Pedersen commitments [20] $C(p), C(q), \ldots$ of p, q and some intermediate variables to V, and by proving in ZK arithmetical relationships between them. Within others, probabilistic primality tests for p and q are run and ZK-verified step by step by V. In the full version of this paper we describe the deployed techniques in more detail.

The protocol given in [6] is not directly applicable to the CreateGroup/AddUser algorithms of our AH-AKE scheme (see Section 3.2) as not only the fact that n is a safe RSA modulus has to be proven, but in addition also the equality $\mathbb{Z}_n^* = \langle -1 \rangle \times \langle g \rangle$. A necessary condition for the latter is that $\operatorname{ord}(g) = \varphi(n)/2 = 2p'q'$ and $g^{p'q'} \neq \pm 1$. As n is a Blum integer there are exactly four elements $a \in \mathbb{Z}_n^*$ with $a^2 = 1$, namely ± 1 and $\pm\omega$ for some $\omega \in \mathbb{Z}_n^*$. It is hence sufficient if P proves to V that $g^{p'q'} = \pm\omega$. In the following, we sketch how this can be done.

The Euclidean algorithm finds integers x, y satisfying $px + qy = 1$. For $\omega = px - qy \pmod{n}$ the Chinese Remainder Theorem shows that $\omega^2 = 1 \pmod{n}$. In a first step, P publishes commitments $C(x), C(y)$ for x and y, respectively, and proves (in ZK) $C(1) = C(p) \cdot C(x) + C(q) \cdot C(y)$. It then computes ω, publishes $C(\omega)$ and proves $C(\omega) = C(p) \cdot C(x) - C(q) \cdot C(y)$. After computing $g' = g^{p'q'}$, it publishes $C(g')$ and proves $C(g') = C(g)^{C(p') \cdot C(q')}$, concluding the proof by revealing either $C(g') = C(\omega)$ or $C(g') = C(-\omega)$.

Privacy-Preserving Group Discovery with Linear Complexity

Mark Manulis[1], Benny Pinkas[2,⋆], and Bertram Poettering[1]

[1] Cryptographic Protocols Group, TU Darmstadt & CASED, Germany
mark@manulis.eu, bertram.poettering@cased.de
[2] Department of Computer Science, University of Haifa, Israel
benny@pinkas.net

Abstract. Affiliation-Hiding Authenticated Key Exchange (AH-AKE) protocols enable two distrusting users, being in possession of membership credentials for some group, to establish a secure session key without leaking any information about this group to non-members. In practice, users might be members of several groups, and such protocols must be able to generate session keys between users who have one or more groups in common. Finding efficient solutions for this *group discovery* problem has been considered an open research problem, inherent to the practical deployment of these protocols.

We show how to solve the privacy-preserving group discovery problem with *linear* computational and communication complexity, namely $O(n)$ complexity where n is the number of groups per user. Our generic solution is based on a new primitive — *Index-Hiding Message Encoding (IHME)*, for which we provide definitions and an unconditionally secure construction. Additionally, we update the syntax and the security model of AH-AKE protocols to allow multiple input groups per participant and session. Furthermore, we design a concrete multi-group AH-AKE protocol by applying IHME to a state-of-the-art single-group scheme.

1 Introduction

Privacy-Preserving Key Establishment. Affiliation-Hiding Authenticated Key Exchange (AH-AKE) protocols [12,13] combine the privacy-preserving authentication properties of Secret Handshakes (SH) [2,6,21,20,19,1,14,16] with secure establishment of session keys. The typical setting of AH-AKE assumes that participants are registered members of some groups, whereby each group is administrated by a separate Group Authority (GA). Each GA is responsible for the admission of users to its group and for the issue of corresponding membership credentials. Most protocols also support revocation of users, which is also performed by the GA. The actual privacy of authentication stems from the requirement to hide the affiliations (i.e. groups) of users participating in a handshake protocol from outsiders, and also from each other, as long as their

⋆ Research partially supported by EU project CACE and by the ERC project SFEROT.

groups do not satisfy some predefined matching relationship, such as equality or dynamic matching [1].

Linkability vs. Unlinkability. AH-AKE and SH protocols come in two flavors: Linkable protocols such as [2,6,12,13] are realized using pseudonyms and allow users to recognize each other across multiple sessions. In many applications linkability is essential, e.g. in social networks it is necessary for distinguishing amongst different members of the community. Linkable protocols enjoy very efficient forms of revocation where pseudonyms are simply added to revocation lists authenticated by the GAs. In contrast, unlinkable AH-AKE and SH protocols such as [21,14,1,16] preclude the ability to link communication sessions of the same user. To handle revocation these protocols usually deploy less efficient group management schemes. In addition, the practical usage of unlinkable AH-AKE protocols for many envisioned group-oriented applications seems further away than that of protocols that support linkability.

Group Discovery Problem. The affiliation-hiding property provided by AH-AKE protocols is meaningful only if multiple groups are present in the system. Since users may belong to several groups at the same time, the inherent problem in practice is not to decide whether two given users are members of the same single group, but rather whether there is a non-empty intersection between the two sets of groups to which the users belong. Current AH-AKE and SH protocols ignore the latter problem by design, i.e. the handshake execution is performed typically with respect to only a *single* input group per participant and session. Little attention has been paid so far to possible solutions for the more general problem, termed as the *group discovery* problem in [13, p. 356]. A protocol that solves the group discovery problem would take as input a *(sub)set of groups* per participant and session, output the intersection of these sets, and, in the case that this intersection is not empty, provide a session key to the users for their subsequent communication. One of the main challenges here is to prevent the event that participants inadvertently reveal non-matching groups from their input sets to each other or to outsiders. A trivial solution for group discovery is to execute a single-group protocol for all possible combinations of membership groups, and whenever some session is successful its input group is added to the intersection set. Clearly, this solution is highly inefficient (see also discussion in Section 2). Another challenge would be the computation of the session key in a way that ensures that leakage of this key does not reveal any information about groups in the intersection set. Motivated by the importance of group discovery for the practical use of AH-AKE protocols we highlight in this paper the main challenges and explore various solutions.

1.1 Related Work

The concept of linkable Secret Handshakes [2,6] evolved into linkable Affiliation-Hiding Key Exchange [13]. These protocols commonly consider user pseudonyms as part of their group membership credentials. These pseudonyms are sent in the clear and allow for linking the sessions of the same user. Membership credentials in [2] are further bound to secret elements of a bilinear group, in [6] they are derived from private/public key pairs of a CA-oblivious PKI-enabled encryption that is realized via Schnorr signatures, and in [20, 13] they are given by full domain hash RSA signatures on the pseudonyms. All linkable protocols can be tweaked to support unlinkability through the use of one-time credentials (and pseudonyms). Due to the impracticality of this approach several unlinkable Secret Handshakes [1,14,19,16] based on reusable credentials have been proposed. However, these schemes suffer from a rather complicated group management, e.g. [1] does not support revocation, [14] requires synchronization of revocation epochs amongst members, [19] is a heavy-weight framework scheme that involves the use of group signatures and broadcast encryption techniques, while the state-of-the-art scheme in [16] uses group signatures with verifier-local revocation for group management and private conditional oblivious transfer for the handshake session in the pairing-based setting. In Section 2 we discuss in more detail why known flavors of Private Set Intersection (PSI) protocols do not give solutions to the group discovery problem.

1.2 Contribution and Organization

To this end, our goal is to explore implicit and efficient solutions for the group discovery problem in AH-AKE protocols (and consequently also in pure Secret Handshake schemes where the computation of secure session keys is not amongst the necessary requirements). We start in Section 2 by commenting on the relationship of this problem to different flavors of Private Set Intersection protocols. We show that group discovery has a more complicated setting compared to the latter (although this may not appear so at first sight). We then continue by describing potential solutions with varying efficiency and security based on probabilistic hashing.

In Section 3 we introduce *Index-Hiding Message Encoding (IHME)* — our new technique that allows for efficient group discovery in AH-AKE protocols with linear communication and computation complexity. We give a perfectly secure construction of IHME using polynomial interpolation and relate the performance of IHME-based group discovery to the mentioned hashing-based approaches.

In Section 4 we then introduce a group-discovering linkable AH-AKE protocol by applying IHME to the state-of-the-art linkable AH-AKE protocol from [13], which admits only one input group per user and session. The index-hiding property is hereby essential to preserve the affiliation-hiding property in the multi-group setting.

In Section 5 we formalize the security of linkable AH-AKE protocols by adopting a model from [13] to the setting that admits group discovery. In particular, in the definition of *affiliation hiding* we resolve the challenge of relating the outcome of the protocol containing the intersection of input group (sub)sets to the knowledge of the adversary that may have valid membership credentials for some of these groups. Following the model we finally give the security analysis of our scheme in Section 6.

2 The Group Discovery Problem and Potential Solutions

In this section we focus on various solutions for the group discovery problem in AH-AKE protocols. One of these solutions is trivial but inefficient, whereas some other solutions, while being more sophisticated, may be less secure.

In order to illustrate different approaches we briefly introduce the setting for the group discovery problem in AH-AKE protocols. Consider N different groups $G_1, \ldots, G_N$ managed by distinct group authorities. We assume that user U_1 is a registered member of n_1 groups, i.e. U_1 holds a set of n_1 different membership credentials $\{(G_i, \mathsf{cred}_i^1)\}_i$. Similarly, U_2 is a registered member of n_2 groups holding own set of n_2 credentials $\{(G_j, \mathsf{cred}_j^2)\}_j$. To simplify the exposition, let us further assume that $n_1 = n_2$ and use the notation $n = n_1 = n_2$. At a high level, the goal of the group discovery problem in AH-AKE protocols is to execute a handshake session between U_1 and U_2 such that at the end of the session (a) the users identify the subset of groups for which both have respective membership credentials (without disclosing information about any other credentials they possess), and (b) if this subset of groups is non-empty, then the two users agree on a secret key. Current AH-AKE (and SH) protocols admit only one input group per handshake participant and session, basically allowing for privacy-preserving matching of some input groups G_i and G_j used by U_1 and U_2, respectively. In our description we will utilize this ability of U_1 and U_2 to execute such single-group AH-AKE protocols using any of their membership credentials.

2.1 Relationship to (Authorized) Private Set Intersection

We investigate first whether the group discovery problem can be solved in a more general way using Private Set Intersection (PSI) protocols such as [9,17,10,15, 8,11]. In a typical PSI setting users execute the protocol using sets of elements as inputs. Each user has its own input set and the main goal of a PSI protocol is to allow users to learn the intersection of their input sets without disclosing any information about further elements. One might attempt to design a group discovery protocol by simply letting group credentials be random nonces from a large domain (but identical for all members of the group) and use a PSI protocol to check if the two users have nonces of the same group. With this solution, however, a number of problems arise: (a) providing the same credential to all group members precludes member revocation since users can trivially create and admit new members to their groups without the GA noticing it, simply by revealing their nonces to other users, (b) the proposed technique leads to an AH-AKE protocol which is not affiliation-hiding in the sense defined in Section 5 as shown in our full version, and (c) although PSI protocols with linear computation overhead are known [8,11], our group discovery protocol presented in Section 4 can be implemented more efficiently as it relies on simpler building blocks.

A related class of protocols [7,8], called Authorized PSI (APSI), strengthens the requirements of PSI protocols in that the computed intersection of the users' inputs must contain authorized elements only, i.e. elements that have been previously certified by some trusted authority. A technique for computing the intersection of certified sets has been introduced in [4]. One may think that authorization of elements in APSI corresponds to the registration process of users to groups in AH-AKE protocols. However, the APSI setting assumes that the *same* authority certifies all elements in the input sets. In contrast, the AH-AKE setting explicitly requires existence of *multiple* independent group authorities providing users with membership certificates. In addition, problem (a) in the PSI setting (support for revocation) also applies here. Since neither PSI nor APSI protocols help to solve the group discovery problem directly, we discuss in the following several alternatives before coming to our most efficient solution.

2.2 Possible Solutions

Naïve Approach. The trivial solution is for U_1 and U_2 to use a single-group AH-AKE protocol for any possible combination of their membership groups. This requires n^2 different AH-AKE sessions, which might be too high in practice.

Reducing the Overhead by Using Hashing. A possible improvement that decreases the overhead involves the usage of hashing. We describe here only the basic ideas of this solution, since the focus of this paper is on a more efficient solution based on a new encoding technique, which we introduce in Section 3.

In the hashing-based approach, the parties use a random hash function h, which is either chosen in advance or jointly defined by the two parties. The hash function maps arbitrary values to an output in the range $[1, B], B \in \mathbb{N}$, namely into one of B bins. Each party then assigns its membership credentials in group G_i into bin $h(i)$. Now, when U_1 and U_2 meet, they need not run the AH-AKE protocol between each of the n^2 combinations of their potential input groups. Instead, the protocol must only be run between the membership credentials that were mapped by both parties to the same bin. Indeed, for every group G_i for which both U_1 and U_2 have membership credentials, both parties map these credentials to the same bin $h(i)$ and will, therefore, run the AH-AKE protocol with these credentials.

The basic idea described above succeeds in finding every match between membership credentials of the two parties. However, in order to protect privacy, a protocol which is based on this approach must hide from each party how many credentials were mapped by the other party to each of the bins (otherwise some data is leaked; for example, if the first bin of U_1 is empty then U_2 learns that U_1 in not a member of any group G_i for which $h(i) = 1$). Hiding the number of items in every bin can be done (following [9]) by finding a bound M such that the following property holds with high probability: when n items are mapped by a random hash function to B bins, then no more than M items are mapped to any single bin. Given this bound M, each party first maps its credentials to the B bins, and then adds to each bin that has less than M credentials, additional "dummy" values, which are indistinguishable from real credentials, so that the total number of items in the bin is M. The protocol now requires to run M^2 handshakes of the single-group AH-AKE protocols for every bin, resulting in the total of BM^2 sessions. In order to set the right parameters, we can use the following well known fact [18, Theorem 1]:

Fact 1. *If n items are mapped at random to $B = n/\log n$ bins, then the probability that there is a bin with more than $M = O(\log n)$ items is $o(1)$.*

The communication overhead of the protocol is obviously $O(BM^2)$. Also, the computation requires each party to run the single-group handshake BM^2 times. Plugging in the parameters $B = n/\log n$ and $M = O(\log n)$, we get that the communication and computation overheads are both $O(n \log n)$. (An even more efficient variant of this technique, based on *balanced allocation hashing*, is explored in the full version of this paper. Its communication and computation complexity is $O(n \log \log n)$. That variant leaks however some additional information about group affiliations.)

3 Index-Hiding Message Encoding

The main tool we will use in our protocol is a new primitive called *Index-Hiding Message Encoding (IHME)*. By this term we understand a technique that pools a set of input messages $m_1, \ldots, m_n \in \mathcal{M}$ (where $\mathcal{M}$ is a message space) into a single data structure $\mathcal{S}$. Any message can be recovered from $\mathcal{S}$ individually by addressing it via its *index* which is arbitrarily chosen from an index set $\mathcal{I}$ and specified at encoding-time. If it is impossible for an adversary to reveal information about the deployed indices by inspecting $\mathcal{S}$ then the scheme is called *index-hiding.* This notion is now formalized by first presenting the syntax of the related concept of *index-based message encoding* and its correctness definition, and then by giving a game-based definition of the index-hiding property.

Definition 1 (Index-Based Message Encoding). *An* index-based message encoding *scheme* $(\mathsf{iEncode}, \mathsf{iDecode})$ *over an index space $\mathcal{I}$ and a message space $\mathcal{M}$ consists of two efficient algorithms:*

$\mathsf{iEncode}(\mathcal{P})$ *On input consisting of a tuple of index-message pairs $\mathcal{P} = \{(i_1, m_1), \ldots, (i_n, m_n)\} \subseteq \mathcal{I} \times \mathcal{M}$ with distinct indices $i_1, \ldots, i_n$, this algorithm outputs an encoding $\mathcal{S}$.*

$\mathsf{iDecode}(\mathcal{S}, i)$ *On input of an encoding $\mathcal{S}$ and an index $i \in \mathcal{I}$ this algorithm outputs a message $m \in \mathcal{M}$.*

An index-based message encoding scheme is correct *if* $\mathsf{iDecode}(\mathsf{iEncode}(\mathcal{P}), i_j) = m_j$ *for all $j \in \{1, \ldots, n\}$ and all tuples $\mathcal{P} = \{(i_1, m_1), \ldots, (i_n, m_n)\} \subseteq \mathcal{I} \times \mathcal{M}$ with distinct indices i_j.*

An index-based message encoding scheme is called *index-hiding* if it hides the indices in which the messages are encoded. That is, it ensures that an attacker, who sees an encoding $\mathcal{S}$ and might even know some of the indices and corresponding messages, cannot identify any other indices in which messages are encoded.

Given this property, which is formalized below, an index-hiding message encoding can be used for solving group discovery in AH-AKE in the following way:

The first party, U_1, IHME-encodes its first messages m_j of each of the AH-AKE protocols in the groups it belongs to, using fixed indices i_j assigned to these groups. The resulting encoding $\mathcal{S}$ is sent to the second party, U_2, which can retrieve and answer the messages encoded in the indices corresponding to U_2's credentials. However, messages corresponding to other groups cannot be recognized by U_2, since it does not have the credentials required to participate in the AH-AKE protocols of the corresponding groups, and therefore it is in the same situation as someone participating in a one-on-one AH-AKE protocol without being a member of the relevant group. U_2 therefore cannot identify neither these messages nor the deployed indices.

We note that our construction of IHME, presented below, has the attractive property that encodings $\mathcal{S}$ are of the same size (in bits) as the sets of embedded messages. Given this property one can pass n messages associated with n specific indices in an encoding $\mathcal{S}$ which is only as large as the original set of messages, and with the following two properties: (a) for each index known to the receiving party, decoding is possible in a single attempt (since the receiving party knows where to look for the message), and (b) the indices of messages which the other party is not authorized to decode are kept hidden.

Definition 2 (Index-Hiding Message Encoding (IHME)). *Let* $\mathsf{IHME} = (\mathsf{iEncode}, \mathsf{iDecode})$ *denote a correct index-based message encoding scheme over index space* $\mathcal{I}$ *and message space* $\mathcal{M}$*. Let* $b \in \{0,1\}$ *be a randomly chosen bit and let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a PPT adversary that participates in the following game.*

$\mathsf{Game}^{\mathsf{ihide},b}_{\mathcal{A},\mathsf{IHME}}(\kappa)$:

- $(I_0, I_1, M', St) \leftarrow \mathcal{A}_1(1^\kappa)$ *such that* $I_0, I_1 \subseteq \mathcal{I}$ *with* $|I_0| = |I_1| = n$, *and* $M' = (m'_1, \ldots, m'_{|I_0 \cap I_1|})$ *with each* $m'_j \in \mathcal{M}$*; (the adversary chooses two subsets of* n *indices each, as well as a message* m'_j *for each index* i_j *in the intersection of the sets);*
- *denote the indices in* $I_b \backslash I_{1-b}$ *as* $\{i_1, \ldots, i_r\}$ *and define* $m_1, \ldots, m_r \xleftarrow{\$} \mathcal{M}$, *(additional* $r = n - |I_0 \cap I_1|$ *messages are chosen uniformly at random in the message space),*
  let $\mathcal{S} \leftarrow \mathsf{iEncode}(\{(i_j, m'_j) \mid i_j \in I_0 \cap I_1\} \cup \{(i_j, m_j) \mid i_j \in I_b \setminus I_{1-b}\})$, *(the messages are encoded for the indices in* I_b *);*
- $b' \leftarrow \mathcal{A}_2(St, \mathcal{S})$ *(the adversary is given* S *and attempts to find* b*);*
- *if* $b' = b$ *then return 1 else return 0.*

The advantage of $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}^{\mathsf{ihide}}_{\mathcal{A},\mathsf{IHME}}(\kappa) := \left|\Pr[\mathsf{Game}^{\mathsf{ihide},0}_{\mathcal{A},\mathsf{IHME}}(\kappa) = 1] - \Pr[\mathsf{Game}^{\mathsf{ihide},1}_{\mathcal{A},\mathsf{IHME}}(\kappa) = 1]\right|.$$

By $\mathsf{Adv}^{\mathsf{ihide}}_{\mathsf{IHME}}(\kappa)$ *we denote the maximum advantage over all PPT adversaries* $\mathcal{A}$*. We say that* IHME *provides* index-hiding *if this advantage is negligible in* κ*. Moreover, if* $\mathsf{Adv}^{\mathsf{ihide}}_{\mathsf{IHME}}(\kappa) = 0$ *for all* κ*, the* IHME*-scheme is called* perfect.

The definition above enables the adversary to choose the sets of indices, and the messages corresponding to the intersection of the sets. The other messages are chosen at random. The adversary is given one of the two sets and its goal is to identify which set it is. The definition requires that the adversary's success probability be negligible.

An Implementation of Perfect IHME. One way to efficiently implement IHME is by using polynomial interpolation in a finite field $\mathbb{F}$. The index and message spaces are then specified as $\mathcal{I} = \mathcal{M} = \mathbb{F}$. Algorithms iEncode and iDecode are specified as follows:

iEncode($\mathcal{P}$) On input of $\mathcal{P} = \{(i_1, m_1), \ldots, (i_n, m_n)\} \subseteq \mathcal{I} \times \mathcal{M} = \mathbb{F}^2$, the encoding is defined as the list $\mathcal{S} = (a_{n-1}, \ldots, a_0)$ of coefficients of the (unique) polynomial $f = \sum_{k=0}^{n-1} a_k x^k \in \mathbb{F}[x]$ which satisfies $\forall (i_j, m_j) \in \mathcal{P} : f(i_j) = m_j$. This polynomial can easily be determined by Lagrange interpolation.

iDecode($\mathcal{S}, i$) On input of $\mathcal{S} = (a_{n-1}, \ldots, a_0)$ and index $i \in \mathcal{I}$ this algorithm outputs the evaluation $m = f(i) = \sum_{k=0}^{n-1} a_k i^k$ of f at position i.

The correctness of the proposed IHME scheme is obvious. Its index-hiding ability is assured by the following theorem, proven in the full version of this paper based on the fact that the distributions of the encodings of I_0 and I_1 are identical.

Theorem 1 (Index-Hiding Property of our IHME Construction). *The proposed IHME scheme provides perfect index-hiding.*

In our solution for the group discovery problem in AH-AKE protocols, the index set $\mathcal{I}$ is identified with the set of all possible groups. U_1 follows, for each of the n groups that it is affiliated with, the computation rules of a single-group AH-AKE handshake. The computed first messages from all these handshake instances are then encoded into a single structure using the IHME approach by considering the identifiers of the n groups (e.g. hashes of the public parameters) as indices for the corresponding messages. The encoding is sent over to U_2 which extracts the handshake messages for only the groups it is affiliated with. Note that for all matching groups G_i (i.e. groups in which both U_1 and U_2 are members) the first messages of handshake instances are correctly transferred from U_1 to U_2. The IHME technique is then applied independently to all subsequently exchanged handshake messages. Observe that for the secure deployment of IHME (as per Definition 2) it is essential that messages exchanged between users in the given single-group AH-AKE handshake are polynomially indistinguishable from random in $\mathcal{M} = \mathbb{F}$. This property is satisfied by some protocols, in particular by the linkable AH-AKE protocol from [13] that is underlying our construction with implicit group discovery, as presented in Section 4.

On General Performance of IHME. The IHME technique suggested above has the nice property of zero message expansion: in iEncode( ), the length of the input messages is equal to the length of the output IHME encoding $\mathcal{S}$. The communication complexity of our protocol is therefore $O(n)$. Since users perform computations of a single-group handshake only for groups in which they are members, they need to participate in only n group handshakes. Thus, in contrast to possible solutions from Section 2.2 our IHME technique with perfect index-hiding can solve the group discovery problem with *linear* communication and computation complexity[1] as summarized in Table 1.

Table 1. Solutions for the group discovery problem with n input groups per participant

Technique	Computation (AH-AKE invocations)	Communication	Remarks
Naïve approach	$O(n^2)$	$O(n^2)$	
Hashing into bins	$O(n \log n)$	$O(n \log n)$	
Balanced allocation hashing	$O(n \log \log n)$	$O(n \log \log n)$	not privacy preserving
Our perfect IHME	$O(n)$	$O(n)$	

We notice that the general technique for solving group discovery in AH-AKE schemes based on our IHME construction clearly outperforms all other solutions suggested above.

4 Affiliation-Hiding Key Exchange with Group Discovery

In this section we illustrate how the IHME technique can be used to construct a concrete AH-AKE protocol with *implicit* solution to the group discovery problem. Our protocol is based on the state-of-the-art linkable AH-AKE (LAH-AKE) scheme from [13], which allows to privately check the match of a single group per user and handshake session only. This protocol has the nice property that its messages remain indistinguishable from random strings to anyone who is not a group member. Therefore, we can directly apply IHME to compress the complexity of group discovery without assuming any further building blocks. However, in order to address the group discovery problem implicitly we have to update the syntax of LAH-AKE protocols.

4.1 Syntax of LAH-AKE with Implicit Group Discovery

A LAH-AKE scheme, which admits *multiple* group credentials as input per user and session, denoted $\mathsf{MLAH\text{-}AKE}$, consists of four algorithms:

$\mathsf{CreateGroup}(1^\kappa)$**.** This probabilistic algorithm sets up a new group G and is executed by the corresponding group authority (GA). On input of security parameter 1^κ it generates a public/private group key pair $(G.\mathsf{pk}, G.\mathsf{sk})$, initializes the group's pseudonym revocation list $G.\mathsf{prl}$ to $\emptyset$ and outputs public group parameters $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl})$, and private key $G.\mathsf{sk}$. It is assumed that public key $G.\mathsf{pk}$ may serve as a non-ambiguous identifier for group G.

[1] The overhead of interpolating the polynomial in the IHME method is $O(n^2)$ multiplications. However, the overhead of these operations is negligible compared with the overhead of computing exponentiations in the AH-AKE handshake protocols.

$\mathsf{AddUser}(U, G)$**.** This is a protocol that is executed between a prospective group member U and the group authority of G. User U presents a pseudonym id and is issued a private membership credential $\mathsf{cred}_{G.\mathsf{pk}}$ for id in group G. It is legitimate for users to register the same pseudonym id in different groups. The communication channel between U and G is assumed to be authentic.

$\mathsf{Handshake}(U_i, U_j)$**.** This is the key exchange protocol (handshake), executed between two users U_i and U_j. The input for U_i is $(\mathsf{id}_i, \mathcal{G}_i, r_i)$ where id_i is his pseudonym, $\mathcal{G}_i$ is a set of triples of the form $(G.\mathsf{pk}, \mathsf{cred}_{G.\mathsf{pk}}, G.\mathsf{prl})$, and $r_i \in \{\mathsf{init}, \mathsf{resp}\}$. All values $\mathsf{cred}_{G.\mathsf{pk}}$ in $\mathcal{G}_i$ are credentials previously registered in the respective group $G.\mathsf{pk}$ by using the $\mathsf{AddUser}$ algorithm on pseudonym id_i. By $G.\mathsf{prl}$ we denote the respective pseudonym revocation list. For user U_j the protocol's input $(\mathsf{id}_j, \mathcal{G}_j, r_j)$ is defined analogously.

Users keep track of the state of created $\mathsf{Handshake}(\mathsf{id}, \mathcal{G}, r)$ protocol sessions π through session variables that are initialized as follows: $\pi.\mathsf{state} \leftarrow \mathsf{running}$, $\pi.\mathsf{id} \leftarrow \mathsf{id}$, $\pi.\mathcal{G} \leftarrow \mathcal{G}$ and $(\pi.\mathsf{key}, \pi.\mathsf{partner}, \pi.\mathsf{groups}) \leftarrow (\perp, \perp, \emptyset)$. At some point the protocol will complete and $\pi.\mathsf{state}$ is then updated to either $\mathsf{rejected}$ or $\mathsf{accepted}$. In the latter case $\pi.\mathsf{key}$ is set to the established session key (of length κ), the handshake partner's pseudonym is assigned to $\pi.\mathsf{partner}$, and $\pi.\mathsf{groups}$ holds a non-empty set of group identifiers.

$\mathsf{Revoke}(G, \mathsf{id})$**.** This algorithm is executed by the group authority of G and results in the update of G's pseudonym revocation list: $G.\mathsf{prl} \leftarrow G.\mathsf{prl} \cup \{\mathsf{id}\}$.

Definition 3 (Correctness of MLAH-AKE). *Assume that users U_i and U_j participate in a* Handshake *protocol with inputs $(\mathsf{id}_i, \mathcal{G}_i, r_i)$ and $(\mathsf{id}_j, \mathcal{G}_j, r_j)$, respectively, and let π_i and π_j denote the corresponding sessions. By $\mathcal{G}_\cap$ we denote the set of groups that appear in both $\mathcal{G}_i$ and $\mathcal{G}_j$ with the restriction that neither id_i nor id_j are contained in the respective groups' revocation lists. The* MLAH-AKE *scheme is called* correct *if (1) π_i and π_j complete in the same state, which is* accepted *iff $(\mathcal{G}_\cap \neq \emptyset \wedge r_i \neq r_j)$, and (2) if both sessions accept then $(\pi_i.\mathsf{key}, \pi_i.\mathsf{partner}, \pi_i.\mathsf{id}) = (\pi_j.\mathsf{key}, \pi_j.\mathsf{id}, \pi_j.\mathsf{partner})$ and $\pi_i.\mathsf{groups} = \pi_j.\mathsf{groups} = \mathcal{G}_\cap$.*

4.2 Number-Theoretic Assumptions and Building Blocks

Assumptions. The security of our protocol builds on the hardness of the following RSA problem, which is a standard RSA assumption for *safe* moduli, also used in the design of AH-AKE and SH protocols from [20,12,13].

Definition 4 (RSA Assumption on Safe Moduli). *Let $\mathsf{RSA\text{-}G}(\kappa')$ be a probabilistic algorithm that outputs pairs (n, e) where (a) $n = pq$ for random κ'-bit primes $p \neq q$, (b) $p = 2p' + 1, q = 2q' + 1$ for primes p', q', and (c) $e \in \mathbb{Z}_{\varphi(n)}$ is coprime to $\varphi(n)$. The RSA-success probability of a PPT solver $\mathcal{A}$ is defined as*

$$\mathsf{Succ}^{\mathsf{rsa}}_{\mathcal{A}}(\kappa') = \Pr\big[(n, e) \leftarrow \mathsf{RSA\text{-}G}(\kappa');\ z \xleftarrow{\$} \mathbb{Z}_n^*;\ m \leftarrow \mathcal{A}(n, e, z) \text{ with } m^e = z\big].$$

The RSA assumption on safe moduli *states that the maximum RSA-success probability $\mathsf{Succ}^{\mathsf{rsa}}(\kappa')$ (defined over all PPT solvers $\mathcal{A}$) is negligible in κ'.*

Perfect IHME. Let κ, κ' be security parameters (where κ' is polynomially dependent on κ) and p be the smallest prime number satisfying $p > 2^{2\kappa' + \kappa}$. A central building block of our $\mathsf{MLAH\text{-}AKE}$ protocol is the perfect IHME scheme presented in Section 3, defined over finite field $\mathbb{F} = GF(p)$. (As alternative, finite field $GF(2^{2\kappa' + \kappa})$ could be used. However, in this paper we use $GF(p)$ to simplify notations.)

Hash Functions. The protocol makes use of three different hash functions (modeled as random oracles):

$$H : \{0,1\}^* \to [0, p-1] \qquad H' : \{0,1\}^* \to \{0,1\}^\kappa \qquad H^* : \{0,1\}^* \to [0, p-1]$$

For convenience, for each $n \in \mathbb{N}$ of length $2\kappa'$ we define

$$H_n : \{0,1\}^* \to \mathbb{Z}_n;\ x \mapsto H^*(n \,\|\, x) \bmod n.$$

4.3 The Protocol Specification

$\mathsf{CreateGroup}(1^\kappa)$ *Algorithm.* Group authorities setup new group parameter sets as follows: in a first step, two κ'-bit primes $p, q \in \mathbb{N}$ with $p = 2p' + 1$ and $q = 2q' + 1$ for prime numbers p' and q' are picked. For $n = pq$ an exponent $e \in \mathbb{Z}_{\varphi(n)}$ which is coprime to $\varphi(n) = (p-1)(q-1) = 4p'q'$ is chosen, and $d = e^{-1} \pmod{\varphi(n)}$ is computed.

As $\mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ the largest element order in $\mathbb{Z}_n^*$ is $\operatorname{lcm}(\varphi(p), \varphi(q)) = 2p'q' = \varphi(n)/2$, and hence $\mathbb{Z}_n^*$ is not cyclic. Let the $\mathsf{CreateGroup}$ algorithm pick an element $g \in \mathbb{Z}_n^*$ with $\operatorname{ord}(g) \geq p'q'$ and $g^{p'q'} \neq \pm 1$. It follows that $\operatorname{ord}(g) = 2p'q'$ and that $-1 \notin \langle g \rangle$, and we may infer $\mathbb{Z}_n^* \cong \langle -1 \rangle \times \langle g \rangle$. As about a half of the elements in $\mathbb{Z}_n^*$ have the desired two properties [13], g can easily be found by random sampling and testing.

The algorithm sets $G.\mathsf{pk} \leftarrow (n, g, e)$, $G.\mathsf{prl} \leftarrow \emptyset$ and $G.\mathsf{sk} \leftarrow d$, and outputs $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl})$ and $G.\mathsf{sk}$ as public and private key, respectively.

$\mathsf{AddUser}(U, G)$ *Protocol.* To admit a new member to a group the corresponding GA issues as credential $\mathsf{cred}_{G.\mathsf{pk}}$ the full domain hash RSA signature [3] on the pseudonym id specified by the user, i.e. $\mathsf{cred}_{G.\mathsf{pk}} = H_n(\mathsf{id})^{G.\mathsf{sk}} \bmod n$. The communication between U and G is assumed to be authentic and confidential.

$\mathsf{Handshake}((\mathsf{id}_i, \mathcal{G}_i, \mathsf{init}), (\mathsf{id}_j, \mathcal{G}_j, \mathsf{resp}))$ *Protocol.* The handshake protocol is executed between two users U_i and U_j holding corresponding pseudonyms id_i and id_j as well as membership lists $\mathcal{G}_i$ and $\mathcal{G}_j$, respectively. Each user's membership list contains triples $(G.\mathsf{pk}, \mathsf{cred}_{G.\mathsf{pk}}, G.\mathsf{prl})$ for all affiliations of that user. The handshake protocol is detailed in Figure 1.

The lines where the numbering is formatted in bold face coincide with those from [13]; in particular this includes the calculation of the $\theta = (-1)^b g^t \mathsf{cred} + kn$ values (lines 4–8), the partial keys $r = (\theta^e H_n(\mathsf{id})^{-1})^{2t}$ and the confirmation messages c (lines 19–20). Lines 17 and 22 effectively implement user revocation.

Innovative in this protocol is the parallel transmission of multiple θ and c values encoded as IHME-structures $\mathcal{S}$ and $\mathcal{S}'$, respectively. Note the usage of RSA moduli n as group specific indices. The lists $\mathcal{T}$ and $\mathcal{R}$ are not transmitted, but hold the inner state of the protocol.

Note that protocol correctness demands that the string X in the third code block (lines 26–32) is mounted in the same order for both U_i and U_j. This can be achieved by letting the corresponding FOR-loop iterate in the order of ascending n_i.

$\mathsf{Revoke}(G, \mathsf{id})$ *Algorithm.* The revocation of pseudonyms is handled by the particular group authority of G by including the pseudonym id into the corresponding pseudonym revocation list $G.\mathsf{prl}$. We assume that this list is distributed authentically to all members of the group.

4.4 Protocol Correctness, Efficiency Analysis, and Optimizations

In the following we show why the protocol is correct and discuss its concrete efficiency, including possible optimizations. The actual security analysis of the protocol is postponed to Section 6 after the specification of the security model with regard to the group discovery problem.

Correctness. Let $(n, g, e) = G.\mathsf{pk}$ denote a group to which U_i and U_j are both registered, i.e. U_i owns a credential $\mathsf{cred}_{G.\mathsf{pk}} = H_n(\mathsf{id}_i)^d \bmod n$ for pseudonym id_i, while U_j possesses credential $H_n(\mathsf{id}_j)^d \bmod n$ for id_j. Then, by construction, the value $\mathsf{iDecode}(\mathcal{S}_i, n)$ has the form $\theta_i = (-1)^{b_i} g^{t_i} H_n(\mathsf{id}_i)^d \bmod n$. The value r_j computed by U_j for group $G.\mathsf{pk}$ is

$$r_j = (\theta_i{}^e H_n(\mathsf{id}_i)^{-1})^{2t_j} = (((-1)^{b_i} g^{t_i} H_n(\mathsf{id}_i)^d)^e H_n(\mathsf{id}_i)^{-1})^{2t_j} = g^{2et_it_j} \pmod n.$$

Symmetrically, for group $G.\mathsf{pk}$ user U_i computes the same value $r_i = g^{2et_it_j}$ from $\mathsf{iDecode}(\mathcal{S}_j, n)$ and id_j. The protocol's correctness is now verifiable by inspection. The case that $H_n(\mathsf{id})^{-1}$ is not defined occurs only with negligible probability.

Efficiency Analysis. The computational costs of the $\mathsf{Handshake}$ protocol mainly consist of the exponentiations by t_i (resp. t_j), which are executed twice per group. Precisely, the computational effort a user U_i starting a $\mathsf{Handshake}(\mathsf{id}_i, \mathcal{G}_i, \mathsf{init})$ protocol session has to stem can be estimated by $2|\mathcal{G}_i|$ exponentiations with modulus size $2\kappa'$, where $|\mathcal{G}_i|$ denotes the number of groups U_i is member in. That is, the computational overhead scales linearly with the number of affiliations. Also the size of IHME-encodings $\mathcal{S}, \mathcal{S}'$ grows linearly with the number of affiliations. More precisely, the total communication complexity of the handshake amounts to $2(2\kappa' + \kappa)(|\mathcal{G}_i| + |\mathcal{G}_j|)$ bits (without considering pseudonyms that can be short).

```
1   $U_i$ ON INPUTS $(\mathsf{id}_i, \mathcal{G}_i, \mathsf{init})$
2   $\mathcal{P}_i \leftarrow \emptyset$, $\mathcal{T}_i \leftarrow \emptyset$
3   FOR ALL $(G.\mathsf{pk}, \mathsf{cred}_{G.\mathsf{pk}}, G.\mathsf{prl}) \in \mathcal{G}_i$:
4       LET $(n_i, g_i, e_i) = G.\mathsf{pk}$
5       $(b_i, t_i) \leftarrow_R \mathbb{Z}_2 \times \mathbb{Z}_{n_i/2}$
6       $\theta'_i \leftarrow (-1)^{b_i}(g_i)^{t_i}\mathsf{cred}_{G.\mathsf{pk}} \bmod n_i$
7       $k_i \leftarrow_R [0, \lfloor p/n_i \rfloor - 1]$
8       $\theta_i \leftarrow \theta'_i + k_i n_i$
9       $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{(n_i, \theta_i)\}$
10      $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \{(G.\mathsf{pk}, t_i, G.\mathsf{prl})\}$
11  $\mathcal{S}_i \leftarrow \mathsf{iEncode}(\mathcal{P}_i)$
12
13  $\mathsf{sid}_i \leftarrow m_i \| m_j$
14  $\mathcal{P}'_i \leftarrow \emptyset$, $\mathcal{R}_i \leftarrow \emptyset$
15  FOR ALL $(G.\mathsf{pk}, t_i, G.\mathsf{prl}) \in \mathcal{T}_i$:
16      LET $(n_i, g_i, e_i) = G.\mathsf{pk}$
17      IF $\mathsf{id}_j \notin G.\mathsf{prl}$:
18          $\theta_j \leftarrow \mathsf{iDecode}(\mathcal{S}_j, n_i)$
19          $r_i \leftarrow ({\theta_j}^{e_i} H_{n_i}(\mathsf{id}_j)^{-1})^{2t_i} \bmod n_i$
20          $c_i \leftarrow H(G.\mathsf{pk} \| r_i \| \mathsf{sid}_i \| \mathsf{init})$
21          $\mathcal{R}_i \leftarrow \mathcal{R}_i \cup \{(G.\mathsf{pk}, r_i)\}$
22      ELSE: $c_i \leftarrow_R [0, p-1]$
23      $\mathcal{P}'_i \leftarrow \mathcal{P}'_i \cup \{(n_i, c_i)\}$
24  $\mathcal{S}'_i \leftarrow \mathsf{iEncode}(\mathcal{P}'_i)$
25
26  $X_i \leftarrow$ "", $\mathsf{groups}_i \leftarrow \emptyset$
27  FOR ALL $(G.\mathsf{pk}, r_i) \in \mathcal{R}_i$:
28      LET $(n_i, g_i, e_i) = G.\mathsf{pk}$
29      $c_j \leftarrow \mathsf{iDecode}(\mathcal{S}'_j, n_i)$
30      IF $c_j = H(G.\mathsf{pk} \| r_i \| \mathsf{sid}_i \| \mathsf{resp})$:
31          $\mathsf{groups}_i \leftarrow \mathsf{groups}_i \cup \{G.\mathsf{pk}\}$
32          $X_i \leftarrow X_i \| G.\mathsf{pk} \| r_i$
33
34  IF $\mathsf{groups}_i \neq \emptyset$ THEN
35      $\mathsf{key}_i \leftarrow H'(X_i \| \mathsf{sid}_i)$
36      $\mathsf{partner}_i \leftarrow \mathsf{id}_j$
37      TERMINATE WITH "ACCEPT"
38  ELSE
39      $(\mathsf{key}_i, \mathsf{partner}_i) \leftarrow (\bot, \bot)$
40      TERMINATE WITH "REJECT"
```

```
(line 12)  $\xrightarrow{m_i \;=\; (\mathsf{id}_i, \mathcal{S}_i)}$
(line 13)  $\xleftarrow{m_j \;=\; (\mathsf{id}_j, \mathcal{S}_j)}$
(line 24)  $\xrightarrow{\mathcal{S}'_i}$
(line 25)  $\xleftarrow{\mathcal{S}'_j}$
```

```
$U_j$ ON INPUTS $(\mathsf{id}_j, \mathcal{G}_j, \mathsf{resp})$
$\mathcal{P}_j \leftarrow \emptyset$, $\mathcal{T}_j \leftarrow \emptyset$
FOR ALL $(G.\mathsf{pk}, \mathsf{cred}_{G.\mathsf{pk}}, G.\mathsf{prl}) \in \mathcal{G}_j$:
    LET $(n_j, g_j, e_j) = G.\mathsf{pk}$
    $(b_j, t_j) \leftarrow_R \mathbb{Z}_2 \times \mathbb{Z}_{n_j/2}$
    $\theta'_j \leftarrow (-1)^{b_j}(g_j)^{t_j}\mathsf{cred}_{G.\mathsf{pk}} \bmod n_j$
    $k_j \leftarrow_R [0, \lfloor p/n_j \rfloor - 1]$
    $\theta_j \leftarrow \theta'_j + k_j n_j$
    $\mathcal{P}_j \leftarrow \mathcal{P}_j \cup \{(n_j, \theta_j)\}$
    $\mathcal{T}_j \leftarrow \mathcal{T}_j \cup \{(G.\mathsf{pk}, t_j, G.\mathsf{prl})\}$
$\mathcal{S}_j \leftarrow \mathsf{iEncode}(\mathcal{P}_j)$

$\mathsf{sid}_j \leftarrow m_i \| m_j$
$\mathcal{P}'_j \leftarrow \emptyset$, $\mathcal{R}_j \leftarrow \emptyset$
FOR ALL $(G.\mathsf{pk}, t_j, G.\mathsf{prl}) \in \mathcal{T}_j$:
    LET $(n_j, g_j, e_j) = G.\mathsf{pk}$
    IF $\mathsf{id}_i \notin G.\mathsf{prl}$:
        $\theta_i \leftarrow \mathsf{iDecode}(\mathcal{S}_i, n_j)$
        $r_j \leftarrow ({\theta_i}^{e_j} H_{n_j}(\mathsf{id}_i)^{-1})^{2t_j} \bmod n_j$
        $c_j \leftarrow H(G.\mathsf{pk} \| r_j \| \mathsf{sid}_j \| \mathsf{resp})$
        $\mathcal{R}_j \leftarrow \mathcal{R}_j \cup \{(G.\mathsf{pk}, r_j)\}$
    ELSE: $c_j \leftarrow_R [0, p-1]$
    $\mathcal{P}'_j \leftarrow \mathcal{P}'_j \cup \{(n_j, c_j)\}$
$\mathcal{S}'_j = \mathsf{iEncode}(\mathcal{P}'_j)$

$X_j \leftarrow$ "", $\mathsf{groups}_j \leftarrow \emptyset$
FOR ALL $(G.\mathsf{pk}, r_j) \in \mathcal{R}_j$:
    LET $(n_j, g_j, e_j) = G.\mathsf{pk}$
    $c_i \leftarrow \mathsf{iDecode}(\mathcal{S}'_i, n_j)$
    IF $c_i = H(G.\mathsf{pk} \| r_j \| \mathsf{sid}_j \| \mathsf{init})$:
        $\mathsf{groups}_j \leftarrow \mathsf{groups}_j \cup \{G.\mathsf{pk}\}$
        $X_j \leftarrow X_j \| G.\mathsf{pk} \| r_j$

IF $\mathsf{groups}_j \neq \emptyset$ THEN
    $\mathsf{key}_j \leftarrow H'(X_j \| \mathsf{sid}_j)$
    $\mathsf{partner}_j \leftarrow \mathsf{id}_i$
    TERMINATE WITH "ACCEPT"
ELSE
    $(\mathsf{key}_j, \mathsf{partner}_j) \leftarrow (\bot, \bot)$
    TERMINATE WITH "REJECT"
```

Fig. 1. Specification of $\mathsf{Handshake}((\mathsf{id}_i, \mathcal{G}_i, \mathsf{init}), (\mathsf{id}_j, \mathcal{G}_j, \mathsf{resp}))$

Further Optimizations. The following optimization idea would render the Handshake protocol slightly more efficient. It is based on the observation that in the Handshake protocol both the θ values and the confirmation tags c are transferred by IHME using the same finite field $\mathbb{F} = GF(p)$. The protocol would stay secure if the confirmation tags c would be shortened from $2\kappa' + \kappa$ bits to just κ bits. To implement this, the confirmation tags c have to be computed by an auxiliary hash function whose range is $\mathbb{F}'$, where $\mathbb{F}'$ is a finite field of order $\approx 2^\kappa$, and the IHME scheme for transferring $\mathcal{S}'$ would have to be defined over $\mathbb{F}'$ instead of $\mathbb{F}$. The deployment of this idea would save $2\kappa'(|\mathcal{G}_i| + |\mathcal{G}_j|)$ communication bits, resulting in the total complexity of $(2\kappa' + 2\kappa)(|\mathcal{G}_i| + |\mathcal{G}_j|)$.

5 Security Model for LAH-AKE with Group Discovery

In this section we introduce the security model for LAH-AKE protocols while taking into account various challenges implied by the group discovery problem and the updated syntax of such protocols, by modifying the current state-of-the-art model from [13].

5.1 Adversary Model

After describing the basic set of adversarial queries we define two security properties: Linkable Affiliation-Hiding security and Authenticated Key Exchange security (with forward secrecy). Both requirements are defined with regard to multiple input groups per user and session. The following definition is helpful to keep track on executed handshake sessions:

Definition 5 (Session IDs and Partnered Session). *For a* $\mathsf{Handshake}$ *session* π *with* $\pi.\mathsf{state} = \mathsf{accepted}$ *the* session id $\pi.\mathsf{sid}$ *is a value that uniquely identifies* π *in the set of all protocol sessions started by* $\pi.\mathsf{id}$*. Two sessions* π, π' *are called* partnered *if* $\pi.\mathsf{state} = \pi'.\mathsf{state} = \mathsf{accepted}$ *and* $(\pi.\mathsf{sid}, \pi.\mathsf{id}, \pi.\mathsf{partner}) = (\pi'.\mathsf{sid}, \pi'.\mathsf{partner}, \pi'.\mathsf{id})$.

The adversary $\mathcal{A}$ is modeled as a PPT machine that interacts with protocol participants and can mount attacks via the following set of queries.

$\mathsf{Handshake}(\mathsf{id}, \mathcal{G}, r)$. This query lets the holder of pseudonym id start a new session π of the $\mathsf{Handshake}$ protocol. It receives as input a set $\mathcal{G}$ of public group keys $G.\mathsf{pk}$ and a role identifier $r \in \{\mathsf{init}, \mathsf{resp}\}$ that determines whether the session will act as protocol initiator or responder. Session variable $\pi.\mathsf{revealed}$ is initialized to false. If there is a group $G.\mathsf{pk}$ listed in $\mathcal{G}$ for which id has no private credential $\mathsf{cred}_{G.\mathsf{pk}}$ then this query is ignored. Optionally, this query returns a first protocol message M.

$\mathsf{Send}(\pi, M)$. Message M is delivered to session π. After processing M the eventual output is given to $\mathcal{A}$. This query is ignored if π is not waiting for input.

$\mathsf{Reveal}(\pi)$. If $\pi.\mathsf{state} = \mathsf{running}$ then this query is ignored. Otherwise the flag $\pi.\mathsf{revealed}$ is set to true and $(\pi.\mathsf{state}, \pi.\mathsf{key}, \pi.\mathsf{groups})$ is returned.

$\mathsf{Corrupt}(\mathsf{id}, G)$. Membership credential $\mathsf{cred}_{G.\mathsf{pk}}$ of pseudonym id in group G is passed to the adversary. Note that this query models the possibility of selective corruptions.

$\mathsf{Revoke}(G, \mathsf{id})$. This query lets the GA of G include id in its revocation list $G.\mathsf{prl}$.

5.2 Linkable Affiliation-Hiding Security

We start with the property of Linkable Affiliation-Hiding (LAH). At a high level the goal here is to protect the disclosure of non-matching affiliations of handshake participants. We model LAH-security using the indistinguishability approach (similar to that used for encryption schemes). The goal of the adversary is to decide which of the two sets of affiliations $\mathcal{G}_0^*$ or $\mathcal{G}_1^*$ some challenge session π^* is using. The adversary can also invoke any number of handshake sessions, and ask Reveal and Corrupt queries at will. This intuition is formalized as follows.

Definition 6 (LAH-Security). *Let* $\mathsf{MLAH\text{-}AKE} = \{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$*,* b *be a randomly chosen bit, and* $\mathcal{Q} = \{\mathsf{Handshake}, \mathsf{Send}, \mathsf{Reveal}, \mathsf{Corrupt}, \mathsf{Revoke}\}$ *denote the set of queries the adversary* $\mathcal{A}$ *has access to. We consider the following game between a challenger and the adversary* $\mathcal{A}$*:*

$\mathsf{Game}^{\mathsf{lah},b}_{\mathcal{A},\mathsf{MLAH\text{-}AKE}}(\kappa, n, m)$:

– *the challenger creates users* $U_1, \ldots, U_n$ *and pseudonyms* $ID = \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$*;*
– *the challenger creates* m *groups* $\mathcal{G} = \{G_1, \ldots, G_m\}$ *and registers user* U_i *with pseudonym* id_i *in group* G_j *for all* $(i, j) \in [1, n] \times [1, m]$*;*
– $\mathcal{A}^{\mathcal{Q}}$ *interacts with all participants using the queries in* $\mathcal{Q}$*; at some point* $\mathcal{A}^{\mathcal{Q}}$ *outputs a tuple* $(\mathsf{id}^*, \mathcal{G}_0^*, \mathcal{G}_1^*, r^*)$ *where* $\mathsf{id}^* \in ID$*,* $\mathcal{G}_0^*, \mathcal{G}_1^* \subseteq \mathcal{G}$ *with* $|\mathcal{G}_0^*| = |\mathcal{G}_1^*|$*, and* $r^* \in \{\mathsf{init}, \mathsf{resp}\}$*. The set* $\mathcal{D}^* = (\mathcal{G}_0^* \setminus \mathcal{G}_1^*) \cup (\mathcal{G}_1^* \setminus \mathcal{G}_0^*) = (\mathcal{G}_0^* \cup \mathcal{G}_1^*) \setminus (\mathcal{G}_0^* \cap \mathcal{G}_1^*)$ *is called the* distinguishing set*;*
– *the challenger invokes a* $\mathsf{Handshake}(\mathsf{id}^*, \mathcal{G}_b^*, r^*)$ *session* π^* *(and provides all needed credentials);*
– $\mathcal{A}^{\mathcal{Q}}$ *continues interacting via queries (including on session* π^**) until it terminates and outputs bit* b'*;*
– *the output of the game is 1 if all of the following hold; else the output is 0:*
  (a) $b = b'$*,*
  (b) if π^* *accepted and there is a* Handshake *session* π' *with* $\mathcal{D}^* \cap \pi'.\mathcal{G} \neq \emptyset$ *which was in state* running *while* π^* *was in state* running*, then no* $\mathsf{Reveal}(\pi^*)$ *query was asked,*
  (c) no $\mathsf{Reveal}(\pi')$ *query was asked for any* Handshake *session* π' *with* $\mathcal{D}^* \cap \pi'.\mathcal{G} \neq \emptyset$ *and* $\pi'.\mathsf{partner} = \mathsf{id}^*$ *that was in state* running *while* π^* *was in state* running*,*
  (d) no $\mathsf{Corrupt}(\mathsf{id}, G)$ *query with* $(\mathsf{id}, G) \in ID \times \mathcal{D}^*$ *was asked before* π^* *left* running *state.*

We define $\mathsf{Adv}^{\mathsf{lah}}_{\mathcal{A},\mathsf{MLAH\text{-}AKE}}(\kappa, n, m) :=$

$$\left|\Pr[\mathsf{Game}^{\mathsf{lah},0}_{\mathcal{A},\mathsf{MLAH\text{-}AKE}}(\kappa, n, m) = 1] - \Pr[\mathsf{Game}^{\mathsf{lah},1}_{\mathcal{A},\mathsf{MLAH\text{-}AKE}}(\kappa, n, m) = 1]\right|$$

and denote with $\mathsf{Adv}^{\mathsf{lah}}_{\mathsf{MLAH\text{-}AKE}}(\kappa, n, m)$ *the maximum advantage over all PPT adversaries* $\mathcal{A}$*. We say that* MLAH-AKE *is* LAH-secure *if this advantage is negligible in* κ *(for all* n, m *polynomially dependent on* κ*).*

Conditions (b)–(d) exclude some trivial attacks on affiliation hiding. Condition (b) thwarts the attack where $\mathcal{A}$ starts a $\mathsf{Handshake}(\mathsf{id}', \mathcal{G}', r')$ session π' with $\mathcal{G}' \cap \mathcal{D}^* \neq \emptyset$, relays all messages between π^* and π' and finally asks $\mathsf{Reveal}(\pi^*)$. By protocol correctness $\pi^*.\mathsf{groups}$ would contain elements from $\mathcal{D}^*$ and it would be trivial to correctly decide about b. Condition (c) handles the same attack, but from the point of view of π'. Condition (d) prevents $\mathcal{A}$ to corrupt a pseudonym in a group from $\mathcal{D}^*$, to impersonate that user and to decide about bit b from the output of its protocol run.

5.3 Authenticated Key Exchange Security

Authenticated Key Exchange (AKE) security of $\mathsf{MLAH\text{-}AKE}$ schemes is modeled similarly to [13] where the goal of the adversary $\mathcal{A}$ is to distinguish the session key computed by some test session π^* from a random value. $\mathcal{A}$ may invoke any number of handshake sessions, corrupt pseudonyms, and reveal established sessions keys at will as long as it does not obtain the session key computed by π^* in some trivial way. For the formal definition of AKE-security we introduce two further queries Reveal' and Test (the latter with secret parameter $b \in \{0,1\}$) and the new session variable $\pi.\mathsf{tested}$, which is initially set to false.

$\mathsf{Reveal}'(\pi)$**.** This query is like the original Reveal query except that it is ignored if $\pi.\mathsf{tested} = \mathsf{true}$ or $\pi'.\mathsf{tested} = \mathsf{true}$ for any session π' partnered with π.

$\mathsf{Test}(\pi)$**.** If π is fresh (see Definition 7) then $\pi.\mathsf{tested}$ is set to true and a key K is returned, where $K = \pi.\mathsf{key}$ if $b = 1$ and $K \overset{\$}{\leftarrow} \{0,1\}^\kappa$ otherwise. In addition, $\pi.\mathsf{groups}$ is returned. This query may be asked at most once.

Definition 7 (Session Freshness). *A session π invoked in response to a* $\mathsf{Handshake}(\mathsf{id}, \mathcal{G}, r)$ *query is called* fresh *if all of the following hold:*

(a) $\pi.\mathsf{state} = \mathsf{accepted}$ *and* $\pi.\mathsf{revealed} = \mathsf{false}$.
(b) $\pi'.\mathsf{revealed} = \mathsf{false}$ *for any session π' that is partnered with π.*
(c) there exists a group $G \in$ $\pi.\mathsf{groups}$ *such that neither* $\mathsf{Corrupt}(\pi.\mathsf{id}, G)$ *nor* $\mathsf{Corrupt}(\pi.\mathsf{partner}, G)$ *has been asked before* $\pi.\mathsf{state}$ *was set to* $\mathsf{accepted}$.

Conditions (a)–(c) imply the usual constraints of key secrecy models that include forward secrecy [5]. Condition (c) demands that a single group G for which $\pi.\mathsf{id}$ and $\pi.\mathsf{partner}$ remain uncorrupted until protocol acceptance suffices for the tested session to be considered fresh.

Definition 8 (AKE-Security with Forward Secrecy). *Let* $\mathsf{MLAH\text{-}AKE} = \{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$, *$b$ be a randomly chosen bit, and* $\mathcal{Q} = \{\mathsf{Handshake}, \mathsf{Send}, \mathsf{Corrupt}, \mathsf{Revoke}, \mathsf{Reveal}', \mathsf{Test}\}$ *denote the set of queries the adversary $\mathcal{A}$ has access to. We consider the following game between a challenger and the adversary $\mathcal{A}$:*

$\mathsf{Game}^{\mathsf{ake},b}_{\mathcal{A},\mathsf{MLAH\text{-}AKE}}(\kappa, n, m)$:

- *the challenger creates users $U_1, \ldots, U_n$ and pseudonyms* $\mathsf{id}_1, \ldots, \mathsf{id}_n$*;*
- *the challenger creates m groups $G_1, \ldots, G_m$ and registers U_i with pseudonym* id_i *in group G_j for all $(i,j) \in [1,n] \times [1,m]$;*
- *$\mathcal{A}^{\mathcal{Q}}$ interacts with all participants using the queries in $\mathcal{Q}$;*
- *at some point $\mathcal{A}^{\mathcal{Q}}$ asks* $\mathsf{Test}(\pi^*)$ *to a fresh session π^*;*
- *$\mathcal{A}^{\mathcal{Q}}$ continues interacting via queries until it terminates and outputs bit b', which is the output of the game.*

We define $\mathsf{Adv}^{\mathsf{ake}}_{\mathcal{A},\mathsf{MLAH\text{-}AKE}}(\kappa, n, m) := \left| 2\Pr[\mathsf{Game}^{\mathsf{ake},b}_{\mathcal{A},\mathsf{MLAH\text{-}AKE}}(\kappa, n, m) = b] - 1 \right|$ *and denote with* $\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{MLAH\text{-}AKE}}(\kappa, n, m)$ *the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that* $\mathsf{MLAH\text{-}AKE}$ *is* AKE-secure *if this advantage is negligible in κ (for all n, m polynomially dependent on κ).*

6 Security Analysis of Our Protocol

Following the extended definitions from the previous sections we prove that our MLAH-AKE protocol from Section 4 satisfies the desired goals. The respective proofs are provided in the full version of this paper.

Theorem 2 (Linkable Affiliation-Hiding Security). *The* MLAH-AKE *protocol from Section 4 is LAH-secure under the RSA assumption on safe moduli in the random oracle model.*

Theorem 3 (Authenticated Key Exchange Security). *The* MLAH-AKE *protocol from Section 4 is AKE-secure under the RSA assumption on safe moduli in the random oracle model.*

The dependence on the RSA assumption and the random oracle model stems from the underlying LAH-AKE protocol [13] (which is proven secure under these assumptions). Note that our IHME approach can be deployed independently of any number-theoretical or non-standard assumption.

7 Conclusion

We discussed several solutions to the open problem of *efficient group discovery* in AH-AKE protocols. We stress that without efficient group discovery, existing AH-AKE schemes (that provide support for only one input group per user and protocol session) would remain of very limited use. Throughout the paper we described how to perform group discovery more efficiently than by the naïve combinatorial approach, for which the computational and communication complexity is $O(n^2)$ (where n is the number of input groups per participant). Our most efficient solution came from the use of a new primitive, called *Index-Hiding Message-Encoding (IHME)*. In addition to the definition of IHME and its *index-hiding* property we gave a construction for which the property holds unconditionally. We then demonstrated how IHME can be applied to the state-of-the-art linkable AH-AKE protocol from [13] in order to discover groups in *linear complexity* $O(n)$. Our construction is supported by appropriate definitions of security and proofs.

References

1. Ateniese, G., Kirsch, J., Blanton, M.: Secret Handshakes with Dynamic and Fuzzy Matching. In: Network and Distributed System Security Symposium (NDSS 2007). The Internet Society (2007)
2. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.-C.: Secret Handshakes from Pairing-Based Key Agreements. In: IEEE Symposium on Security and Privacy 2003, pp. 180–196. IEEE CS, Los Alamitos (2003)
3. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: 1st ACM Conference on Computer and Communications Security (CCS 1993), pp. 62–73. ACM, New York (1993)
4. Camenisch, J., Zaverucha, G.M.: Private Intersection of Certified Sets. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 108–127. Springer, Heidelberg (2009)

5. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
6. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret Handshakes from CA-Oblivious Encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
7. Cristofaro, E., Jarecki, S., Kim, J., Tsudik, G.: Privacy-Preserving Policy-Based Information Transfer. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 164–184. Springer, Heidelberg (2009)
8. Cristofaro, E., Tsudik, G.: Practical Private Set Intersection Protocols with Linear Computational and Bandwidth Complexity. Cryptology ePrint Archive, Report 2009/491. To appear in Financial Cryptography and Data Security. LNCS, vol. 6052. Springer, Heidelberg (2010)
9. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
10. 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)
11. Hazay, C., Nissim, K.: Efficient Set Operations in the Presence of Malicious Adversaries. Cryptology ePrint Archive, Report 2009/594. In: Nguyen, P.Q., Pointcheval, D. (Eds.) PKC 2010. LNCS, vol. 6056, pp. 312–331. Springer, Heidelberg (2010)
12. Jarecki, S., Kim, J., Tsudik, G.: Group Secret Handshakes or Affiliation-Hiding Authenticated Group Key Agreement. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 287–308. Springer, Heidelberg (2006)
13. Jarecki, S., Kim, J., Tsudik, G.: Beyond Secret Handshakes: Affiliation-Hiding Authenticated Key Exchange. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 352–369. Springer, Heidelberg (2008)
14. Jarecki, S., Liu, X.: Unlinkable Secret Handshakes and Key-Private Group Key Management Schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 270–287. Springer, Heidelberg (2007)
15. 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)
16. Jarecki, S., Liu, X.: Private Mutual Authentication and Conditional Oblivious Transfer. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. LNCS, vol. 5677, pp. 90–107. Springer, Heidelberg (2009)
17. Kissner, L., Song, D.X.: Privacy-Preserving Set Operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
18. Raab, M., Steger, A.: Balls Into Bins — A Simple and Tight Analysis. In: Rolim, J.D.P., Serna, M., Luby, M. (eds.) RANDOM 1998. LNCS, vol. 1518, pp. 159–170. Springer, Heidelberg (1998)
19. Tsudik, G., Xu, S.: A Flexible Framework for Secret Handshakes. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 295–315. Springer, Heidelberg (2006)
20. Vergnaud, D.: RSA-Based Secret Handshakes. In: Ytrehus, Ø. (ed.) WCC 2005. LNCS, vol. 3969, pp. 252–274. Springer, Heidelberg (2006)
21. Xu, S., Yung, M.: k-Anonymous Secret Handshakes with Reusable Credentials. In: 11th ACM Conference on Computer and Communications Security (CCS 2004), pp. 158–167. ACM, New York (2004)

Two New Efficient PIR-Writing Protocols

Helger Lipmaa[1,2] and Bingsheng Zhang[1,3]

[1] Cybernetica AS, Estonia
[2] Tallinn University, Estonia
[3] University of Tartu, Estonia

Abstract. Assume that a client outsources his database to a remote storage-provider (the server), so that for privacy reasons, the client's database is encrypted by his secret key. During a PIR-writing protocol, the client updates one element of the encrypted database without revealing to the semi-honest server which element was updated and, of course, to which value. The best previous PIR-writing protocols had square-root communication complexity. In this paper, we propose two new PIR-writing protocols. The first one can be based on (say) the Damgård-Jurik additively homomorphic public-key cryptosystem, and it has (amortized) polylogarithmic communication for a limited number of updates. The second one is based on a fully-homomorphic public-key cryptosystem, a much stronger primitive, but it achieves optimal logarithmic communication.

Keywords: Cryptocomputing, binary decision diagram, circuits, fully-homomorphic encryption, PIR-writing, PrivateBDD.

1 Introduction

With the progress of network facilities, an increasing number of services are based on remote storage (also known as online disks), such as Google Doc, virtual OS and many other other web applications. Meanwhile, the clients of such services do not always trust the storage provider to keep their privacy. Therefore, any client would like to only outsource an encrypted database that only he can decrypt. On the other hand, most applications require that the storage provider should allow clients to add, retrieve, modify and delete documents of their encrypted databases. In particular, in a PIR-writing protocol (also known as private database modification, [2]), the client updates one element of the encrypted database so that the semi-honest server does not get to know which element was updated and to which value.

More precisely, assume that the unencrypted database is $f = (f_0, \ldots, f_{n-1})$ of ℓ-bit elements, the client's private index is x and the private update value is y. After executing a PIR-writing protocol, the server updates the x-th element of the client's database to y. Since server's part in the PIR-writing protocol can be executed without knowing the secret key, it also possibly allows a third party to (homomorphically) modify the client's database without compromising the client's privacy, and this property makes it possible to combine PIR-writing protocols with other client-server protocols.

In a trivial PIR-writing protocol with $\Theta(n)$ communication, the server transfers the encrypted database to the client, who replaces the x-th element by an encryption of y, and then re-encrypts the database and sends the new database back. The first non-trivial solution to this problem was recently proposed in [2]. Their protocol has communication complexity $O(\sqrt{n})$ when modifying 1 bit of the database. By repeating the protocol, one has a PIR-writing protocol with the communication complexity $O(\ell\sqrt{n})$ for modifying ℓ bits.

Another—and strongly related—protocol was proposed in a yet unpublished eprint [3]. They propose a pair of amortized protocols with communication complexity $O(\sqrt{\ell^{1+\alpha} \cdot n} \cdot \mathrm{polylog}(n))$ (where $\alpha \in (0,1)$ is a constant) for modifying ℓ bits of the database. One protocol achieves this amortized communication complexity for arbitrary bit modifications, while the second one achieves this amortized communication complexity only when flipping a 0 bit into a 1 bit. Both square-root communication solutions use the BGN cryptosystem [1] as the underlying cryptographic primitive, so their security is based on the hardness of the Subgroup Decision assumption [1]. A drawback of the protocols of [2,3] is that they require the client to precompute the increment between the new value and the original one of the xth element, by say using an additional communication-efficient CPIR protocol to first retrieve the current value of this element. This adds complexity to their protocols, and in particular the PIR-writing protocols of [2,3] consist of more than one message. In this paper, we propose two new nontrivial PIR-writing protocols.

First New, PrivateBDD-Based PIR-Writing Protocol. The first new PIR-writing protocol is based on the cryptocomputing protocol PrivateBDD of Ishai and Paskin [9], or more precisely on an efficient variation of it as described in [11]. As described in [11], the PrivateBDD cryptocomputing protocol is based on an efficient binary decision diagram (BDD) together with an efficient $(2,1)$-CPIR protocol. See Sect. 2.1 for background about the BDDs (a very well known computational model) and the PrivateBDD protocol in general.

In the first new PIR-writing protocol, the server constructs a (multi-terminal) binary decision diagram (BDD, also known us branching program, [15]) for the function

$$g_i(x, y, m) := \begin{cases} y \ , & x = i \ , \\ m \ , & x \neq i \ . \end{cases}$$

The corresponding BDD has size and length $\lceil \log_2 n \rceil$. In the nonprivate version of the PIR-writing protocol, the server just sets in parallel $f_i \leftarrow g_i(x, y, f_i)$ for every $i \in \{0, \ldots, n-1\}$. In the actual new PIR-writing protocol, the server uses the PrivateBDD protocol to update the encrypted version of f_i, given encryptions of x, y and a (multiple-)encryption of the old version of f_i.

Now, the output of the PrivateBDD protocol is a multiple-encryption of the actual value of the protocol, where the number of multiples is equal to the length of the BDD, that is, to $\lceil \log_2 n \rceil$ in this concrete case. Therefore, after every update, the length of every (multiple-encrypted) database element kept by the server increases by $\lceil \log_2 n \rceil \cdot \kappa$ bits, where κ is the security parameter. More precisely, for some upper bound u on the number of updates, this PIR-writing protocol achieves *amortized* communication complexity $\Theta(u\ell \cdot \log n + u^2\kappa \cdot \log^2 n)$. This improves upon the protocol of [3] for $u = o(\sqrt[4]{\ell^{1+\alpha} \cdot n} \cdot \text{polylog}(n))$. In addition, since the working time of the PrivateBDD is proportional to the size of the BDD, the working time of this new PIR-writing protocol is $\Theta(n \cdot \log n)$ public-key operations, where the cost of public-key operations depends on ℓ and increases after every update.

We also construct a slight modification of this protocol, where the server does not update the database elements but instead just stores all the client's write requests. This makes the server's computational complexity during the PIR-writing protocol very small. However, when the client requests to read some element of the database, the server executes all write requests on the original database. Therefore, this modification is useful in applications where updating is much more frequent than reading.

Importantly, this protocol does *not* use pairings. In particular, it relies on the classical DCR assumption which is by far the weakest security assumption under which a nontrivial PIR-writing protocol is known. Clearly, even in the trivial PIR-writing protocol, the database has to be encrypted by using a CPA-secure public-key cryptosystem. It is conceivable that this cryptosystem must be additively homomorphic to allow some simple cryptocomputing protocols on encrypted database. In such a case, the PrivateBDD-based PIR-writing protocol comes "free" in the sense of security, adding no extra security assumptions.

Second New, FH-Based PIR-Writing Protocol. The second new PIR-writing protocol is based on the (leveled) fully-homomorphic public-key cryptosystem proposed recently by Gentry [7]. (Equivalently, one could also use the cryptosystem from [5]) The idea behind this protocol is similar to the first new PIR-writing protocol. This time we write down a Boolean circuit for $g_i(x, y, f_i)$, which happens to have exactly the same size as the corresponding BDD but has depth $\approx \log_2 \log_2 n$. We now use the fully-homomorphic cryptosystem to evaluate in parallel n copies of this circuit for $i \in \{0, \dots, n-1\}$. Importantly, (1) this circuit only has multiplicative depth $\log \log n$ (as compared to the length $\log n$ of the corresponding BDD), and (2) this process does *not* increase the size of the server's database. The actual FH-based PIR-writing protocol has communication complexity $O((\log n + \ell) \cdot \kappa)$, and computation complexity of $O(n \cdot \log n + \ell n)$ evaluations of the fully-homomorphic cryptosystem.

However, since every multiplication (or Boolean AND) increases dramatically the noise used in encryption, some care has to be taken to bootstrap the stored database values, see [7]. Since the multiplicative depth of n parallel applications of the circuit for g is $\lceil \log_2 \lceil \log_2 n \rceil \rceil$ (which is never larger than 5 in practice), we can just assume that the security parameter is big enough to accommodate the evaluation of circuits of multiplicative depth $\Theta(\log \log n)$. However, a bootstrapping should be done at the end of the PIR-writing protocol. See [7,6] for more discussion.

Table 1. Comparison of previous PIR-writing protocols and our two new protocols. In the case of the protocol from Sect. 3 (and Sect. 3), u is the number of updates, and we give amortized communication over the first u updates. Note that the meaning of the unit computation depends on the protocol.

Scheme	Communication Complexity	Computation Complexity	Security Assumption	Increase Database Size
Trivial	$n(\ell + \kappa)$	$O(\ell n)$	CPA-security of underlying cryptosystem	None
[2]	$O(\ell\sqrt{n})$	$O(\ell n)$	subgroup decision [1] + polylog-communication CPIR [8,10]	None
[3]	$\sqrt{\ell^{1+\alpha} \cdot n} \cdot \text{polylog}(n)$	$O(n \cdot \text{polylog}(n))$	subgroup decision [1] + polylog-communication CPIR [8,10]	None
Sect. 3	$\Theta(u\ell \cdot \log n + u^2\kappa \cdot \log^2 n)$	$O(n \cdot \log n)$	DCR assumption	Increased by $\kappa \log n$
Sect. 3.1	$\Theta(u\ell \cdot \log n + u^2\kappa \cdot \log^2 n)$	Trivial	DCR assumption	None
Sect. 4	$O(\kappa \log n + \kappa\ell)$	$O(n \cdot \log n + \ell n)$	CPA-security of fully-homomorphic cryptosystem	None

In particular, if we use *leveled* fully-homomorphic cryptosystem, then the length of the public key is linear in U, where U is the maximal number of imagined updates. Since this basically means that one element of the public key has to be transfered to the server per every update, it does not increase the amortized communication cost significantly. Moreover, if we assume that the fully-homomorphic cryptosystem is secure against key-dependent message (KDM) attacks, then it is fully-homomorphic (and not leveled fully-homomorphic) and we can just use a constant-in-U-size public key. Since the security of the cryptosystems from [7,5] has not been well-studied with or without KDM attacks, we will not mention the cost of maintaining and transferring the public key anymore.

Comparison. In Table 1, we compare the trivial solution, two previously known sublinear-communication protocols, and the two new protocols. Note that in the protocol from Sect. 3, the cost of a single public-key operation depends on ℓ, and on the number of the update.

2 Preliminaries

Notation. Within this paper, κ is always the security parameter. The client has outsourced a database $f = (f_0, \ldots, f_{n-1})$ to the server. Client's private inputs are an $m = \lceil \log_2 n \rceil$-bit index $x = (x_0, \ldots, x_{m-1})$ and an ℓ-bit value $y = (y_0, \ldots, y_{\ell-1})$. All logarithms are taken on basis 2. For a set(or possibly a probabilistic algorithm) S, $x \leftarrow S$ means a uniformly random assignment of an element from S to x. Linearity and polylogarithmicity is usually measured with respect to n, while in security proofs polynomiality and negligibility is measured with respect to $\ll$.

2.1 PrivateBDD Protocol

Additively-Homomorphic Cryptosystems. Let $\mathsf{P} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$ be a length-flexible additively-homomorphic public-key cryptosystem [4], where G is a randomized key generation algorithm, E is a randomized encryption algorithm and D is a decryption algorithm. Here, both E and D receive an additional length parameter ℓ, so that $\mathsf{E}_{\mathsf{pk}}(\ell, \cdot)$ encrypts plaintexts from some set $\{0,1\}^{\leq \ell}$. In the case of the DJ01 cryptosystem from [4], for every integer $\ell > 0$, $\mathsf{E}_{\mathsf{pk}}(\ell, \cdot)$ is a valid plaintext of $\mathsf{E}_{\mathsf{pk}}(\lceil \ell/\kappa \rceil \cdot \kappa + \kappa, \cdot)$, and therefore one can multiple-encrypt messages as say in

$$C \leftarrow \mathsf{E}_{\mathsf{pk}}(\ell + 2\kappa, \mathsf{E}_{\mathsf{pk}}(\ell + \kappa, \mathsf{E}_{\mathsf{pk}}(\ell, M))) \ ,$$

and then recover M by multiple-decrypting,

$$M \leftarrow \mathsf{D}_{\mathsf{sk}}(\ell + 2\kappa, \mathsf{D}_{\mathsf{sk}}(\ell + \kappa, \mathsf{D}_{\mathsf{sk}}(\ell, C))) \ .$$

If N is the public key of the DJ01 cryptosystem. then $2^\ell < N$. Additionally, in any length-flexible additively-homomorphic cryptosystem, $\mathsf{E}_{\mathsf{pk}}(\ell, M_1) \cdot \mathsf{E}_{\mathsf{pk}}(\ell, M_2) = \mathsf{E}_{\mathsf{pk}}(\ell, M_1 + M_2)$, where the addition is modulo the public key N. We will explicitly need the existence of a compression function C that, given pk, ℓ' and ℓ for $\ell' \geq \ell$, and $\mathsf{E}_{\mathsf{pk}}(\ell', M)$ for $M \in \{0,1\}^\ell$, returns $\mathsf{E}_{\mathsf{pk}}(\ell, M) \in \{0,1\}^{\lceil \ell/\kappa \rceil \cdot \kappa + \kappa}$.

In the CPA (*chosen-plaintext attack*) game, the challenger first generates a random $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{G}(1^\kappa)$, and sends pk to the attacker. Attacker chooses two messages (M_0, M_1) (such that $|M_0| = |M_1|$) and a length parameter ℓ, and sends them to the challenger. Challenger picks a random bit b, and sends a ciphertext $\mathsf{E}_{\mathsf{pk}}(\ell, M_b)$ to attacker. Attacker outputs a bit b', and wins if $b = b'$.

In the LFCPA (*length-flexible chosen-plaintext attack*) game [10], the challenger first generates a random $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{G}(1^\kappa)$, and sends pk to the attacker. Attacker chooses a polynomial number of message pairs (M_{j0}, M_{j1}) (such that $|M_{j0}| = |M_{j1}|$) and length parameters ℓ_j, and sends them to the challenger. Challenger picks a random bit b, and sends all ciphertexts $\mathsf{E}_{\mathsf{pk}}(\ell_j, M_{jb})$ to attacker. Attacker outputs a bit b', and wins if $b = b'$. Because of the existence of the compress function, LFCPA security follows from the CPA security [11]. Thus, the DJ01 cryptosystem [4] is LFCPA-secure under the Decisional Composite Residuosity Assumption.

Cryptocomputing. Let m and ℓ be public parameters, and let $\mathcal{F}$ a class of functions $\{0,1\}^m \rightarrow \{0,1\}^\ell$. In a cryptocomputing protocol for $\mathcal{F}$ between a client and a server, the client has an input $x \in \{0,1\}^m$ and the server has an input $f \in \mathcal{F}$. The client obtains $f(x)$. Every cryptocomputing protocol $\Gamma = (\mathsf{Q}, \mathsf{R}, \mathsf{A})$

has two messages, where the client sends $\mathsf{Q}(\ell, x)$ to the server, the server replies with $\mathsf{R} \leftarrow \mathsf{R}(\ell, f, \mathsf{Q})$, and then finally the stateful client recovers f_x by computing $\mathsf{A}(\ell, x, \mathsf{R})$. Here, Q, R and A are (probabilistic) polynomial-time algorithms.

Client-Privacy of Cryptocomputing Protocols. Let $\Gamma = (\mathsf{Q}, \mathsf{R}, \mathsf{A})$ be a 2-message cryptocomputing protocol. Within this work we use the convention of many previous papers to only require (semisimulatable) privacy in the malicious model. In particular, client's privacy is guaranteed in the sense of indistinguishability (CPA-security), That is, for the the privacy *of the client*, no malicious nonuniform probabilistic polynomial-time server should be able to distinguish, with non-negligible probability, between the distributions $\mathsf{Q}(\ell, x_0)$ and $\mathsf{Q}(\ell, x_1)$ that correspond to any two of client's inputs x_0 and x_1 that are chosen by herself. Within this paper, we are not interested in server-privacy.

Computationally-Private Information Retrieval. A two-message 1-out-of-n computationally-private information retrieval protocol, $(n, 1)$-CPIR, is a special type of cryptocomputing protocol. In a $(n, 1)$-CPIR protocol for ℓ-bit strings, the client has an index $x \in \{0, \dots, n-1\}$ and the server has a database $f = (f_0, \dots, f_{n-1})$ with $f_i \in \{0,1\}^\ell$. The client obtains f_x. An $(n, 1)$-CPIR protocol $\Gamma = (\mathsf{Q}, \mathsf{R}, \mathsf{A}, \mathsf{C})$ is *BDD-friendly* if it satisfies the next four assumptions:

1. Γ has two messages, a query $\mathsf{Q}(\ell, x)$ from the client and a reply $\mathsf{R}(\ell, f, \mathsf{Q})$ from the server, such that the stateful client can recover f_x by computing $\mathsf{A}(\ell, x, \mathsf{R}(\ell, f, \mathsf{Q}))$. Note that $\mathsf{A}(\ell, x, \mathsf{R}(\ell, f, \mathsf{Q}(\ell, x))) = f_x$.
2. Γ is uniform in ℓ, that is, it can be easily modified to work on other values of ℓ.
3. $|\mathsf{Q}(\ell, \cdot)|, |\mathsf{R}(\ell, \cdot, \cdot)| \leq \ell + \Theta(\kappa)$ (with possibly $\mathsf{Q}(\ell, \cdot)$ being even shorter).
4. The compress function C maps $\mathsf{Q}(\ell', x)$ to $\mathsf{Q}(\ell, x)$ for any $\ell' \geq \ell$ and any x.

Here Q, R, A and C are (probabilistic) polynomial-time algorithms. The only known BDD-friendly $(2, 1)$-CPIR was proposed by Lipmaa in [10], see [11] for a compact description. Importantly for us, in Lipmaa's $(2, 1)$-CPIR protocol, $\mathsf{Q}(\ell, x)$ consists of a public key and an additively homomorphic encryption of x under this key.

Any $(n, 1)$-CPIR protocol Γ must be client-private, that is, CPA-secure. Lipmaa's $(2, 1)$-CPIR protocol [10], when based on the DJ01 cryptosystem [4], is CPA-secure and thus LFCPA-secure (which is defined in the same way as LFCPA-security for public-key cryptosystems) under the Decisional Composite Residuosity Assumption.

Binary Decision Diagrams. A *binary decision diagram* (BDD, also known as a branching program, [15]) is a fanout-2 directed acyclic graph $(\mathcal{V}, \mathcal{E})$, where the non-terminal nodes are labeled by variables from some set $\{x_0, \dots, x_{m-1}\}$, the sinks are labeled by ℓ-bit strings and the two outgoing edges of every internal node are respectively labeled by 0 and 1. If $\ell > 1$, then the BDD is called *multi-terminal*. A BDD computes some function $f : \{0,1\}^m \rightarrow \{0,1\}^\ell$. Every assignment of the variables selects one path from the source to some sink as follows. The path starts from the source. If the current version of path does not end at a sink, test the variable at the endpoint of the path. Select one of the outgoing edges depending on the value of this variable, and append this edge and its endpoint to the path. If the path ends at a sink, return the label of this sink as the value of the corresponding source. The BDD's value is then equal to the source value. For a BDD P, let $\mathsf{len}(P)$ be its length (that is, the length of its longest path), and let $\mathsf{size}(P)$ be its size (that is, the number of non-terminal nodes).

PrivateBDD Protocol. In [9], Ishai and Paskin proposed a new cryptocomputing method (PrivateBDD) that uses a BDD-representation of the target function in conjunction with a communication-efficient strong oblivious transfer. In [11], the authors noted that the strong oblivious transfer protocol can be replaced by a BDD-friendly $(2,1)$-CPIR protocol. We now briefly recall the main properties of PrivateBDD, as instantiated by Lipmaa's $(2,1)$-CPIR from [10]. See [11] for the full details of the PrivateBDD protocol.

Theorem 1. *Assume that the Decisional Composite Residuosity Assumption is true. Let $\mathcal{F}$ be a set of functions $f : \{0,1\}^m \to \{0,1\}^\ell$, and for any $f \in \mathcal{F}$ let P_f be some (multi-terminal) BDD with ℓ-bit sink labels that computes f. Let $\mathsf{len}(\mathcal{F}) := \max_{f\in\mathcal{F}} \mathsf{len}(f)$. Then $\mathcal{F}$ has a CPA-secure cryptocomputing protocol with communication upperbounded by $\kappa + m \cdot (\ell + (\mathsf{len}(\mathcal{F}) + 2) \cdot \kappa)$, and server's online computation dominated by $\mathsf{size}(f)$ public-key operations.*

Briefly, client's inputs to the PrivateBDD (when instantiated by Lipmaa's $(2,1)$-CPIR from [10]) are encrypted bitwise by using a length-flexible additively homomorphic public-key cryptosystem like DJ01 [4]. Moreover, let V be any internal node of the BDD such that the longest path between V and any sink has length $\mathsf{len}(V) > 0$. Let V_0 and V_1 be the successors of V by the 0-edge and 1-edge, correspondingly. Then V's value $val[V]$ as recursively computed by the PrivateBDD protocol is

$$\mathsf{R}(\ell + (\mathsf{len}(V) - 1)\kappa, \mathsf{Q}(\ell + (\mathsf{len}(V) - 1)\kappa, x_j), (val[V_0], val[V_1])) \ ,$$

where x_j is V's label, and $val[V_i]$ is the already known value of the node V_i. Moreover, sink values are equal to their labels. Therefore, $val[V]$ is equal to an encryption of $val[V_{x_j}]$. Inductively, $val[V]$ is equal to an $\mathsf{len}(V)$-times encryption of some sink value, and $|val[V]| \approx (\mathsf{len}(V) + 1)\kappa$. In particular, server's message in the PrivateBDD protocol is equal to a $\mathsf{len}(P_f)$-times encryption of some sink value, and this sink value by itself is the output of the PrivateBDD protocol. See [11] for more details.

2.2 Fully-Homomorphic Cryptosystem

In this subsection, we give a brief description of the recent fully-homomorphic cryptosystem by Gentry [7]. We omit all the details of Gentry's cryptosystem that are not relevant to the current paper. In particular, all given details are also true for the more recent cryptosystem of van Dijk, Gentry, Halevi and Vaikuntanathan [5].

Let $\mathsf{P} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$ be the initial version of Gentry's cryptosystem [7]. The length ℓ of plaintext space $\mathcal{P}$ is fixed. Similarly to the previous subsection, G is a randomized key generation algorithm such that $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{G}(1^\kappa)$; E is randomized encryption algorithm such that for any $M \in \mathcal{P}$, $C \leftarrow \mathsf{E}_{\mathsf{pk}}(M)$; D is decryption algorithm such that for any $C \in \mathcal{C}$, where $\mathcal{C}$ is ciphertext space, $M \leftarrow \mathsf{D}_{\mathsf{sk}}(C)$.

While encrypting, Gentry's cryptosystem masks the plaintext in particular with an additive noise R from some "small" set $\mathcal{R}_0$, and correct decrypting is guaranteed when the noise belongs to some "large" set $\mathcal{R}_1$, $\mathcal{R}_0 \subset \mathcal{R}_1$. Since the noise is additive, one has

$$\mathsf{E}_{\mathsf{pk}}(M_1; R_1) + \mathsf{E}_{\mathsf{pk}}(M_2; R_2) = \mathsf{E}_{\mathsf{pk}}(M_1 + M_2; R_1 + R_2)$$

and

$$\mathsf{E}_{\mathsf{pk}}(M_1; R_1) \cdot \mathsf{E}_{\mathsf{pk}}(M_2; R_2) = \mathsf{E}_{\mathsf{pk}}(M_1 \cdot M_2; R_1 \cdot R_2) \enspace .$$

Here, the addition of plaintexts is modular, while the noise increases unboundedly. Thus, one can evaluate arbitrary $+/\cdot$ (or alternatively, $\oplus/\wedge$) circuits of only some bounded length before the noise has increased to the level where correct decryption is not anymore possible. Therefore, this simple version of Gentry's cryptosystem is only somewhat homomorphic [7], that is, homomorphic for small-depth circuits.

However, as shown in [7], the somewhat homomorphic version of Gentry's cryptosystem is sufficient to homomorphically evaluate its own decryption circuit augmented with basic Boolean operations. Hence, one can strengthen the somewhat homomorphic version with a bootstrapping step. Assume that the plaintexts have been encrypted by using some public key pk_1. Now, just before the circuit depth has reached the level where decryption becomes incorrect, one encrypts the ciphertexts $\mathsf{E}_{\mathsf{pk}_1}(\cdot)$ by using a different public key pk_2, and then homomorphically decrypts the results, obtaining new encryptions of the same plaintexts but under the new key pk_2 and with decreased noise. This step is called *bootstrapping*. After that, one can homomorphically execute another few levels of the circuit, until one needs to bootstrap again.

Thus, given the somewhat homomorphic cryptosystem, Gentry constructed a leveled fully-homomorphic cryptosystem that enables one to homomorphically evaluate circuits of any *bounded* and *a priori fixed* depth d. In such a *leveled fully-homomorphic cryptosystem*, the public-key size is linear in d, and in particularly it includes encryptions of all bits of sk_i under the public key pk_{i+1}, for linear-in-d needed public/secret key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$.

To overcome this restriction, one can assume that Gentry's cryptosystem is secure against key-dependent message (KDM) attacks. In this case, one can, given a *single* valid key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{G}(1^\kappa)$, circularly encrypt the bits of sk by using pk. Therefore, after every few levels of the circuit, one can use the same public key pk to bootstrap the circuit. In particularly, Gentry showed [7,6] that his leveled fully-homomorphic cryptosystem is fully-homomorphic under the random oracle assumption. Therefore, within this paper, we will assume that Gentry's cryptosystem is fully-homomorphic (and not leveled fully-homomorphic).

Finally, since we only need to encrypt Boolean plaintexts 0 and 1, other details of Gentry's cryptosystem—for example, the fact that it is lattice-based—are not important for our purposes. We refer an interested reader to [7,6] for many further details. In particular, we will only assume that Gentry's fully-homomorphic cryptosystem is CPA-secure (and KDM-secure). The underlying assumptions that are needed for CPA-security (and KDM-security) can again be found from [7].

2.3 PIR-Writing

Assume that the client has outsourced his database to the server. To protect his privacy, the database is encrypted by using client's public key. In a PIR-writing protocol (also known as a private database modification protocol, [2]), the client updates a single element of the database so that the server does not know which element was changed. More precisely, the database $f = (f_0, \dots, f_{n-1})$ has n elements $f_i \in \{0,1\}^\ell$. The client has private inputs (sk, x, y), where sk is his secret key, $x \in \{0, \dots, n-1\}$ is the element to be changed, and $y \in \{0,1\}^\ell$ is its new version. The server has an encrypted version $(c_0, \dots, c_{n-1}) = (\mathsf{E}_{\mathsf{pk}}(f_0), \dots, \mathsf{E}_{\mathsf{pk}}(f_{n-1}))$ of the database and a copy of client's public key pk. Ideally, the protocol consists of only a single message, from the client to the server. The client has no private output, while server obtains a new encrypted database $c' = (c'_0, \dots, c'_{n-1})$, such that c'_i and c_i decrypt to the same value if $i \neq x$, while c'_x decrypts to y.

The privacy definition of a PIR-writing protocol is formalized by the following PIR-writing game. Let A be a semi-honest probabilistic-polynomial time adversary (that is, the server), and let C be the challenger. The game consists of the following steps:

1. The challenger C generates a pair of keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{G}(1^\kappa)$, and sends pk to A.
2. A picks and sends to C a database $f = (f_0, \dots, f_{n-1})$ of n elements with length ℓ.
3. C encrypts the database with pk and sends it back to A.
4. (Challenge phase:) A picks and sends to C two index and value pairs $(x_0^*, y_0^*), (x_1^*, y_1^*)$. C picks $b_c \leftarrow \{0,1\}$, and executes the PIR-writing protocol with input (x_b^*, y_b^*), with A playing the role of the server.
5. A outputs her guess $b_c^* \in \{0,1\}$ for b_c.

Note that here we do not have a query phase, since in our one-message protocols the malicious server can just play the PIR-writing part with herself. (Recall that the database she has is encrypted by using client's public key.) The situation is different in say [2] where the client of the PIR-writing protocol had to known the current value of the modified database element.

Definition 1 (Client-privacy of PIR-writing). *Let the adversary's advantage in the previous game be*

$$\mathsf{Adv}_A(1^\kappa) := \left|\Pr[b_c^* = b_c] - \frac{1}{2}\right| \enspace .$$

We say that a PIR-writing protocol is client-private*, if for all probabilistic-polynomial time adversaries A,* $\mathsf{Adv}_A(1^\kappa)$ *is a negligible function (in κ).*

Previous PIR-Writing Protocols. In a trivial (two-message) linear-communication protocol, the server sends the encrypted database back to the client, who updates the xth element, re-encrypts other elements, and sends the new encrypted database back to the server. Another linear-communication PIR-writing protocol can be based on an arbitrary additively homomorphic public-key cryptosystem as follows. The client and the server first execute an $(n, 1)$-CPIR protocol, so that the client obtains the current value of f_x. Then client forwards to the server n ciphertexts c_j, where c_x decrypts to $y - f_x$ and other c_j-s decrypt to 0. The server multiplies the encryptions of f_j with the ciphertexts c_j, this clearly correctly updates the database.

The first sublinear-communication PIR-writing protocol was proposed in [2]. Essentially, it uses the bilinear-pairing based cryptosystem of [1] to send $2 \cdot \sqrt{n}$ ciphertexts c'_j and c''_j—such that the decryption of c_j is equal to the product of decryptions of c'_j and c''_j—instead of n ciphertexts as in the previous protocol. Thus, this protocol has communication complexity $O(\sqrt{n})$ to modify one bit of a database. Clearly, by repeating the protocol, one will have a PIR-writing protocol with communication complexity $O(\ell \cdot \sqrt{n})$ for modifying ℓ bits.

A way to decrease the communication complexity for larger ℓ was proposed in an unpublished eprint [3]. The solution consists of a pair of amortized protocols that have communication complexity $O(\sqrt{\ell^{1+\alpha} \cdot n} \cdot \text{polylog}(n))$ (where $\alpha \in (0, 1)$ is a constant) for modifying ℓ bits of the database. The idea is to encode an n-bit database as a Dn-bit "virtual database" shared on M different servers by using unbalanced lossless expander graphs. One of their protocol achieves the claimed amortized communication complexity for any arbitrary bit modifications, while the other one achieves the same amortized communication complexity only when flipping a 0 bit into a 1 bit. All protocols from [2,3] use the BGN cryptosystem [1] as cryptographic primitive, so their security is based on the hardness of subgroup decision problem [1].

A drawback of the protocols from [2,3] is that the client has to know the current value of f_x before applying their protocols. Thus, the client has to use a communication-efficient CPIR protocol [10,8] first to retrieve f_x. This increases both the computational and communication complexity. Most importantly, this means that the protocols of [2,3] have more than one message.

Finally, in [13], the authors studied the same problem in a different, information-theoretic setting with multiple databases.

3 New PrivateBDD-Based PIR-Writing Protocol

The first new PIR-writing protocol is based on the cryptocomputing protocol PrivateBDD of Ishai and Paskin [9]. Thus, it is based on an $(2, 1)$-CPIR protocol of Lipmaa [10] and in particular on a length-flexible additively-homomorphic cryptosystem $\mathsf{P} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$. Briefly, in this protocol the server constructs a binary decision diagram (BDD) $P_i(x, y, m)$ for the function

$$g_i(x, y, y') := \begin{cases} y \ , & x = i \ , \\ y' \ , & x \neq i \ . \end{cases}$$

(Here, we assume that n is implicitly fixed.) Denote $m := \lceil \log_2 n \rceil$. Recall that $x = (x_0, \dots, x_{m-1}$ and $y = (y_0, \dots, y_{\ell-1})$.

Lemma 1. *Let i be known and fixed. Then the functionality $g_i(x, y, y')$ can be implemented by a BDD $P_i(x, y, y')$ of size and length $m = (1 + o(1)) \log_2 n$.*

Proof. Formally, $P_i(x, y, y')$ consists of m nodes V_i, $i \in \{0, \dots, m-1\}$, such that the ith node is labeled by x_i. It also has two sinks labeled by y and y'. Moreover, there is an x_i-edge from V_i to V_{i+1} for $i < m - 1$, and an x_i-edge from V_{m-1} to the sink labeled by y. Finally, there is an $(1 - x_i)$-edge from every V_i to the sink labeled by y'. □

As an example, $P_6(x, y, f_6)$ is depicted by Fig. 1, left.

In the nonprivate version of the resulting PIR-writing protocol, the server just sets in parallel $f_i \leftarrow g_i(x, y, f_i)$ for every $i \in \{0, \dots, n-1\}$. (She can do it by using the BDD P_i, or by any other means.)

In the new PrivateBDD-based PIR-writing protocol, the server has to use a slightly different BDD $\tilde{P}_i$, as depicted by Fig. 1, right. Namely, due to the design of the PrivateBDD protocol, the siblings of any internal node have to have the same length. This is necessary to protect client's privacy. See Sect. 2.1 and [11] for more discussion. Since the value of the 0-sibling of the source on Fig. 1 is of form $\mathsf{E}_{\mathsf{pk}}(\mathsf{E}_{\mathsf{pk}}(\mathsf{E}_{\mathsf{pk}}(\cdot)))$, the easiest way to achieve the same length for the 1-sibling

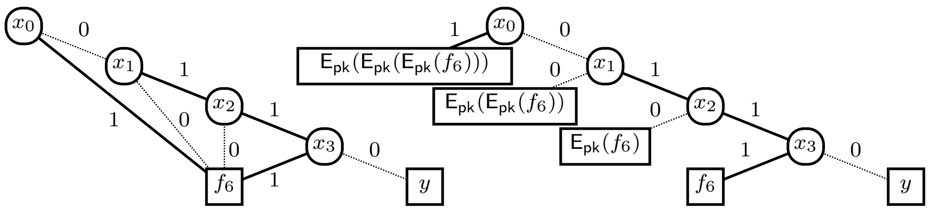

x_0
x_1
x_2
x_3
f_6
y
0
1
$\mathsf{E}_{\mathsf{pk}}(\mathsf{E}_{\mathsf{pk}}(\mathsf{E}_{\mathsf{pk}}(f_6)))$
$\mathsf{E}_{\mathsf{pk}}(\mathsf{E}_{\mathsf{pk}}(f_6))$
$\mathsf{E}_{\mathsf{pk}}(f_6)$

Fig. 1. Left: the BDD $P_6(x, y, f_6)$ that returns y if $x = 6$, and returns f_6 otherwise. The BDD outputs y only if $x_0 = 0$, $x_1 = 1$, $x_2 = 1$ and $x_3 = 0$. Right: the BDD $\tilde{P}_6$ that is actually used in the new PIR-writing protocol. Here $n = 16$ and $m = 4$.

is to use $\mathsf{E}_{\mathsf{pk}}(\mathsf{E}_{\mathsf{pk}}(\mathsf{E}_{\mathsf{pk}}(f_6)))$ as the value of its 1-sibling. The formal description of $\tilde{P}_i$ follows straightforwardly, and we will omit it.

Now, the server uses the PrivateBDD protocol with $\tilde{P}_i$ to update the encrypted version of f_i, given encryptions of x, y and an encryption of the old version of f_i. Thus, instead of sending the output back to the client, as in the original PrivateBDD protocol, the server uses it to update a value that is privately held by herself.

A formal protocol description of the new PrivateBDD-based PIR-writing protocol follows. Note that since we use a length-flexible cryptosystem, we can have $\ell > \kappa$.

PrivateBDD-Based New PIR-Writing Protocol

Common inputs: Database size n, $m \leftarrow \lceil \log_2 n \rceil$, element length ℓ.
Client's inputs: Secret key sk, $x = (x_0, \ldots, x_{m-1})$, $y \in \{0,1\}^\ell$.
Server's inputs: Public key pk, $c = (c_0, \ldots, c_{n-1})$, where c_j is a multiple-encryption of f_j.
Server's private output: Updated database $c' = (c'_0, \ldots, c'_{n-1})$, where c'_j is a multiple-encryption of updated f'_j.

1. Client sends to the server $\mathsf{E}_{\mathsf{pk}}(x_i)$, for $i \in \{0, m-1\}$, and $\mathsf{E}_{\mathsf{pk}}(y)$.
2. The server does:
   (a) For $i \in \{0, \ldots, n-1\}$, the server executes PrivateBDD by using $\tilde{P}_i(x, y, c_i)$, and sets c'_i to be equal to its output.
   (b) The server stores $c = (c'_0, \ldots, c'_{n-1})$ as the new database with elements having length $\ell' \leftarrow \ell + \kappa \cdot \lceil \log_2 n \rceil$.

Security. We claim the client-privacy only in the special case when the PrivateBDD protocol is based on Lipmaa's $(2,1)$-CPIR protocol from [10]. Obviously, this result can be generalized if other communication-efficient $(2,1)$-CPIR protocols were to be constructed.

Theorem 2. *1) The PrivateBDD-based PIR-writing protocol is correct. 2) If the Decisional Composite Residuosity assumption [14] holds, then the PrivateBDD-based PIR-writing protocol is client-private in the presence of a computationally-bounded malicious server.*

Proof (Sketch.)
1) The correctness of the PrivateBDD-based PIR-writing protocol follows from the correctness of the PrivateBDD protocol by Ishai and Paskin [9], and from the correctness of the BDDs $\tilde{P}_i$.

2) The DJ01 cryptosystem used in the this PIR-writing is known [4] to be CPA-secure under Decisional Composite Residuosity Assumption [14]. Because Lipmaa's $(2,1)$-CPIR is BDD-friendly and CPA-secure, the CPA-security of the PrivateBDD follows from a standard hybrid argument. □

Efficiency. Assume that the database elements have length ℓ. Recall that we are using the Damgård-Jurik cryptosystem [4]. Then before the first run of the PIR-writing protocol, the server has a database of once-encrypted elements that have size $(s+1)\kappa$, where $s = \lceil \ell/\kappa \rceil$. The output of the PrivateBDD protocol, as modified by Lipmaa [11], is a multiple encryption of the actual value of corresponding BDD sink value, where the number of multiples is equal to the depth of the BDD, that is, to $m = \lceil \log_2 n \rceil$ in this concrete case. Therefore, after every update, the (multiple-encrypted) database elements kept by the server increases by $m \cdot \kappa$ bits.

More precisely, the first run of the PIR-writing protocol has communication complexity

$$\leq (m+1) \cdot (\mathsf{len}(P)\kappa + \ell) = (m^2+m) \cdot \kappa + (m+1)\ell \ .$$

The new database has elements of size

$$(s+m+1) \cdot \kappa \leq (m+1) \cdot \kappa + \ell$$

and thus the next update protocol has communication

$$(m^2+m) \cdot \kappa + (m+1) \cdot (m\kappa + \kappa + \ell) \ .$$

Analogously, the jth update protocol has communication

$$(m^2+m) \cdot \kappa + (m+1)(jm\kappa + j\kappa + \ell) \ ,$$

and thus the first u protocols have total communication

$$u(m^2+m) \cdot \kappa + (m+1) \sum_{j=0}^{u-1} (jm\kappa + j\kappa + \ell) = O(mu\ell + m^2u^2\kappa) \ .$$

Since the working time of the PrivateBDD protocol is proportional to the size of the BDD, the computation of the new PrivateBDD-based PIR-writing protocol is dominated by $n \cdot (2m-1)$ public-key operations, where the cost of public-key operations increases after every update. Here, $m-1$ public-key operations come from the need to multiple-encrypt every database element f_i, so that it would have correct length to be on the correct layer of $\tilde{P}_i$. However, since we do not care about server's privacy at all, $\tilde{P}_i$ can be simplified: instead of an j-times encryption of f_i we just use a suitably padded once-encryption of f_i. Thus, we have proven the next result:

Lemma 2. *The amortized communication complexity of the first u updates of the PrivateBDD-based PIR-writing protocol is $O(u\ell \cdot \log n + u^2\kappa \cdot \log^2 n)$, and its computational complexity is dominated by $nm \approx n \cdot \log_2 n$ public-key operations, where the cost of a single public-key operation depends in ℓ, and increases with every update.*

Comparison to Previous Work. The PrivateBDD-based PIR-writing protocol is more communication-efficient than the PIR-writing protocols of [2,3] if $u = o(\sqrt[4]{\ell^{1+\alpha} \cdot n} \cdot \text{polylog}(n))$. Unfortunately, during this protocol the database kept by the server will increase in length, and therefore every new update will be more and more expensive. This can be partially solved by letting the client and the server refresh the database after every (say) $\sqrt[4]{n}$ updates.

However, importantly, this new PrivateBDD-based PIR-writing protocol does *not* use pairings. In particular, it relies on the classical Decisional Composite Residuosity assumption [14] which is by far the weakest security assumption under which a nontrivial PIR-writing protocol is known. Even in the trivial PIR-writing protocol, the database has to encrypted by using a CPA-secure public-key cryptosystem. It is conceivable that this cryptosystem must be additively homomorphic to allow the client and the server to execute some additional simple cryptocomputing protocols on encrypted database. In such a case, the PrivateBDD-based PIR-writing protocol comes "free" in the sense of security, adding no extra security assumptions.

3.1 Write-Optimized PrivateBDD-Based PIR-Writing Protocol

Clearly, the client's write requests can be seen as program code that modifies the database. In the previously presented PrivateBDD-based PIR-writing protocol, the server applies the client's program (BDD) to her database and stores the program outputs as the new database. Unfortunately, the program's outputs are multiple-encrypted and thus the new database will have longer elements.

Given this interpretation, it is easy to see that one can optimize the presented PIR-writing protocol as follows. The server will not run the client's program (BDD) at every run of the PIR-writing protocol. Instead, she will append it to the list of previous "programs". More precisely, the server will just store all client's messages as follows.

Write-Optimized PrivateBDD-Based PIR-Writing Protocol

Common inputs: Database size n, $m \leftarrow \lceil \log_2 n \rceil$, element length ℓ.
Client's inputs: Secret key sk, $x = (x_0, \dots, x_{m-1})$, $y \in \{0,1\}^\ell$.
Server's inputs: Public key pk, $c = (c_0, \dots, c_{n-1})$, where c_j is an encryption of f_j.
Server's private output: Updated state *state*. Initially *state* is empty string.

1. Client sends to the server $C \leftarrow \mathsf{E}_{\mathsf{pk}}(x_i)$, for $i \in \{0, m-1\}$, and $D \leftarrow \mathsf{E}_{\mathsf{pk}}(y)$.
2. The server does: Set $state \leftarrow state || C$.

In the corresponding CPIR protocol (that later reads an element from the database), the client sends his query to the server, who then applies all stored programs to it, and to the database:

CPIR Protocol Corresponding to Write-Optimized PIR-Writing

Common inputs: Database size n, $m \leftarrow \lceil \log_2 n \rceil$, element length ℓ.
Client's inputs: Secret key sk, x.
Server's inputs: Public key pk, $c = (c_0, \ldots, c_{n-1})$, where c_j is an encryption of f_j, and $state = (C_1, D_1)|| \ldots ||(C_u, D_u)$.
Client's private output: Current value of f_x.

```
1. Client sends to the server the query of CPIR protocol.
2. The server does:
   (a) For $i \in \{0, \ldots, n-1\}$:
         i. Let $c'_{0,i} \leftarrow c_i$.
        ii. For $j \in \{1, \ldots, u\}$: execute PrivateBDD by using $\tilde{P}_i(x, y, c'_{j-1,i})$,
            and set $c'_{j,i}$ to be equal to its output.
3. The server executes server's part in the CPIR protocol by using database
   $c'_u = (c'_{u1}, \ldots, c'_{un})$.
```

Note that this variant achieves very fast writing (and also, good memory efficiency) by offloading computational cost to the reading phase. This solution is good, for example, when the client updates his database often but needs to read its values quite rarely.

4 New FH-Based PIR-Writing Protocol

In this section, we describe another PIR-writing protocol that is based on a fully-homomorphic cryptosystem $\mathsf{P} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$. First, we need the next result.

Lemma 3. *Assume i is known. Let $\mathsf{eq}_i(x) = 1$ if $i = x$, and let $\mathsf{eq}_i(x) = 0$ otherwise. Then a server who knows $\mathsf{E}_{\mathsf{pk}}(x_j)$, for $0 \leq j < m$, can homomorphically evaluate $\mathsf{E}_{\mathsf{pk}}(\mathsf{eq}_i(x))$ by using a circuit of size $m-1$ and of depth $\lceil \log_2 m \rceil$.*

Proof. The circuit has m leaves, labeled by $x_j^{i_j}$ for $0 \leq j < m$, where $x_j^0 = \neg x_j = 1 - x_j$, and $x_j^1 = x_j$. This can implemented with almost no cost: if $i_j = 1$, then the server inputs $\mathsf{E}_{\mathsf{pk}}(x_j)$ to the gate; if $i_j = 0$, then the server inputs $\mathsf{E}_{\mathsf{pk}}(1 - x_j)$ to the gate. After that the server "AND"-s all m inputs together by using $m-1$ AND-gates arranged in a circuit of depth $\lceil \log_2 m \rceil$, as depicted by Fig. 2. Finally, a 2-fan-in AND-gate can be implemented by setting $\mathsf{E}_{\mathsf{pk}}(a \wedge b) = \mathsf{E}_{\mathsf{pk}}(a) \cdot \mathsf{E}_{\mathsf{pk}}(b)$. $\square$

The server then cryptocomputes the value of

$$\mathsf{E}_{\mathsf{pk}}(f'_i) \leftarrow (\mathsf{E}_{\mathsf{pk}}(y) - \mathsf{E}_{\mathsf{pk}}(f_i)) \cdot \mathsf{E}_{\mathsf{pk}}(\mathsf{eq}_i(x)) + \mathsf{E}_{\mathsf{pk}}(f_i) \ , \tag{1}$$

and then replaces $\mathsf{E}_{\mathsf{pk}}(f_i)$ with $\mathsf{E}_{\mathsf{pk}}(f'_i)$.

This idea can be obviously extended to ℓ-bit database elements, although then every bit f_{ij}, $0 \leq j < \ell$, of every database element f_i, $0 \leq i < n$, has to be encrypted separately. On the other hand, the circuit eq_i has only to be evaluated once per every $i \in \{0, \ldots, n-1\}$. The full protocol description follows.

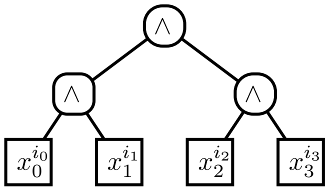


Fig. 2. Circuit eq_i for comparing x and i. Here, $n = 16$ and thus $m = 4$. Moreover, $x^i = x$ if $i = 1$, and $x^i = 1 - x$ if $i = 0$.

FH-Based New PIR-Writing Protocol

Common inputs: Database size n, $m \leftarrow \lceil \log_2 n \rceil$, element length ℓ.
Client's inputs: Secret key sk, $x = (x_0, \dots, x_{m-1})$, $y \in \{0,1\}^\ell$.
Server's inputs: Public key pk, $c = (c_{00}, \dots, c_{n-1,\ell-1})$ where $c_{ij} = \mathsf{E}_{\mathsf{pk}}(f_{ij})$.
Server's private output: New updated database $c' = (c'_{00}, \dots, c'_{n-1,\ell-1})$.

1. The client sends $(\mathsf{E}_{\mathsf{pk}}(x_0), \dots, \mathsf{E}_{\mathsf{pk}}(x_{m-1}))$ and $(\mathsf{E}_{\mathsf{pk}}(y_0), \dots, \mathsf{E}_{\mathsf{pk}}(y_{\ell-1}))$ to the server.
2. The server does in parallel for $i \in \{0, \dots, n-1\}$:
   (a) The server runs encrypted circuit $b_i \leftarrow \mathsf{E}_{\mathsf{pk}}(\mathsf{eq}_i(x))$.
   (b) For $0 \le j \le \ell$, the server computes and stores
   $$c'_{ij} \leftarrow (\mathsf{E}_{\mathsf{pk}}(y_j) - c_{ij}) \cdot b_i + c_{ij} \enspace .$$
   (c) (If needed,) the server bootstraps all values c'_{ij} to decrease the noise.

Theorem 3. *Assume that the underlying fully-homomorphic cryptosystem is correct and CPA-secure. Then the new FH-based PIR-writing protocol is correct and client-private. Moreover, it has communication complexity $O(\kappa \cdot \log n + \kappa\ell)$, and its computational complexity is dominated by $O(n \cdot \log n + \ell \cdot n)$ applications of Gentry's cryptosystem. (Here we do not count the cost of bootstrapping.)*

Proof (Sketch.). Correctness follows from the correctness of the fully-homomorphic cryptosystem, and from the correctness of the circuit for eq_i. Moreover, since the depth of the circuit used inside the FH-based protocol is only $\approx \log_2 \log_2 n$ (which is never larger than 5 is practice), we can freely assume that there is no need to bootstrap the circuit before the end of the PIR-writing protocol.

The client-privacy of the FH-based PIR-writing protocol follows from the CPA-security of the fully-homomorphic cryptosystem by a standard hybrid argument with $m + \ell$ games. Again, the security is even achieved for computationally bounded malicious server. The efficiency is clear. □

Comparison to Previous Work. The FH-based protocol has (asymptotically) optimal communication complexity, and differently from the PrivateBDD-based protocol, the database elements do not grow in length after every update. On the other hand, it is based on a fully-homomorphic cryptosystem, and all existing fully-homomorphic cryptosystems rely on relatively new security assumptions. Moreover, the necessary bootstrapping step makes it computationally less efficient.

Further Work. The concept of PIR-writing is somewhat similar to the concept of oblivious RAM, where the client (a CPU) wants to make a number of read/write queries to the client (a RAM). It is required that the distribution of client's executed read/write queries must be completely independent of the read/write queries the client actually intends to do. More precisely, for every "program" of the same length, the distribution of executed queries must be the same. The adversary is positioned inbetween the client and (honest) server, but has read access to all the executed queries. It was shown in [12] that one can implement oblivious RAM with a polylogarithmic overhead (in the size of the "program"). We leave it as an interesting open question whether the techniques of [12] can be used to construct (even more) efficient PIR-writing protocols, or alternatively, our techniques can be used to construct more efficient oblivious RAM protocols.

Acknowledgments. The authors were supported by Estonian Science Foundation, grant #8058, and European Union through the European Regional Development Fund.

References

1. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
2. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith III, W.E.: Public Key Encryption That Allows PIR Queries. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 50–67. Springer, Heidelberg (2007)
3. Chandran, N., Ostrovsky, R., Skeith III, W.E.: Public-Key Encryption with Efficient Amortized Updates. Tech. Rep. 2008/429, International Association for Cryptologic Research (2008), http://eprint.iacr.org/2008/429
4. 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)
5. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, Springer, Heidelberg (2010)
6. Gentry, C.: A Fully Homomorphic Encryption Scheme. Ph.D. thesis, Stanford (September 2009)
7. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: Mitzenmacher, M. (ed.) STOC 2009, May 31-June 2, pp. 169–178. ACM Press, Bethesda (2009)

8. Gentry, C., Ramzan, Z.: Single-Database Private Information Retrieval with Constant Communication Rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg (2005)
9. 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)
10. Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)
11. Lipmaa, H.: First CPIR Protocol with Data-Dependent Computation. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984. Springer, Heidelberg (2009)
12. Ostrovsky, R.: Efficient Computation on Oblivious RAMs. In: STOC 1990, Baltimore, Maryland, USA, May 14-16, pp. 514–523 (1990)
13. Ostrovsky, R., Shoup, V.: Private Information Storage. In: STOC 1997, pp. 294–303 (1997)
14. 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)
15. Wegener, I.: Branching Programs and Binary Decision Diagrams: Theory and Applications. Monographs on Discrete Mathematics and Applications. Society for Industrial Mathematics (2000)

Regulatory Compliant Oblivious RAM

Bogdan Carbunar[1] and Radu Sion[2]

[1] Motorola Labs
carbunar@motorola.com
[2] Stony Brook Network Security and Applied Cryptography Lab
sion@cs.stonybrook.edu

Abstract. We introduce WORM-ORAM, a first mechanism that combines Oblivious RAM (ORAM) access privacy and data confidentiality with Write Once Read Many (WORM) regulatory data retention guarantees. Clients can outsource their database to a server with full confidentiality and data access privacy, and, for data retention, the server ensures client access WORM semantics. In general *simple confidentiality* and *WORM* assurances are easily achievable e.g., via an encrypted outsourced data repository with server-enforced read-only access to existing records (albeit encrypted). However, this becomes hard when also *access privacy* is to be ensured – when client access patterns are necessarily hidden and the server cannot enforce access control directly. WORM-ORAM overcomes this by deploying a set of zero-knowledge proofs to convince the server that all stages of the protocol are WORM-compliant.

1 Introduction

Regulatory frameworks impose a wide range of policies in finance, life sciences, health-care and the government. Examples include the Gramm-Leach-Bliley Act [1], the Health Insurance Portability and Accountability Act [2] (HIPAA), the Federal Information Security Management Act [3], the Sarbanes-Oxley Act [4], the Securities and Exchange Commission rule 17a-4 [5], the DOD Records Management Program under directive 5015.2 [6], the Food and Drug Administration 21 CFR Part 11 [7], and the Family Educational Rights and Privacy Act [8]. Over 10,000 regulations are believed to govern the management of information in the US alone [9].

A recurrent theme to be found throughout a large part of this regulatory body is the need for assured lifecycle storage of records. A main goal there is to support WORM semantics: once written, data cannot be undetectably altered or deleted before the end of its regulation-mandated life span. This naturally stems from the perception that the primary adversaries are powerful insiders with superuser powers coupled with full access to the storage system. Indeed much recent corporate malfeasance has been at the behest of CEOs and CFOs, who also have the power to order the destruction or alteration of incriminating records [10].

Major storage vendors have responded by offering compliance storage and WORM products, for on-site deployment, including IBM [11], HP [12], EMC

[13]. Hitachi Data Systems [14], Zantaz [15], StorageTek [16], Sun Microsystem [17] [18], Network Appliance [19]. and Quantum Inc. [20].

However, as data management is increasingly outsourced to third party "clouds" providers such as Google, Amazon and Microsoft, existing systems simply do not work. When outsourced data lies under the incidence of both mandatory *data retention* regulation and privacy/confidentiality concerns – as it often does in outsourced contexts – new enforcement mechanisms are to be designed.

This task is non-trivial and immediately faces an apparent contradiction. On the one hand, data retention regulation stipulates that, once generated, data records cannot be erased until their "mandated expiration time", *even by their rightful creator* – history cannot be rewritten. On the other hand, access privacy and confidentiality in outsourced scenarios mandate non-disclosure of data and patterns of access thereto to the providers' servers, and can be achieved through "Oblivious RAM" (ORAM) based client-server mechanisms [21,22]. Yet, by their very nature, existing ORAM mechanisms allow clients unfettered read/write access to the data, including the full ability to alter or remove previously written data records – thus directly contradicting data retention requirements.

Basic *confidentiality* and *WORM* assurances are achievable e.g., via traditional systems that could encrypt outsourced data and deploy server-enforced read-only access to data records once written. Yet, when also *access privacy* is to be ensured, client access patterns become necessarily hidden and the server cannot enforce WORM semantics directly.

In this paper we introduce WORM-ORAM, a first mechanism that combines the access privacy and data confidentiality assurances of traditional ORAM with Write Once Read Many (WORM) regulatory data retention guarantees. Clients can outsource their database to a server with full confidentiality and data access privacy, and, for data retention, the server ensures client access WORM semantics, i.e., specifically that client access is append-only: – once a data record has been written it cannot be removed or altered even by its writer.

WORM-ORAM is built around a set of novel efficient zero knowledge (ZK) proofs. The main insight is to allow the client unfettered ORAM access with full privacy to the server-hosted encrypted data set while simultaneously proving to the server *in zero-knowledge* – at all stages of the ORAM access protocol – that no existing records are overwritten and WORM semantics are preserved.

Specifically, clients can add encrypted data records to the database ("the ORAM") hosted by a service provider. Each record will be associated with a regulatory mandated expiration time. Once stored, the client can read all data obliviously, (and add new records) – only leaking that access took place and nothing else. No access patterns or data records or any other information is leaked. The server, without having plaintext access to the data or the client access patterns then ensures – in a client-server interaction – that any client access is WORM compliant: it is either a read of an existing record, or an addition of a new record (with a new index – no overwriting permitted).

To achieve the above, at an overview level, the solution outlines as follows. The server hosts two ORAMs, one storing the actual data items (the W-ORAM) and one allowing the private retrieval of items expiring at any given time (the E-ORAM). The E-ORAM is effectively a helper data structure allowing the client to determine which items to expire at given time intervals. Client access to the E-ORAM needs to be private, but does not need to be proved correct.

The server exports an access API to the W-ORAM to the client consisting of four types of operations: write, read, expire and compliance verification. For access pattern privacy purposes as in the traditional ORAM protocols, the data set stored at the server contains both "real" and "fake" items – this is discussed later. Then, during any legitimate access to the W-ORAM the client will prove in ZK to the server that the item written is real, "well formed" and can be decrypted later in the case of an audit. Moreover she also proves that the access does not in fact overwrite any existing database item. Similarly, in the expiration operation, the client proves in ZK that the element to be removed from the W-ORAM has indeed expired. Finally, at audit time, the data has to be accessible to an authorized auditor, even in the case of a non-cooperating client (e.g., that could refuse to reveal encryption keys).

We show that our solution does not change the computational complexity of existing ORAM implementations. However, we warn that the constants involved are non-negligible and render this result of theoretical interest only for now. Future work focuses on reducing these overheads towards true practical efficiency.

2 Related Work

2.1 Oblivious RAM

Oblivious RAM [21] provides access pattern privacy to clients (or software processes) accessing a remote database (or RAM), requiring only logarithmic storage at the client. The amortized communication and computational complexities are $O(log^3 n)$. Due to a large hidden constant factor, the ORAM authors offer an alternate solution with computational complexity of $O(log^4 n)$, that is more efficient for all currently plausible database sizes.

In ORAM, the database is considered a set of n encrypted blocks and supported operations are read(id), and write(id, $newvalue$). The data is organized into $log_4(n)$ levels, as a pyramid. Level i consists of up to 4^i blocks; each block is assigned to one of the 4^i buckets at this level as determined by a hash function. Due to hash collisions each bucket may contain from 0 to $\log n$ blocks.

ORAM Reads. To obtain the value of block id, the client must perform a read query in a manner that maintains two invariants: (i) it never reveals which level the desired block is at, and (ii) it never looks twice in the same spot for the same block. To maintain (i), the client always scans a single bucket in every level, starting at the top (Level 0, 1 bucket) and working down. The hash function informs the client of the candidate bucket at each level, which the client then scans. *Once the client has found the desired block, the client still proceeds to each lower level, scanning random buckets instead of those indicated by their hash function.* For (ii), once all levels have been queried, the client re-encrypts the query result with a different nonce and places it in the *top* level. This ensures that when it repeats a search for this block, it will locate the block immediately (in a different location), and the rest of the search pattern will be randomized. The top level quickly fills up; how to dump the top level into the one below is described later.

ORAM Writes. Writes are performed identically to reads in terms of the data traversal pattern, with the exception that the new value is inserted into the top level at the end. Inserts are performed identically to writes, since no old value will be discovered in the query phase. Note that semantic security properties of the re-encryption function ensure the server is unable to distinguish between reads, writes, and inserts, since the access patterns are indistinguishable.

Level Overflow. Once a level is full, it is emptied into the level below. This second level is then re-encrypted and re-ordered, according to a new hash function. Thus, accesses to this new generation of the second level will hence-forth be completely independent of any previous accesses. Each level overflows once the level above it has been emptied 4 times. Any re-ordering must be performed obliviously: once complete, the adversary must be unable to make any correlation between the old block locations and the new locations. A sorting network is used to re-order the blocks.

To enforce invariant (i), note also that all buckets must contain the same number of blocks. For example, if the bucket scanned at a particular level has no blocks in it, then the adversary would be able to determine that the desired block was *not* at that level. Therefore, each re-order process fills all partially empty buckets to the top with *fake* blocks. Recall that since every block is encrypted with a semantically secure encryption function, the adversary cannot distinguish between fake and real blocks.

Oblivious Scramble. In [22] Williams et al.introduced an algorithm that performs an oblivious scramble on a array of size n, with $c\sqrt{n}$ local storage, in $O(n \log \log n)$ time with high probability. Informally, the algorithm is a merge sort, except a random number generator is used in place of a comparison, and multiple sub-arrays are merged simultaneously. The array is recursively divided into segments, which are then scrambled together in groups. The time complexity of the algorithm is better than merge sort since multiple segments are merged together simultaneously. Randomly selecting from the remaining arrays avoids comparisons among the leading items in each array, so it is not a comparison sort.

2.2 Oblivious Transfer with Access Control

Camenish et al. [23] study the problem of performing k sequential oblivious transfers (OT) between a client and a server storing N values. The work makes the case that previous solutions tolerate selective failures. A selective failure occurs when the server may force the following behavior in the ith round (for any i=1..k): the round should fail if the client requests item j (of the N items) and succeed otherwise. The paper introduces security definitions to include the selective failure problem and then propose two protocols to solve the problem under the new definitions.

Coull et al. [24] propose an access control oblivious transfer problem. Specifically, the server wants to enforce access control policies on oblivious transfers performed on the data stored: The client should only access fields for which it has the credentials. However, the server should not learn which credentials the client has used and which items it accesses.

Note that the above oblivious transfer flavors do not consider by definition the problem of obliviously enforcing WORM semantics as well as writing to the data. Our regulatory compliant problem is complicated by the fact that we also allow clients to add to the database while proving that operations performed on the data do not overwrite old records. One can trivially extend OT with an add call, by imposing $O(N)$ communication and computation overheads. However, by building our solution on ORAM we can perform both read and add operations with only poly-logarithmic complexity and traffic overheads.

3 Model and Preliminaries

3.1 Deployment and Threat Model

In the deployment model for networked compliance storage, a legitimate client creates and stores records with a (potentially untrusted) remote WORM storage service. These records are to be available later to both the client for read as well as to auditors for audits. Network layer confidentiality is assured by mechanisms such as SSL/IPSec. Without sacrificing generality, we will assume that the data is composed of equal-sized blocks (e.g., disk blocks, or database rows).

At a later time, a previously stored record's existence is regretted and the client will do everything in her power – e.g., attempt to convince the server to remove the record – to prevent auditors from discovering the record. The main purpose of a traditional WORM storage service is to defend against such an adversary.

Moreover, numerous data regulations feature requirements of "secure deletion" of records at the end of their mandated retention periods. Then, in the WORM adversarial model the focus is mainly on preventing clients from "rewriting" history, rather than "remembering" it. Additionally, we prevent the rushed removal of records before their retention periods. Thus, the traditional Write-Once Read-Many (WORM) systems have the following properties:

– Data records may be written by clients to the server once, read many times and not altered for the duration of their life-cycle.
– Records have associated mandatory expiration times. After expiration, they should not be accessible for either audit or read purposes.
– In the case of audits, stored data should be accessible to auditors even in the presence of a non-cooperating client refusing to reveal encryption keys. Compliant record expiration of inaccessible records should be easily proved to auditors.

Additionally, when records and their associated access patterns are sensitive they need to be concealed from a curious server. The main purpose of WORM-ORAM is to enable WORM semantics while preserving data confidentiality and access pattern privacy. This inability of the server to "see" data and associated access patterns prevents the deployment of conventional file/storage system access control mechanisms or data outsourcing techniques. Thus, we have the following additional requirements:

– Data records are encrypted from the server (*confidentiality*).
– The server cannot distinguish between different read operations targeting the same or different data records (*access privacy*).
– During a read, in the ORAM protocol, to enforce *WORM semantics*, clients will need to prove to the server that any access did not remove data records. Specifically, when re-inserting one of the read elements back into the root of the ORAM pyramid, the client needs to prove to the server in ZK that the inserted element is a correct re-encryption of the previously removed "real" element (see Section 2.1 for details).
– During a re-shuffle, in the ORAM protocol, clients can prove that no "real" elements were converted into "fake" ones.

Additionally, in the WORM-ORAM scenario, we assume the following:

– The server is allowed to distinguish between record expiration, read and write operations.
– Clients participate correctly in any record expiration protocol. This is reasonable to assume because the regulatory compliance scenario allows clients always to by-pass the server-enforced storage service and store select records elsewhere.

Several participants are of concern. First, clients have incentives to *rewrite history* and alter or completely remove *previously written* records. We note that in the regulatory scenario, there exists an apparent imbalance – clients are assume to correctly store records at the time of their creation – only later does regretting the past becoming a concern. Thus the main focus of WORM assurances is not to prevent history but rather just its rewriting. In reality, the "regret" time interval between the creation/storage and regretting of a record is non-zero, application-specific, and often quite large. To remove any application dependency, here we consider the strongest WORM guarantees, in which records are not to be altered as soon as they are written.

Second, the storage provider (server) is *curious* and has incentives to illicitly gain information about the stored data and access patterns thereto. As the regulatory storage provider, the server is the main enforcer of WORM semantics and record expiration. Naturally, the server is assumed to not collude with clients illicitly desiring to alter their data. In summary, the server is trusted to run protocols correctly yet it may try to use information obtained from correct runs to obtain undesired information. This assumption is natural and practical as otherwise one can easily imagine a server simply deleting stored records in a denial of service attack. Basic denial of service on the client or server side is not of interest here.

We consider a server S with $O(N)$ storage and a client C with $O(\sqrt{N}\log N)$ local storage. The client stores $O(N)$ items on the server. We denote the regulatory compliance auditor by $\mathcal{A}$.

3.2 Cryptography

We require several cryptographic primitives with all the associated semantic security [25] properties including: a secure, collision-free hash function which builds a distribution from its input that is indistinguishable from a uniform random distribution, a semantically secure cryptosystem (Gen, Enc_k, Dec_k), where the encryption function Enc generates unique ciphertexts over multiple encryptions of the same item, such that a computationally bounded adversary has no non-negligible advantage at determining whether a pair of encrypted items of the same length represent the same or unique items, and a pseudo random number generator whose output is indistinguishable from a uniform random distribution over the output space.

The Decisional Diffie-Hellman (DDH) assumption over a cyclic group G of order q and a generator g states that no efficient algorithm can distinguish between two distributions (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where a, b and c are randomly chosen from $\mathbb{Z}_q$.

An integer v is said to be a *quadratic residue* modulo an integer n if there exists an integer x such that $x^2 = v \, mod \, n$. Let QRA be the quadratic residuosity predicate modulo n. That is, $QR(v, n) = 1$ if v is a residue mod n and $QR(v, n) = 0$ if v is a quadratic non-residue.

Given an odd integer $n = pq$, where p and q are odd primes, the quadratic residuosity (QR) assumption states that given n but not its factorization and an integer v whose Jacobi symbol $(v|n) = 1$ it is difficult to determine whether $QR(v, n)$ is 1 or 0.

The Goldwasser, Micali and Rackoff [26] *zero knowledge proof of quadratic non-residuosity* proceeds roughly as follows. Given two parties A and B, A claims knowledge of $QR(v, n) = 0$, for $(v|n) = 1$. A proves this in zero knowledge to B, that is, without revealing n's factorization. To achieve this, B selects m random values $r_1, .., r_m$ and flips m coins. For each coin c_i, if $c_i = 0$ B computes $x_i = r_i^2 \, mod \, n$ to A, otherwise it computes $x_i = vr_i^2 \, mod \, n$. B sends all computed x_i values to A. A needs to send back the square roots of the quadratic residues it detects in the list $x_1, ..x_m$. If $QR(v, n) = 0$, then A correctly detects the residues. If $QR(v, n) = 1$, all the values received by A will be quadratic residues. A can then cheat only with probability $1/2^m$.

Notations: Let $n = pq$ be a large composite, where p and q are primes. Let $\phi(n)$ denote the Euler totient of n. We will use $x \in_R A$ to denote the random uniform choice of x from the set A. Given a value m, let $\mathcal{P}(m)$ denote the group of permutations over the set $\{0, 1\}^m$. Let $\mathbf{k} < |n|$ be a security parameter. Let

N denote the set of elements stored in the ORAM. Let $\mathbb{W}_m$ be the universe of all sets of m quadratic residues.

4 Solution Overview

A WORM-ORAM system, consists of two ORAMs (W-ORAM,E-ORAM) and a set of operations (Gen, Enc, Dec, RE, Write, Read, Expire, Shuffle, Audit) that can be used to access the ORAMs. The client needs to store elements at the server while preserving the privacy of its accesses and allowing the server to preserve the data's WORM semantics. W-ORAM serves this purpose: it is used by the client to store $(label, element)$ pairs.

We organize time into epochs: each element stored at the server expires in an integer number of epochs, as determined by the client. The client needs to remember the expiration time of each element stored in the W-ORAM. The client uses the E-ORAM to achieve this, to store expiration times of labels used to index elements stored in W-ORAM. When queried with a time epoch, E-ORAM provides a list of labels expiring in that epoch. The labels are then used to retrieve the expiring elements from the W-ORAM.

The E-ORAM is stored and accessed as a regular ORAM [27]. It is used as an auxiliary storage structure by the client and it needs not be WORM compliant. The W-ORAM on the other hand stores actual elements and needs to be made WORM compliant. The W-ORAM stores two types of elements: "reals" and "fakes". A real element has a quadratic non-residue component, whereas a fake has a quadratic residue. Each time the ORAM is accessed, elements are re-encrypted to ensure access privacy. The client has then to prove in ZK that (i) an element is real or fake and (ii) a re-encrypted element decrypts to the same cleartext as the original element. We now provide a brief overview of each operation described above and follow with a detailed description in the next section.

Gen. Operation executed initially, to generate system parameters for each participant: client, server, auditor.

Enc, Dec, RE. *Enc* and *Dec* provide the basic encryption and decryption operations for elements to be stored in the W-ORAM. *RE* is the W-ORAM element re-encryption operation. *RE* is needed to ensure that the server cannot distinguish the same W-ORAM element accessed multiple times, while allowing the server to prove in zero knowledge the element's correctness.

Write. Operation used by the client to store an element on the server. The client needs to label the element and determine its expiration time. The client stores the element indexed by the label on the W-ORAM and the label indexed by the element's expiration time in the E-ORAM.

Read. Allows the client to retrieve from the W-ORAM an element indexed by an input label. The operation is based on existing ORAM reading techniques. In addition, it obliviously ensures that the client cannot remove or alter any real element from the W-ORAM.

Shuffle. Re-shuffles a level (provided as input) in the W-ORAM. Based on existing ORAM shuffling techniques, it needs to ensure that the client cannot remove or alter existing W-ORAM elements.

Expire. This operation makes use of both the E-ORAM and W-ORAM to remove all elements from the W-ORAM whose expiration time equals an input expiration time epoch. The operation needs to obliviously convince the server that only expiring elements are removed and no other W-ORAM elements are altered.

Audit. Enables an auditor to access the entire W-ORAM and search for keywords of interest.

5 Solution

Gen(k). Generate $p = 2p' + 1$, $q = 2q' + 1$ such that p, p', q, q' are primes. Let $n = pq$. Let G be the cyclic subgroup of order $(p-1)(q-1)$. DDH is believed to be intractable in G [28]. Let g be a generator of G. Let a be a random value and let $d = a^{-1} \ mod \ \phi(n)$. Let k be a random key in a semantically secure symmetric cryptosystem. *Gen* gives k, n, g, $h = g^a \in G$, p, q, a and d to the client and n, g, h to the server. *Gen* also gives k, p, q, a and d to the auditor.

Enc((x,T_{exp}),k,g,h,G,f). Encrypt an element of value x with expiration time T_{exp}, using the client's view of *Gen*'s output as input parameters. The output of the operation is a tuple $(A, B) \in G \times G$ that can be stored on the W-ORAM. If $f = 0$, *Enc* generates a "real" W-ORAM element: the first field of such elements is a quadratic non residue, $QR(A, n) = 0$. The tuple is computed as follows. First, generate a random $r \in \{0,1\}^{\mathbf{k}}$ and use it to compute $M(x) = \{E_k(x), T_{exp}, \text{“}real'', r\}$ where $E_k(x)$ denotes the semantically secure encryption of item x with symmetric key k and "real" is a pre-defined string. The random r is chosen (using trial and failure) such that $QR(M(x), n) = 0$ (quadratic non residue mod n) whose Jacobi symbol is 1. Second, generate a random odd value $b \in_R \{0,1\}^{\mathbf{k}}$ and output the tuple $S(x) \in G \times G$ as

$$S(x) = (A, B) = (M(x)g^{2b}, h^{2b})).$$

$S(x)$ is said to be an "W-ORAM element", whose first field is the "encrypted element" and second field is called the "recovery key". Notice that since $M(x)$ is a QNR, $QR(M(x)g^{2b}, n) = 0$ with (Jacobi symbol) $(M(x)g^{2b}|n) = 1$.

If $f = 1$, *Enc* generates a "fake" W-ORAM element: the first field of fake elements is a quadratic residue, $QR(A, n) = 1$. To compute a fake element, *Enc* generates random $s, k \in_R \{0,1\}^{\mathbf{k}}$ and outputs the tuple $(s^2 \ mod \ n, k)$.

Dec((A,B),d,k). Decrypt a real W-ORAM element, given the secret key $d = a^{-1}$. Compute $M = AB^{-d}$. M has format $\{E, T_{exp}, \text{“}real'', r\}$. The operation outputs the tuple $Dec_k(E), T_{exp}$.

RE(A,B). Re-encrypt element (A, B). Choose $u \in_R \{0,1\}^{\mathbf{k}}$, called re-encryption factor. Output pair $(A', B') = (Ag^{2u}, Bh^{2u})$. Note that knowledge of the message M encoded in (A, B) is not required. Alternatively, if M is known such that $A = Mg^{2b}$ and $B = h^{2b}$, then output $(A', B') = (Mg^{2u}, h^{2u})$. Note that u may also be used as an input parameter by $RE((A, B), u)$.

RE(L). Generalization of $RE((A, B))$, where $L = \{(A_1, B_1), .., (A_m, B_m)\}$ is a list of W-ORAM elements. Choose $\bar{u} = \{u_1, .., u_m\}$, such that $u_i \in_R \{0,1\}^{\mathbf{k}}$. $\bar{u}$ is called the re-encryption vector. Output $L' = \{RE((A_i, B_i), u_i)\}_{i=1..m}$. We also use the notation $L' = L\bar{u}$ and call L' a "correct re-shuffle" of L.

We now prove the semantic security of *Enc*.

Theorem 1. *Enc is IND-CPA secure.*

Proof. Let Q be an adversary that can break the semantic security of *Enc* with advantage ϵ. We then build an adversary Q^* that can break the DDH assumption in G without knowing n's factors, with probability ϵ. Let CH be a challenger. CH interacts with Q^* by sending the triple $(A = g^a, B = g^b, C = g^c)$. Q^* needs to decide whether $c = ab$ or is randomly distributed.

Q^* sends A to Q as the public key (h in our protocol). A then sends to Q^* two messages M_0 and M_1. Q^* picks a bit $\alpha \in_R \{0,1\}$ randomly and sends back to Q the tuple $(M_\alpha B^2, C^2)$. Q sends back its guess for α. If Q guesses correctly, Q^* sends to CH the value 1 ($c = ab$) or 0 (c is random).

When $c = ab$, the tuple $(M_\alpha B^2, C^2)$ is a correct ciphertext of M_α. Then, the interaction between Q^* and Q is correct and the probability of Q^* to output 1 is $1/2 + \epsilon$. When c is distributed randomly, the tuple $(M_\alpha B^2, C^2)$ is independent of α. The probability of Q^* outputting 1 is then 1/2. Thus, Q^* has advantage ϵ in the DDH game.

Note that we can similarly prove that RE is IND-CPA. That is, given two encryptions (A_0, B_0) and (A_1, B_1) of any two messages and a re-encryption $RE(A_b, B_b)$, $b \in_R \{0,1\}$ of one of the two encryptions, an attacker cannot guess b with non-negligible probability over 1/2. In the following, we describe first the main E-ORAM operations and then the W-ORAM accessing operations.

5.1 Accessing E-ORAM

The E-ORAM is a standard ORAM, storing labels indexed under expiration time epochs. The E-ORAM needs to provide C with the means to determine how many and which labels expire at a given time epoch and also to insert a new $(epoch, label)$ pair. This is achieved in the following manner. For each T_{exp} value used to index labels in E-ORAM, a *head* value is used to store the number of labels expiring at T_{exp}: $(T_{exp}, (label, counter))$. *label* is the first label that was indexed under T_{exp}. Each of the remaining $c - 1$ labels is stored under a unique index: The ith label's index is (T_{exp}, i), that is, the label's expiration time concatenated with the label's counter at its insertion time.

Algorithm 1. E-ORAM: Write a new label under an expiration time and enumerate all labels indexed under an expiration time.

```
1. Write($\mathtt{E-ORAM : ORAM, T_{exp} : int, lbl : id}$)
2.  $(e, A) := Read_{ORAM}(E-ORAM, T_{exp})$;
3.  if ($e = null$) then
4.      $e' := E_k(lbl, 1)$;
5.      $Write_{ORAM}(T_{exp}, e')$;
6.  else
7.      $(l, c) := D_k(e)$;
8.      $Write_{ORAM}(T_{exp}, E_k(l, c + 1))$;
9.      $Write_{ORAM}((T_{exp}, c + 1), E_k(lbl))$;
10. fi
11. end
12. Enumerate($E-ORAM : ORAM, T_{exp} : int$)
13. $L : id[]$; #store result labels
14. $L := \emptyset$;
15. $(e, A) := Read_{ORAM}(E-ORAM, T_{exp})$;
16. if ($e! = null$) then
17.     $(l, c) := D_k(e)$;
18.     $L := L \cup l$;
19.     for ($i := 2; i \le c; i++$) do
20.         $(e, A) := Read_{ORAM}(E-ORAM, (T_{exp}, i))$;
21.         $l := D_k(e)$;
22.         $L := L \cup l$;
23.     od
24. fi
25. return $L$;
26. end
```

As mentioned in Section 2, $Read_{ORAM}$ denotes the standard ORAM read operation, taking as input the ORAM and a label and returns an element stored under that label along with the list of all elements removed from the ORAM (including the one of interest). $Write_{ORAM}$ is the standard ORAM write operation, which takes as input a label and an element and stores the element indexed under the label. Note that in the standard ORAM implementation, both operations are performed in the same manner. Their operation is only different for the client. Let *ObliviousScramble* be the standard ORAM re-shuffle operation (see Section 2), which takes as input a level id and generates a pseudo-random permutation of the re-encrypted elements at that level. We now present the most important operations for accessing the E-ORAM, Write and Enumerate.

Write(E-ORAM,T_{exp},label). The pseudo-code of this operation is shown in Algorithm 1, lines 1-11. It allows the client to record the fact that *label* expires at time T_{exp}. It first reads the element currently stored under T_{exp} (line 2). If no such element exists (line 3), it generates an element encoding the fact that this label is the first to be stored under T_{exp} (line 4) and writes it on the E-ORAM (line 5). If however a label is already stored under T_{exp} (line 6), retrieve that label l along with the counter c that specifies how many labels are currently expiring (stored in E-ORAM) at T_{exp} (line 7). Note that the read operation performed on line 2 removes this element from the E-ORAM. Then, since now $c + 1$ labels expire at T_{exp}, store label l and the incremented counter in the E-ORAM under T_{exp} (line 8). Finally, store the input *label* under an index consisting of a unique value: T_{exp} concatenated with $c+1$. This will allow the client to later enumerate all labels expiring at T_{exp} (see next).

Enumerate(E-ORAM,T_{exp}). This operation enables C to retrieve all the labels in E-ORAM that expire at T_{exp}. Its pseudo-code is shown in Algorithm 1, lines 12-26. First, initialize the result label list (line 13). Then, read the head label stored under T_{exp} along with the counter of labels expiring at T_{exp} (lines

14,16). If such an element exists (line 15), record the head label (line 17). Then, for each of the $c-1$ ($i = 2, .., c$) remaining labels, retrieve their actual value by reading from E-ORAM the element stored under a unique index consisting of T_{exp} concatenated with i. Note that *Enumerate* removes all labels expiring at T_{exp} from E-ORAM ($Read_{ORAM}$ removes accessed elements).

5.2 Generating Labels

Elements in the standard ORAM model are stored as a pair ($label, value$), where *label* may denote a memory location or the subject of an e-mail. In our case to prevent the server from launching a dictionary attack, we use the a $Label(label, lkey)$ operation to generate labels. Besides the input *label*, *Label* also uses a (random) labeling key, which is used to define a pseudo-random function F_{lkey}. The output of *Label* coincides then with the output of $F_{lkey}(label)$.

In the following we describe the main W-ORAM accessing operations.

5.3 Writing on the Server

Write((W-ORAM,E-ORAM,v,l,T_{exp},params). Insert a value v under a label l, with expiration time T_{exp} on the server (on the W-ORAM and E-ORAM). It takes as input also C's view of *Gen*'s output, $params = k, g, h, G$. Algorithm 2 shows the pseudo-code of this operation. It first generates a new *label* (line 2) and calls *Enc* to produce a W-ORAM tuple (A, B) (line 3). It then generates a non-interactive zero knowledge proof of $QR(A, n) = 0$ (A's quadratic non-residuosity). If the proof verifies (line 5) the server inserts the tuple (A, B) in the top level of the W-ORAM (line 6) and stores *label* under the tuple's expiration time T_{exp} in E-ORAM (see Section 5.1).

Algorithm 2. W-ORAM: Write value v expiring at T_{exp}.

```
1.  Write(W − ORAM : ORAM, E − ORAM : ORAM,
          v : string, l : id, $T_{exp}$ : int)
2.  label := newLabel(l, lkey);
3.  (A, B) := Enc(label, v, $T_{exp}$, params);
4.  ZKP := getQNRProof(A, n);
5.  if (verify(ZKP, A) = 1) then
6.      $T_0$ := getLevel(W − ORAM, 1);
7.      insert($T_0$, (A, B));
8.      Write(E − ORAM, $T_{exp}$, label);
9.  else
10.     return error;
11. fi
12. end
```

Algorithm 3. W-ORAM: Read *label*.

```
1.  Read(W − ORAM : ORAM, label : id)
2.  (R, L) := $Read_{ORAM}$(W − ORAM, label);
3.  U := ($A_u$, $B_u$) := RE(R);
4.  Proof := ZK − POR(L, U);
5.  if (verifyQNR($A_u$, n)
    & verify(Proof, L, U)) then
6.      $T_0$ := getLevel(W − ORAM, 1);
7.      insert($T_0$, U);
8.      return Dec(R, d, k);
9.  else
10.     undo(W − ORAM);
11.     return error;
12. fi end
```

5.4 Reading from the Server

Read(W-ORAM,label). Read takes as input the W-ORAM and a *label* and returns an element of format $(label, x, T_{exp})$. Algorithm 3 shows the pseudo-code for this operation. Read first performs on W-ORAM a standard ORAM read on the desired *label* (line 2). This returns both the W-ORAM element R of interest and the list L of elements (containing R) removed from the W-ORAM. C computes $U = (A_u, B_u)$, a re-encryption of R (line 3) and calls ZK-POR to prove in zero knowledge that U is a re-encryption of the only real element in L (line 4). ZK-POR is described in detail in Section 5.4. S verifies in ZK that $QR(A_u, n) = 0$ and also the validity of the ZK-POR proof. If the proofs are valid (line 5), S inserts U in the first level of the W-ORAM (lines 6-7). C decrypts the desired element R and returns the result (line 8). If any proof fails (line 9) S restores the W-ORAM to the state before the start of Read and returns error (lines 10-11).

Zero Knowledge Proof of ORAM Read. We now present ZK-POR, the zero-knowledge proof of WORM compliance of the read operation performed on the W-ORAM. ZK-POR takes as argument the list L of elements removed from W-ORAM in line 2 of Algorithm 3 and U, the re-encryption of the real element from L. ZK-POR is executed by the client C and the server S. Let m denote the number of levels in the W-ORAM, $m = \log N$.

Let $L = \{(s_1^2, k_1),...,(s_{r-1}^2, k_{r-1}), S(x_r), (s_{r+1}^2, k_{r+1}),.., (s_m^2, k_m)\}$ where the elements are listed in the order in which they were removed from the W-ORAM. C is interested in the item from the rth ORAM layer, $R = S(x_r)$. Let $S(x_r) = (M(x_r)g^{2t_r}, h^{2t_r}) = (A_r, B_r)$. Its first field is a quadratic non-residue. All other elements from L are fakes – their first field is a quadratic residue. Let $U = RE(R) = (M(x_r)g^{2u}, h^{2u}) = (A_u, B_u)$ be the re-encryption of $S(x_r)$. The following steps are executed s times by C and S.

Step 1: Proof Generation. C selects a random permutation $\pi \in_R \mathcal{P}(m)$. C generates $\bar{w} = \{w_1, ..w_m\}$, where each $w_i \in_R \{0,1\}^m$ is odd and generates the proof list $P = \pi(L\bar{w})$. That is, $P = \pi\{(s_1^2 g^{2w_1}, k_1 h^{2w_1}),..,(A_r g^{2w_r}, B_r h^{2w_r}),$ $..,(s_m^2 g^{2w_m}, k_m h^{2w_m})\}$, where, $(A_r g^{2w_r}, B_r h^{2w_r})$ is a re-encryption of $S(x_r)$. C sends P to S. The client locally stores w_i, $i = 1..m$. As assumed in the model, C has $O(\sqrt{N}\log N)$ storage which is sufficient to store $m = O(\log N)$ values.

Step 2: Proof Validation. S flips a coin b. If b is 0, C reveals $w_1, .., w_m$. S verifies that all w_i are odd and $\forall(A_i, B_i) \in L$, $(A_i g^{2w_i}, B_i h^{2w_i}) \in P$. If b is 1, C sends to S the values $s_i g^{w_i}$, $i = 1..m, i \neq r$ along with the value $\Gamma = (t_r + w_r - u)$. Note that given $s_i^2 \ mod \ n$ and n's factorization, C can easily recover s_i. S verifies first that $(s_i g^{w_i})^2$, $i = 1..m, i \neq r$ occurs in the first field of exactly one tuple in P. That is, $m - 1$ of the elements from P are fakes. S then verifies that $(A_r g^{2w_r}, B_r h^{2w_r}) = RE((A_u, B_u), \Gamma)$. If any verification fails, S outputs "error" and stops.

Theorem 2. *A correct execution of Read from W-ORAM has* $O(\log N)$ *complexity.*

Theorem 3. *ZK-POR is a zero knowledge proof system of Read ∈ WORM. That is, Read is WORM compliant.*

Due to lack of space, the proofs will be included in the journal version of the paper.

Note that the soundness property of ZK-POR ensures that a cheating client can remove an element from the ORAM during the Read operation without being detected with probability at most $1/2^s$.

5.5 Shuffling the W-ORAM

When the $l-1$th level of W-ORAM stores more than 4^{l-1}, due to element insertions occurring during Read operation, the level needs to be spilled over into level l. Let $T[l]$ denote the list of elements stored in the W-ORAM at the l-th level. The l-th level then needs to be filled with fakes. The fakes are needed to ensure that subsequent Read accesses will not run out of fakes (see [27] for more details). The l-th level then needs to be obliviously permuted, using only $O(\sqrt{N}\log N)$ client space. Let $T^{new}[l]$ denote the re-shuffled l-th level elements. Due to the WORM semantics, the client also needs to prove that the reshuffle is correct: (i) $T^{new}[l]$ is a re-encryption of the old $T[l]$ and (ii) $|T^{new}[l]| - |T[l]| - |T[l-1]|$ elements from $T^{new}[l]$ are fakes. Shuffle performs this operation.

Algorithm 4. Shuffle of level l.

```
1.Shuffle(W − ORAM : ORAM, l : int)
2.  T^new[l] : string[]#new level l array

#spill T[l − 1] into T[l]
3.  T[l − 1] := getLevel(W − ORAM, l − 1);
4.  T[l] := getLevel(W − ORAM, l);
5.  T[l] := T[l − 1] ∪ T[l];
6.  T[l − 1] := ∅;

#re − encrypt elements from T[l]
7.  for (i := 1; i ≤ |T[l]|; i + +) do
8.      e := T[l][i];
9.      u[i] := genRandom();
10.     T^new[l][i] := E_k(RE(e, u[i]));
11.     for (j := 1; j ≤ s; j + +) do
12.         w[i] := genRandom();
13.         P_j[i] := E_k(RE(e, w[i]),
                "mv", u[i], u[i] − w[i]);
#add fakes
14. f := fakeCount(T[l]);
15. for (i := 1; i ≤ f; i + +) do
16.     (s[i], k[i]) := genRandom();
17.     e := (s[i]^2, k[i]);
18.     append(T^new[l], E_k(e));
19.     for (j := 1; j ≤ s; j + +) do
20.         w[i] := genRandom();
21.         re := RE(e, w[i]),
                "add", s[i]g^w[i], u[i] − w[i]);
22.         append(P_j[i], E_k(re));

#Shuffle T^new[l] and proofs
23. T^new[l] := ObliviousScramble(T^new[l]);
24. for (j := 1; j ≤ s; j + +) do
25.     P_j := ObliviousScramble(P_j);
#decrypt shuffled elements
26. for (i := 1; i ≤ |T^new[l]|; i + +) do
27.     e := T^new[l][i];
28.     T^new[l][i] := D_k(e);
29.     for (j := 1; j ≤ s; j + +) do
30.         e := P_j[i];
31.         (A, B, str, C, D) := D_k(e);
32.         P_j[i] := (A, B, E_k(str, C, D));
#proof verification step
34. for (j := 1; i ≤ s; i + +) do
35.     if (!verify(T[l], T^new[l], P_j)) then
36.         undo(W − ORAM, l − 1, l);
37.         return error;

#commit new level
38. T[l] := T^new[l];
```

Shuffle(W-ORAM,l). This operation takes as input the W-ORAM and the index of its level that needs to be reshuffled. Algorithm 4 shows the pseudo-code of this operation. It first spills the content of level $l-1$ into level l (lines 3-6). Then, it needs to compute the oblivious permutation and build its ZK proof of correctness. We call this procedure ZK-PRS and describe it in detail in the following.

Zero Knowledge Proof of Re-Shuffle. The pseudo-code of ZK-PRS is in Algorithm 4, lines 7-38. Similar to ZK-POR (see Section 5.4), ZK-PRS consists of s rounds executed by the client C and the server S. During each round, a proof list P_j is built by C (line 14 of Algorithm 4). P_j has the same number of elements as $T^{new}[l]$, $O(N)$. The client builds the list $T^{new}[l]$ and each of the s proofs P in the following steps. Initially, $T^{new}[l]$ and each proof list P_j is stored as an empty list at the server S. The client C generates a symmetric key k for the (G, E, D) cryptosystem.

Step 1: Element Re-Encryption. First, C takes each element from $T[l]$ and stores a re-encrypted version in $T^{new}[l]$ and in each proof P_j (lines 7-13). That is, for each element $S_i = (A_i, B_i) \in T[l]$ (stored at S), C generates fresh random odd values $u_i, w_i \in \{0,1\}^{\mathbf{k}}$ (lines 9 and 12) and produces one element S'_i to be inserted in $T^{new}[l]$ (line 10) $S'_i = E_k(A_i g^{2u_i}, B_i h^{2u_i})$ and one element P to be inserted in P_j (line 13) $P = E_k(A_i g^{2w_i}, B_i h^{2w_i}, "mv", \Gamma_1[i], \Gamma_2[i])$ where $\Gamma_1[i] = -w_i$ and $\Gamma_2[i] = (u_i - w_i)$. The string "mv" denotes that this proof element corresponds to an element from $T[l]$ moved to $T^{new}[l]$.

Step 2: Fake Insertion. C adds f fake elements (lines 14-22). For each of the f fakes to be inserted in $T^{new}[l]$, C generates two random values $s_i, k_i \in_R \{0,1\}^{\mathbf{k}}$ (line 16), $i = 1..f$, where w_i is odd. C then adds an element $E_k(s_i^2, k_i)$ in $T^{new}[l]$ (lines 17-18). It then generates a random value $w_i \in_R \{0,1\}^{\mathbf{k}}$ (line 20) for each proof list P_j and appends an element $E_k(s_i^2 g^{2w_i}, k_i h^{2w_i}, "add", \Gamma_1[i], \Gamma_2[i])$ to P_j (lines 21-22). $\Gamma_1[i] = s_i g^{w_i}$, $\Gamma_2[i] = (u_i - w_i) \ mod \ \phi(N)$ and the string "add" denotes that this proof element is a fake added to level l.

Note that $\Gamma_1[i]$ and $\Gamma_2[i]$ are used to keep track of the correspondence between the ith element of each P_j and its re-encryptions in $T[l]$ and $T^{new}[l]$ after the list reshuffle step (see next).

Step 3: List Reshuffle. At the end of the set generation step, C (and S) have a one-to-one correspondence between each element in $T^{new}[l]$, each element in each P_j and each element in $T[l]$. C then calls the *ObliviousScramble* procedure using $T^{new}[l]$ and each P_j as inputs (lines 23-25). During the ObliviousScramble call, elements read from $T^{new}[l]$ and P are decrypted (using k) and re-encrypted before being written back. Due to the semantic security properties of the encryption scheme employed, at the end of the ObliviousScramble, S can no longer map elements from $T[l]$ to elements in the reshuffled $T^{new}[l]$ and P_j sets.

Step 4: Decryption. C reads each element from the reshuffled $T^{new}[l]$ list, decrypts the element and writes it back in-place (lines 26-28). C reads each element from each proof list P_j, decrypts it and writes back $(A_i g^{2w_i}, B_i h^{2w_i}, E_k(str, \Gamma_1[i], \Gamma_2[i]))$, where str is either "mv" or "add" – moved or added fake (lines 29-32).

Step 5: Proof Verification. S verifies each proof list P_j (lines 34-37). If any verification fails, restore the W-ORAM to the state at the beginning of the operation and return error (lines 36-37). Each verification, for a proof list P, works as follows.

S flips a coin b. If $b = 0$, S asks C to prove that P is a valid reshuffle of $T[l]$ and all the remaining elements in P are fakes. For this, C reads each element of P, $(A_i g^{2w_i}, B_i h^{2w_i}, E_k(str, \Gamma_1[i], \Gamma_2[i]))$, retrieves $\Gamma_1[i]$ and sends to S, $A_i g^{2w_i}$, $B_i h^{2w_i}$, str and $\Gamma_1[i]$. If $str =$ "mv", S first verifies that indeed $\Gamma_1[i]$ is an odd number, then verifies that $RE((A_i g^{2w_i}, B_i h^{2w_i}), \Gamma_1[i])$ appears in $T[l]$ exactly once. If $str =$ "add", S verifies that $\Gamma_1[i]^2$ is the first field of exactly one tuple in $T^{new}[l]$. If $b = 1$, C needs to prove that P is a valid reshuffle of $T^{new}[l]$. For this, C reads each element from P, recovers $\Gamma_2[i]$ and sends to S the values $A_i g^{2w_i}$, $B_i h^{w_i}$ and $\Gamma_2[i]$. S verifies that $RE((A_i g^{2w_i}, B_i h^{2w_i}), \Gamma_2[i])$ occurs in $T^{new}[l]$ exactly once.

Algorithm 5. Operation that removes all W-ORAM elements that expire at time T.

```
1.Expire(E − ORAM, W − ORAM : ORAM, T : int
2.  L : id[]; #expiring labels
3.  E : string[]; #removed from W − ORAM
4.  L := Enumerate(E − ORAM, T);
5.  for each label in L do
6.      (R, E) := Read_ORAM(W − ORAM, label);
7.      Proof := ZK − PEE(R, E);
8.      if (verify(Proof, E) = 0) then
9.          undo(W − ORAM);
10.         return error;
11.     fi od
12.end
```

The following result holds. Due to lack of space we omit the proof, which will be included in the journal version of the paper.

Theorem 4. *A correct execution of ZK-PRS has* $O(\log N \log\log N)$ *amortized complexity.*

Due to lack of space we omit the proof, which will be included in the journal version of the paper. The journal version also includes the proof that ZK-PRS is a zero knowledge proof system of Shuffle $\in$ WORM.

5.6 Element Expiration

Expire(T). The operation takes as input a time epoch T and removes all the elements from the W-ORAM that expire in that epoch. The pseudo-code of the operation is shown in Algorithm 5. It first uses the E-ORAM to enumerate all the labels that expire at T (line 4). Then, for each such *label* (line 5) it reads (and removes) from the W-ORAM the corresponding element (line 6). Note that the $Read_{ORAM}$ operation also returns the entire list E of elements removed from the W-ORAM – containing $\log N$ elements. It then builds a zero knowledge proof of the fact that E contains one real element that expires at T and the rest ($\log N - 1$ elements) are fakes (line 7). If the proof verifies, the server accepts the expiration, otherwise restores the W-ORAM to the state before the Read of line 6 (line 9) and returns error (line 10).

Zero Knowledge Proof of Element Expiration. We present a sketch of ZK-PEE, which is called in Algorithm 5, line 7. ZK-PEE takes as input the element to be expired, R and the list of all elements that were removed from W-ORAM when R was read (line 6). Let $m = \log N$ be the number of elements in E. ZK-PEE consists of s rounds run between C and S. During each round C generates $\bar{w} = \{w_1, .., w_m\}$, where $w_i \in_R \{0,1\}^{\mathbf{k}}$ are odd. C computes a proof list $P = \pi(E\bar{w})$, where $\pi \in_R \mathcal{P}_m$ is a random permutation. S then flips a bit b. If $b = 0$, C needs to provide $\bar{w}$. S verifies that all $w_i \in \bar{w}$ are odd and that $P = \pi(E\bar{w})$.

If $b = 1$, C reveals $Dec(R, d, k) = M(x) = (E_k(x), T_{exp}, "real", r)$ to S along with the encryption factor u and the square roots of all the other $\log N - 1$ elements in P. S verifies the revealed element: (i) its correctness, $(M(x)g^{2u}, h^{2u}) = R$ and (ii) its format, that is, $T_{exp} = T$ and the third field is "real". It also verifies that the remaining $(\log N - 1)$ elements of P are fakes.

5.7 Audit

Audit(d,k). The basic auditing operation takes as input the decryption keys d and k. It calls $Dec((A, B), d, k)$, for all elements (A, B) in the ORAM. Once all the elements are recovered, they can be searched for desired keywords.

6 Conclusions

In this paper we introduce WORM-ORAM, a solution that provides WORM compliant Oblivious RAMs. Our solution is based on a set of zero knowledge proofs that ensure that all ORAM operations are WORM compliant. The protocol features the same asymptotic computational complexity as ORAM.

Acknowledgments

We would like to thank Dan Boneh and Peter Williams for early comments and suggestions. We thank the reviewers for their excellent feedback. Sion is supported by the U.S. National Science Foundation through awards CCF 0937833, CNS 0845192, CNS 0708025, IIS 0803197, and CNS 0716608, as well as by grants from CA, Xerox, IBM and Microsoft Research.

References

1. National Association of Insurance Commissioners. Graham-Leach-Bliley Act (1999), http://www.naic.org/GLBA
2. U.S. Dept. of Health & Human Services. The Health Insurance Portability and Accountability Act (HIPAA) (1996), www.cms.gov/hipaa

3. U.S. Public Law 107-347. The E-Government Act (2002)
4. U.S. Public Law No. 107-204, 116 Stat. 745. Public Company Accounting Reform and Investor Protection Act (2002)
5. The U.S. Securities and Exchange Commission. Rule 17a-3&4, 17 CFR Part 240: Electronic Storage of Broker-Dealer Records (2003), http://edocket.access.gpo.gov/
6. The U.S. Department of Defense. Directive 5015.2: DOD Records Management Program (2002), http://www.dtic.mil/whs/directives/corres/pdf/50152std_061902/p50152s.p%df
7. The U.S. Department of Health and Human Services Food and Drug Administration. 21 CFR Part 11: Electronic Records and Signature Regulations (1997), http://www.fda.gov/ora/compliance_ref/part11/FRs/background/pt11finr.p%df
8. The U.S. Department of Education. 20 U.S.C. 1232g; 34 CFR Part 99:Family Educational Rights and Privacy Act (FERPA) (1974), http://www.ed.gov/policy/gen/guid/fpco/ferpa
9. The Enterprise Storage Group. Compliance: The effect on information management and the storage industry (2003), http://www.enterprisestoragegroup.com/
10. Enron email dataset, http://www.cs.cmu.edu/enron/
11. IBM Corp. IBM TotalStorage Enterprise (2007), http://www-03.ibm.com/servers/storage/
12. HP. WORM Data Protection Solutions (2007), http://h18006.www1.hp.com/products/storageworks/wormdps/index.html
13. EMC. Centera Compliance Edition Plus (2007), http://www.emc.com/centera/, http://www.mosaictech.com/pdf_docs/emc/centera.pdf
14. Hitachi Data Systems. The Message Archive for Compliance Solution, Data Retention Software Utility (2007), http://www.hds.com/solutions/data_life_cycle_archiving/achievingregcomp%liance.html
15. Zantaz Inc. The ZANTAZ Digital Safe Product Family (2007), http://www.zantaz.com/
16. StorageTek Inc. VolSafe secure tape-based write once read many (WORM) storage solution (2007), http://www.storagetek.com/
17. Sun Microsystems. Sun StorageTek Compliance Archiving system and the Vignette Enterprise Content Management Suite (White Paper) (2007), http://www.sun.com/storagetek/white-papers/Healthcare_Sun_NAS_Vignette_EHR_080806_Final.p%df
18. Sun Microsystems. Sun StorageTek Compliance Archiving Software (2007), http://www.sun.com/storagetek/management_software/data_protection/comp%liance_archiving/
19. Network Appliance Inc. SnapLock Compliance and SnapLock Enterprise Software (2007), http://www.netapp.com/products/software/snaplock.html
20. Quantum Inc. DLTSage Write Once Read Many Solution (2007), http://www.quantum.com/Products/TapeDrives/DLT/SDLT600/DLTIce/Index.aspx, http://www.quantum.com/pdf/DS00232.p%df
21. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious ram. Journal of the ACM 45, 431–473 (1996)
22. Williams, P., Sion, R., Carbunar, B.: Building Castles out of Mud: Practical Access Pattern Privacy and Correctness on Untrusted Storage. In: ACM Conference on Computer and Communication Security, CCS (2008)

23. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
24. Coull, S., Green, M., Hohenberger, S.: Controlling access to an oblivious database using stateful anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography – PKC 2009. LNCS, vol. 5443, Springer, Heidelberg (2009)
25. Goldreich, O.: Foundations of Cryptography I. Cambridge University Press, Cambridge (2001)
26. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1) (1989)
27. Williams, P., Sion, R., Carbunar, B.: Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In: CCS '08: Proceedings of the 15th ACM conference on Computer and communications security (2008)
28. Boneh, D.: The decision diffie-hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)

Revisiting Unpredictability-Based RFID Privacy Models

Junzuo Lai[1,2], Robert H. Deng[2], and Yingjiu Li[2]

[1] Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200030, China
laijunzuo@sjtu.edu.cn
[2] School of Information Systems,
Singapore Management University, Singapore 178902
{robertdeng,yjli}@smu.edu.sg

Abstract. Recently, there have been several attempts in establishing formal RFID privacy models in the literature. These models mainly fall into two categories: one based on the notion of indistinguishability of two RFID tags, denoted as *ind*-privacy, and the other based on the unpredictability of the output of an RFID protocol, denoted as *unp*-privacy. Very recently, at CCS'09, Ma et al. proposed a modified *unp*-privacy model, referred to as unp'-privacy. In this paper, we first revisit the existing RFID privacy models and point out their limitations. We then propose a new RFID privacy model, denoted as unp^*-privacy, based on the indistinguishability of a real tag and a virtual tag. We provide justification for the new model and formally clarify its relationship with *ind*-privacy model. Finally, we modify Ma et al.'s 2-round RFID protocol to a 3-round mutual authentication RFID protocol and prove that it is of unp^*-privacy.

Keywords: RFID, privacy, security.

1 Introduction

Radio Frequency Identification (RFID) has been widely envisioned as an inevitable replacement of barcodes and other consumer labeling techniques for automatic object identification. An RFID system consists of small devices called RFID tags, one or more RFID readers and a back-end database. Unlike barcodes, each RFID tag records a sufficiently long bitstring to uniquely identify the tag or its bearer. RFID readers communicate with RFID tags using RF signals at a distance from a few inches to several feet. Since RF signals are invisible and penetrating, RFID systems provide a perfect environment for attackers. The prevalence of RFID technologies introduces various serious risks and poses unique security concerns [9,15].

Security problems in RFID systems can be classified into two types. The first is concerned with attacks which aim to wipe out the functioning of the system. The second type, the one which interests us here, is related to privacy. In particular, unauthorized tracking of RFID system users and RFID tag bearers has been recognized as one of the most imperative privacy concerns in the deployments of RFID systems. A privacy-preserving RFID system should therefore provide anonymity (i. e., confidentiality of a tag's identity) as well as unlinkability of the protocol transcripts of a tag [17]. Much attention has been devoted to RFID security, and various schemes have been proposed. The research for secure RFID systems can be mainly categorized into physical technologies [11,5] and protocol-based techniques [21,7,14,19,10,18,6,1]. Juels provides a survey of much of the related literature in [9] and Avoine maintains a current online bibliography at [2]. Nevertheless, most of the existing RFID security research efforts lack formal analysis and mainly offer ad hoc notions of security. In this paper, we are concerned with formal provable privacy models for RFID systems, with a focus on protocol-based techniques.

1.1 Related Work

Avoine [3] first formalizes the adversary model in RFID systems and proposes very general and flexible definitions of RFID privacy. Based on the formal adversary model, Juels and Weis [12] define the notion of strong privacy. The aim of Avoine [3] is to capture a range of adversarial abilities, while Juels and Weis [12] seek to characterize a very strong adversary with a relatively simple definition. In other words, Juels and Weis [12] aim for specificity and simplicity over flexibility. The privacy notion in [12] is based on the indistinguishability of two RFID tags, denoted as *ind*-privacy. However, to our knowledge, there is no RFID protocol that has been *directly* proven to be of *ind*-privacy; On the other hand, if an RFID protocol is not of *ind*-privacy, it can be checked against the *ind*-privacy model easily.

Vaudenay [20] considers side-channel attacks in his RFID privacy model and proposes eight privacy classes which are later consolidated to three by Ng et al. [22]. Paise and Vaudenay [16] extend the definitions in [20] to address mutual authentication. However, the privacy definitions in [20,22,16] contradict reader authentication for any privacy notion that allows tag corruption.

In [8], Ha et al. propose a different privacy model based on the unpredictability of tag outputs, denoted as *unp*-privacy. Unfortunately, this model was later shown to have some deficiencies in its definition [4]. Recently, Ma et al. [13] propose a refined *unp*-privacy model for RFID systems, denoted as unp'-privacy, and investigate the relationship between *ind*-privacy and unp'-privacy.

1.2 Our Contributions

In this paper, we address formal RFID privacy models with the following main contributions:

1. We revisit the unp'-privacy model in [13] and point out its limitation. Specifically, though the unp'-privacy model is robust for 2-round RFID protocols but falls short in dealing with 3-round (i. e., mutual authentication) protocols. We demonstrate this by presenting a 3-round RFID protocol which has a flaw with respect to privacy but can be proven to be of unp'-privacy.

2. We propose a new privacy model, denoted as *unp**-privacy, based on the indistinguishability of a real tag and a virtual tag. We clarify the relationship between the *ind*-privacy and the *unp**-privacy by formally proving that the former is weaker than the latter. To understand which level of privacy an RFID system provides, it is critical to clarify the relationship between the privacy notions.
3. We modify and extend the RFID protocol in [13] to 3-round mutual authentication protocol and show that it is of *unp**-privacy.

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we briefly discuss the formal definitions for the *ind*-privacy and the *unp*-privacy models. We revisit and re-examine the *unp*$'$-privacy model in Section 3. In Section 4, we introduce our privacy model, establish its relation with the *ind*-privacy model and show that an improved version of the protocol in [13] is of *unp**-privacy. We conclude in Section 5.

2 Preliminaries

If S is a set, then $s \in_R S$ indicates that s is chosen uniformly at random from S. If $x_1, x_2, \ldots$ are strings, then $x_1 \| x_2 \| \ldots$ denotes their concatenation. Let $y \leftarrow \mathcal{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_n}(x_1, x_2, \ldots)$ denote that y be assigned with the output of the algorithm $\mathcal{A}$ which takes $x_1, x_2, \ldots$ as inputs and has accesses to oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$.

2.1 Pseudorandom Functions

A pseudorandom function is a family of functions with the property that the input-output behavior of a random instance of the family is "computationally indistinguishable" from that of a random function. Let $F : \mathsf{Keys}(F) \times D \to R$ be a family of functions and let $\mathsf{Rand}^{D \to R}$ be the family of all functions with domain D and range R, where $\mathsf{Keys}(F)$ is the set of keys (or indexes) of F. Consider the following game between an attack algorithm $\mathcal{A}$ and a challenger.

```
**Game**$_{\mathcal{A}}^{prf}$
  $\triangleright$ $\beta \in_R \{0,1\}$;
  $\triangleright$ If $\beta = 0$ then $g \in_R \mathsf{Rand}^{D \to R}$, else $g \in_R F$;
  $\triangleright$ $\beta' \leftarrow \mathcal{A}^g$.
```

Throughout the game, we assume that $\mathcal{A}$ makes at most q oracle queries. We define $\mathcal{A}$'s advantage in the above game as

$$Adv_{\mathcal{A}}(q) = \left| \mathsf{Pr}[\beta = \beta'] - \frac{1}{2} \right| .$$

Definition 1. *An adversary $\mathcal{A}$ (t, q, ϵ)-breaks the pseudorandomness of the function family F if the advantage $Adv_{\mathcal{A}}(q)$ of $\mathcal{A}$ in the above game is at least ϵ and the running time of $\mathcal{A}$ is at most t.*

Definition 2. *A function family F is said to be (t, q, ϵ)-pseudorandom if there exists no adversary who can (t, q, ϵ)-break the pseudorandomness of F.*

2.2 An RFID System Model

Without loss of generality, we assume a fixed, polynomial-size tag set $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, a reader $\mathcal{R}$ and a back-end database $\mathcal{DB}$ as the elements of our RFID system, denoted as $\mathcal{S} = \{\mathcal{T}, \mathcal{R}, \mathcal{DB}\}$. Typically, each tag is a passive transponder identified by a unique ID and has only limited memory which can be used to store several keys and/or state information. The reader $\mathcal{R}$ is composed of one or more transceivers and a processing subsystem. The database $\mathcal{DB}$ maintains $\mathcal{T}$'s authentication data such as tag IDs, secret keys, states and session identifiers. Communications between $\mathcal{R}$ and $\mathcal{T}$ take place over an insecure air interface, while communications between $\mathcal{R}$ and $\mathcal{DB}$ are assumed to be over a secure channel.

In addition, the RFID system $\mathcal{S}$ includes a tuple of algorithms described bellow.

$\mathsf{Initialize}(\kappa)$: It takes as input a security parameter κ, generates key k_i for each tag $\mathcal{T}_i \in \mathcal{T}$ and sets the tag's initial state; it also associates $\mathcal{T}_i$ with its unique ID_i and setups the back-end database $\mathcal{DB}$ for $\mathcal{R}$ to store necessary information for tag identification.

$\mathsf{ReaderStart}()$: It invokes $\mathcal{R}$ to output a new session identifier sid and the first protocol message m_1 of the session.

$\mathsf{TagCompute}(sid, m_1, \mathcal{T}_i)$: It takes as input a session identifier sid, a protocol message m_1 and $\mathcal{T}_i$, outputs a message m_2. This algorithm is run by $\mathcal{T}_i$.

$\mathsf{ReaderCompute}(sid, m_2)$: It takes as input a session identifier sid and a protocol message m_2, outputs a protocol message m_3. This algorithm is run by $\mathcal{R}$.

$\mathsf{Execute}(\mathcal{R}, \mathcal{T}_i)$: It takes as input $\mathcal{R}$ and $\mathcal{T}_i$, runs the interactive authentication protocol between $\mathcal{R}$ and $\mathcal{T}_i$ and outputs the entire protocol transcript. For the three-round canonical RFID protocol shown in Fig 1, we have

$$(m_1, m_2, m_3) \leftarrow \mathsf{Execute}(\mathcal{R}, \mathcal{T}_i),$$

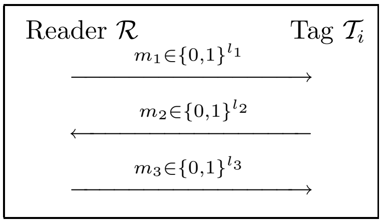


Fig. 1. The canonical RFID Protocol

where $(sid, m_1) \leftarrow \mathsf{ReaderStart}(), m_2 \leftarrow \mathsf{TagCompute}(sid, m_1, \mathcal{T}_i)$ and $m_3 \leftarrow \mathsf{ReaderCompute}(sid, m_2)$.

2.3 Adversaries

An adversary $\mathcal{A}$ is a probabilistic polynomial time (PPT) algorithm and is assumed to have complete control over all communications between $\mathcal{R}$ and $\mathcal{T}$. The interaction between $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary's capabilities in a real attack. In the following, we specify oracles $\mathcal{A}$ is permitted to query.

Launch($\mathcal{R}$): It invokes $\mathcal{R}$ to start a session of the protocol and responds with a session id sid and the first protocol message m_1.
SendTag($sid, m_1', \mathcal{T}_i$): It invokes $\mathcal{T}_i$ and responds with a protocol message m_2.
SendReader(sid, m_2'): It invokes $\mathcal{R}$ and responds with a protocol message m_3.
Reveal($\mathcal{T}_i$): It invokes $\mathcal{T}_i$ and returns the tag's current secret key and internal state.

Queries to **SendTag** and **SendReader** model active attacks, in which the adversary may tamper with the message being sent over the insecure RF channel. Queries to **Reveal** model the leakage of tags' secret information.

Let $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ and $\mathcal{O}_4$ denote **Launch**, **SendTag**, **SendReader** and **Reveal** oracles, respectively. All privacy models in this paper are defined using a game between an adversary $\mathcal{A}$ and a challenger. Throughout a game, we assume that $\mathcal{A}$ is allowed to launch $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ and $\mathcal{O}_4$ oracle queries without exceeding q_{ini}, q_{st}, q_{sr} and q_{rv} overall calls, respectively.

2.4 The *ind*-Privacy and *unp*-Privacy Models

2.4.1 The *ind*-Privacy Model

Juels and Weis [12] present an indistinguishability-based RFID privacy model which is reminiscent of the classic indistinguishability under chosen-plaintext attack (IND-CPA) and under chosen-ciphertext attack (IND-CCA) cryptosystem security.

Figure 2 illustrates the *ind*-privacy game $\mathbf{Game}^{ind}_{\mathcal{A}}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$, in which $\mathcal{A}$ is comprised of a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$. The game proceeds as follows. At first, the challenger initializes the RFID system $\mathcal{S}$ by producing a reader $\mathcal{R}$ and a set of tags $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_n\}$ according to the security parameter κ. Then, $\mathcal{A}_1$ issues $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ and $\mathcal{O}_4$ oracle queries, and outputs two uncorrupted tags $\{\mathcal{T}_i, \mathcal{T}_j\}$ (i.e., tags to which no **Reveal** queries have been issued) as challenge candidates. It also outputs a state information st which will be transmitted to algorithm $\mathcal{A}_2$. One of the two candidates $\mathcal{T}_c$ is then selected based on the value of a random bit and presented to $\mathcal{A}$ (effectively as a tag oracle). $\mathcal{A}_2$ is allowed to query $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ and $\mathcal{O}_4$ oracles on $\mathcal{R}$, $\mathcal{T}_c$ and the tag set $\mathcal{T}' = \mathcal{T} - \{\mathcal{T}_i, \mathcal{T}_j\}$ with the restriction that it cannot query **Reveal**($\mathcal{T}_c$). Finally, $\mathcal{A}_2$ is asked to guess the random bit.

```
$\textbf{Game}_{\mathcal{A}}^{ind}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$
  $\triangleright$ Setup the reader $\mathcal{R}$ and a set of tags $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$;
  $\triangleright$ $\{\mathcal{T}_i, \mathcal{T}_j, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3,\mathcal{O}_4}(\mathcal{R}, \mathcal{T})$;
  $\triangleright$ Set $\mathcal{T}' = \mathcal{T} - \{\mathcal{T}_i, \mathcal{T}_j\}$;
  $\triangleright$ $\beta \in_R \{0, 1\}$;
  $\triangleright$ If $\beta = 0$ then $\mathcal{T}_c = \mathcal{T}_i$, else $\mathcal{T}_c = \mathcal{T}_j$;
  $\triangleright$ $\beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3,\mathcal{O}_4}(\mathcal{R}, \mathcal{T}', \mathcal{T}_c, st)$.
```

Fig. 2. The *ind*-Privacy Game

Definition 3. *The advantage of an adversary $\mathcal{A}$ in the above game is defined as*

$$Adv_{\mathcal{A}}^{ind}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}) = |Pr[\beta' = \beta] - \frac{1}{2}|,$$

where the probability is taken over the choice of tag set $\mathcal{T}$ and the coin tosses of $\mathcal{A}$.

Definition 4. *An adversary $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the ind-privacy of an RFID system $\mathcal{S}$ if the advantage $Adv_{\mathcal{A}}^{ind}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$ of $\mathcal{A}$ in the above game is at least ϵ and the running time of $\mathcal{A}$ is at most t.*

Definition 5. *An RFID system $\mathcal{S}$ is said to be $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-ind-privacy if there exists no adversary who can $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-break the ind-privacy of $\mathcal{S}$.*

2.4.2 The *unp*-Privacy Model

The goal of the adversary in the above *ind*-privacy game is to distinguish two different tags within its computational power and parameters. The idea is intuitively appealing; however, the *ind*-privacy model is difficult to apply *directly* in proving given a protocol is of *ind*-privacy. Juels and Weis [12] only prove the *ind*-privacy of a simple randomized hash-lock RFID protocol. To our knowledge, no mutual authentication RFID protocol has been proven *directly* to be of *ind*-privacy. Ha et al. [8] propose a different privacy model based on the unpredictability of tag outputs, denoted as *unp*-privacy. In fact, Juels and Weis [12] prove the *ind*-privacy of the randomized hash-lock RFID protocol by showing that no adversary can distinguish the real output of a tag from a random value. In other words, Juels and Weis [12] in fact prove the *unp*-privacy of the randomized hash-lock RFID protocol.

Figure 3 depicts the *unp*-privacy game $\textbf{Game}_{\mathcal{A}}^{unp}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$, in which an adversary $\mathcal{A}$ is comprised of a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$. At first, a challenger initializes the RFID system by producing a reader $\mathcal{R}$ and a set of tags $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_n\}$ according to the security parameter κ. Then, $\mathcal{A}_1$ issues $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ and $\mathcal{O}_4$ oracle queries, and outputs an uncorrupted tag $\mathcal{T}_c$ as the challenge tag. It also outputs a state information st which will be transmitted to algorithm $\mathcal{A}_2$. Next, the challenger selects a random bit β and sends m_2^* to

$\mathcal{A}_2$, where m_2^* is taken from $(m_1^*, m_2^*, m_3^*) \leftarrow \mathbf{Execute}(\mathcal{R}, \mathcal{T}_c)$ if $\beta = 1$, and $m_2^* \in_R \{0,1\}^{l_2}$ otherwise. Finally, $\mathcal{A}_2$ is asked to guess the random bit. Note that $\mathcal{A}_2$ is not allowed to query any oracle.

```
$\mathbf{Game}_{\mathcal{A}}^{unp}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$
  $\triangleright$ Setup the reader $\mathcal{R}$ and a set of tags $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$;
  $\triangleright$ $\{\mathcal{T}_c, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3,\mathcal{O}_4}(\mathcal{R}, \mathcal{T})$;
  $\triangleright$ $\beta \in_R \{0,1\}$;
  $\triangleright$ If $\beta$=1 then $m_2^*$ is taken from $(m_1^*, m_2^*, m_3^*) \leftarrow \mathbf{Execute}(\mathcal{R}, \mathcal{T}_c)$,
    else $m_2^* \in_R \{0,1\}^{l_2}$;
  $\triangleright$ $\beta' \leftarrow \mathcal{A}_2(m_2^*, st)$.
```

Fig. 3. The *unp*-Privacy Game

Definition 6. *The advantage of an adversary $\mathcal{A}$ in the above game is defined as*

$$Adv_{\mathcal{A}}^{unp}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}) = |Pr[\beta' = \beta] - \frac{1}{2}|,$$

where the probability is taken over the choice of tag set $\mathcal{T}$ and the coin tosses of $\mathcal{A}$.

Definition 7. *An adversary $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the unp-privacy of RFID system $\mathcal{S}$ if the advantage $Adv_{\mathcal{A}}^{unp}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$ of $\mathcal{A}$ in the above game is at least ϵ and the running time of $\mathcal{A}$ is at most t.*

Definition 8. *An RFID system $\mathcal{S}$ is said to be $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-unp-privacy if there exists no adversary who can $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-break the unp-privacy of $\mathcal{S}$.*

3 The *unp′*-Privacy Model, Revisited

Note that in the *unp*-privacy game, the adversary $\mathcal{A}_2$ does not get the full transcript of the protocol execution between the reader and the challenge tag, but only m_2^* which is either a random message or the message sent by the tag. As a result, an RFID protocol may have known weakness in privacy but can be shown to be of *unp*-privacy, as confirmed by Deursen and Radomirović [4]. At CCS'09, Ma et al. [13] propose an improved *unp*-privacy model, denoted as *unp′*-privacy. In the *unp′*-privacy model, the adversary is given not only m_2^*, but also the last message m_3^* of the protocol. The *unp′*-privacy model is robust for 2-round RFID protocols, as demonstrated in [13]; however, we will show in this section that the model has a deficiency when applied to 3-round protocols.

3.1 The Model

Figure 4 presents the *unp*′-privacy game $\mathbf{Game}_{\mathcal{A}}^{unp'}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$, in which an adversary $\mathcal{A}$ is comprised of a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$. At the start of the game, a challenger initializes the RFID system by producing $\mathcal{R}$ and $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_n\}$ according to the security parameter κ. Then, $\mathcal{A}_1$ issues $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ and $\mathcal{O}_4$ oracle queries, and outputs an uncorrupted challenge tag $\mathcal{T}_c$ and a message m_1^*. It also outputs a state information *st* which will be transmitted to algorithm $\mathcal{A}_2$. Next, the challenger selects a random bit β and sends (m_2^*, m_3^*) to $\mathcal{A}_2$, where $(m_1^*, m_2^*, m_3^*) \leftarrow \mathbf{Execute}(\mathcal{R}, \mathcal{T}_c)$ if $\beta = 1$, and $(m_2^*, m_3^*) \in_R \{0,1\}^{l_2} \times \{0,1\}^{l_3}$ otherwise. Finally, $\mathcal{A}_2$ has oracle accesses to tags except $\mathcal{T}_c$ and is required to infer the value of β.

```
$\mathbf{Game}_{\mathcal{A}}^{unp'}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$
  $\triangleright$ Setup the reader $\mathcal{R}$ and a set of tags $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$;
  $\triangleright$ $\{\mathcal{T}_c, m_1^*, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3,\mathcal{O}_4}(\mathcal{R}, \mathcal{T})$;
  $\triangleright$ Set $\mathcal{T}' = \mathcal{T} - \{\mathcal{T}_c\}$;
  $\triangleright$ $\beta \in_R \{0,1\}$;
  $\triangleright$ If $\beta = 1$ then $(m_1^*, m_2^*, m_3^*) \leftarrow \mathbf{Execute}(\mathcal{R}, \mathcal{T}_c)$,
    else $(m_2^*, m_3^*) \in_R \{0,1\}^{l_2} \times \{0,1\}^{l_3}$;
  $\triangleright$ $\beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3,\mathcal{O}_4}(\mathcal{R}, \mathcal{T}', m_2^*, m_3^*, st)$.
```

Fig. 4. The *unp*′-Privacy Game

Definition 9. *The advantage of an adversary $\mathcal{A}$ in the above game is defined as*

$$Adv_{\mathcal{A}}^{unp'}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}) = |Pr[\beta' = \beta] - \frac{1}{2}|,$$

where the probability is taken over the choice of tag set $\mathcal{T}$ and the coin tosses of $\mathcal{A}$.

Definition 10. *An adversary $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the unp′-privacy of RFID system $\mathcal{S}$ if the advantage $Adv_{\mathcal{A}}^{unp'}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$ of $\mathcal{A}$ in the above game is at least ϵ and the running time of $\mathcal{A}$ is at most t.*

Definition 11. *An RFID system $\mathcal{S}$ is said to be $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-unp′-privacy if there exists no adversary who can $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-break the unp′-privacy of $\mathcal{S}$.*

3.2 A Counterexample

Ma et al. [13] introduce an efficient 2-round protocol and prove that it is of *unp*′-privacy. We now modify the protocol to a 3-round mutual authentication protocol and show that the new protocol has clear weakness with respect to privacy but can be proven to be of *unp*′-privacy. This example therefore exposes a deficiency of the *unp*′-privacy model when it is applied to 3-round mutual authentication protocols.

Let $F : \{0,1\}^{l_k} \times \{0,1\}^{l_d} \to \{0,1\}^{l_r}$ be a PRF family. Let $ctr \in \{0,1\}^{l_r}$ be a counter, $pad_1 \in \{0,1\}^{l_{p1}}$ and $pad_2 \in \{0,1\}^{l_{p2}}$ be two paddings such that $l_r + l_{p1} = l_d$. The RFID system is constructed as follows.

Initialize(κ): It randomly chooses a key $k_i \in \{0,1\}^{l_k}$ for each tag $\mathcal{T}_i \in \mathcal{T}$. $\mathcal{T}_i$ stores k_i, a counter $ctr_i \in \{0,1\}^{l_r}$, and a 1-bit flag s_i in its memory. Initially, $ctr_i = 1$ and $s_i = 0$. It also associates $\mathcal{T}_i$ with a unique ID_i, and stores the tuple $(I_i, k_i, ctr_i, \mathsf{ID}_i)$ in the back-end database $\mathcal{DB}$, where $I_i = F_{k_i}(ctr_i \| pad_1)$.

Execute($\mathcal{R}, \mathcal{T}_i$): $\mathcal{R}$ first sends a challenge $c \in_R \{0,1\}^{l_c}$ to $\mathcal{T}_i$, where $l_c + l_r + l_{p2} = l_d$. Upon receiving c, $\mathcal{T}_i$ executes the following steps:

1. Randomly choose $r_2 \in \{0,1\}^{l_{p2}}$ and compute $I_i = F_{k_i}(ctr_i \| pad_1)$;
2. Set $r_1 = F_{k_i}(c \| I_i \| pad_2) \oplus ctr_i$ if $s_i = 0$, else set $r_1 = F_{k_i}(c \| I_i \| r_2) \oplus ctr_i$;
3. Respond with $(r_1 \| I_i, r_2)$, increment ctr_i by 1 and set $s_i = 1$.

Upon receiving the response $(r_1 \| I_i, r_2)$, $\mathcal{R}$ identifies the tag from its database as follows:

1. Search for the tuple $(I_i, k_i, ctr'_i, \mathsf{ID}_i)$ using I_i as an index. If such a tuple exists, compute $F_{k_i}(c \| I_i \| pad_2)$ and then
   (a) If $ctr'_i = F_{k_i}(c \| I_i \| pad_2) \oplus r_1$, update $ctr'_i = ctr'_i + 1$ and $I_i = F_{k_i}(ctr'_i \| pad_1)$, respond with $f = F_{k_i}(c \| ctr'_i \| r_2)$ and accept the tag;
   (b) Else abort the protocol.
2. Else look up the database for a tuple $(I'_i, k_i, ctr'_i, \mathsf{ID}_i)$ in an exhaustive search such that $ctr_i = F_{k_i}(c \| I_i \| r_2) \oplus r_1$ and $F_{k_i}(ctr_i \| pad_1) = I_i$. Then
   (a) If such a tuple exists, update $ctr'_i = ctr_i + 1$ and $I'_i = F_{k_i}(ctr'_i \| pad_1)$, respond with $f = F_{k_i}(c \| ctr'_i \| r_2)$ and accept the tag;
   (b) Else abort the protocol.

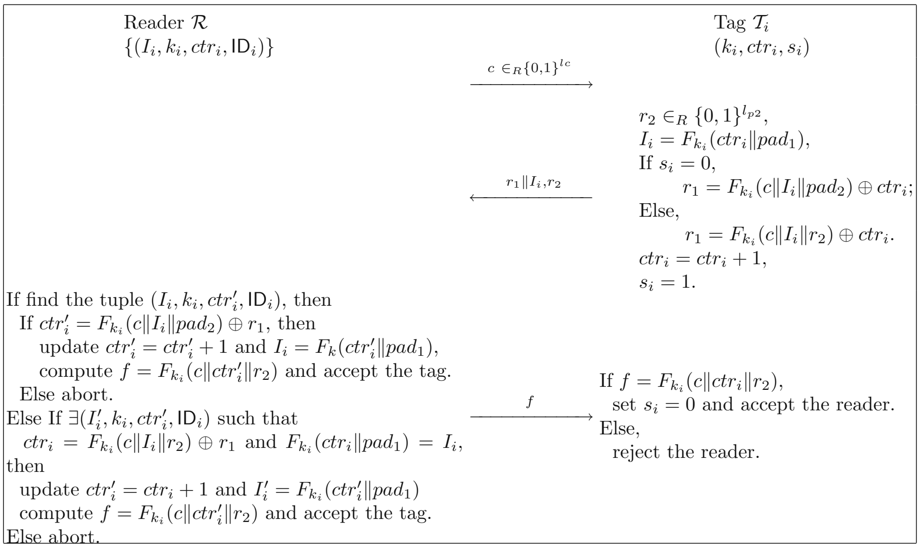
Upon receiving f, $\mathcal{T}_i$ checks whether $f = F_{k_i}(c \| ctr_i \| r_2)$. If not, $\mathcal{T}_i$ rejects the reader. Else, $\mathcal{T}_i$ sets $s_i = 0$ and accepts the reader.

Note that the ReaderStart, TagCompute and ReaderCompute algorithms are not shown explicitly in the above description since they are embedded in the Execute algorithm. The protocol is depicted in 5.

A flaw of the protocol is that an active attacker can find out whether a tag's state is $s = 0$ or $s = 1$. If a tag is in state $s = 0$, the reader does not verify the integrity of r_2; while if the tag is in state $s = 1$, this verification occurs implicitly. Note that under normal circumstances tags will be in state $s = 0$. Hence, an active attacker can flag a tag by setting its state to $s = 1$ and trace the tag in subsequent protocol sessions. However, the following theorem states that the protocol is of *unp*′-privacy.

Theorem 1. *The above mutual authentication RFID protocol is of* $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$*-unp′-privacy, assuming the function family* $F : \{0,1\}^{l_k} \times \{0,1\}^{l_d} \to \{0,1\}^{l_r}$ *is* (t', q, ϵ')*-pseudorandom, where*

$$t' \approx t, \ q \approx q_{st} + q_{sr}, \ \epsilon' = \epsilon/n.$$



Reader $\mathcal{R}$ $\{(I_i, k_i, ctr_i, \mathsf{ID}_i)\}$

Tag $\mathcal{T}_i$ (k_i, ctr_i, s_i)

$\xrightarrow{c \in_R \{0,1\}^{l_c}}$

$r_2 \in_R \{0,1\}^{l_{p2}}$,
$I_i = F_{k_i}(ctr_i \| pad_1)$,
If $s_i = 0$,
  $r_1 = F_{k_i}(c \| I_i \| pad_2) \oplus ctr_i$;
Else,
  $r_1 = F_{k_i}(c \| I_i \| r_2) \oplus ctr_i$.
$ctr_i = ctr_i + 1$,
$s_i = 1$.

$\xleftarrow{r_1 \| I_i, r_2}$

If find the tuple $(I_i, k_i, ctr'_i, \mathsf{ID}_i)$, then
  If $ctr'_i = F_{k_i}(c \| I_i \| pad_2) \oplus r_1$, then
    update $ctr'_i = ctr'_i + 1$ and $I_i = F_k(ctr'_i \| pad_1)$,
    compute $f = F_{k_i}(c \| ctr'_i \| r_2)$ and accept the tag.
  Else abort.
Else If $\exists (I'_i, k_i, ctr'_i, \mathsf{ID}_i)$ such that
  $ctr_i = F_{k_i}(c \| I_i \| r_2) \oplus r_1$ and $F_{k_i}(ctr_i \| pad_1) = I_i$, then
  update $ctr'_i = ctr_i + 1$ and $I'_i = F_{k_i}(ctr'_i \| pad_1)$
  compute $f = F_{k_i}(c \| ctr'_i \| r_2)$ and accept the tag.
Else abort.

$\xrightarrow{f}$

If $f = F_{k_i}(c \| ctr_i \| r_2)$,
  set $s_i = 0$ and accept the reader.
Else,
  reject the reader.

Fig. 5. The Counterexample RFID Protocol

Proof. Suppose there exists an adversary $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the *unp*$'$-privacy of the RFID protocol in Figure 5. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to (t', q, ϵ')-break the pseudorandomness of the function family F.

$\mathcal{B}$ is provided oracle access to a function g and tries to decide if g is drawn at random from F, namely $g \in_R F$ (which means that a key is chosen via $k \in_R \{0,1\}^{l_k}$ and then g is set to F_k.), or is drawn at random from $\mathsf{Rand}^{\{0,1\}^{l_d} \to \{0,1\}^{l_r}}$, namely $g \in_R \mathsf{Rand}^{\{0,1\}^{l_d} \to \{0,1\}^{l_r}}$. $\mathcal{B}$'s goal is to output 0 if $g \in_R \mathsf{Rand}^{\{0,1\}^{l_d} \to \{0,1\}^{l_r}}$ and 1 otherwise. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and proceeds as follows.

Setup $\mathcal{B}$ randomly chooses an index $j \in \{1, \ldots, n\}$. Without loss of generality, we assume $j = n$. $\mathcal{B}$ then randomly chooses a key $k_i \in \{0,1\}^{l_k}$ for each tag $\mathcal{T}_i \in \{\mathcal{T}_1, \ldots, \mathcal{T}_{n-1}\}$. $\mathcal{T}_i$ stores k_i, a counter $ctr_i \in \{0,1\}^{l_r}$, and a 1-bit flag s_i. Initially, $ctr_i = 1$ and $s_i = 0$. $\mathcal{B}$ associates $\mathcal{T}_i$ with a unique ID_i, and stores the tuple $(I_i, k_i, ctr_i, \mathsf{ID}_i)$ in the database $\mathcal{DB}$, where $I_i = F_{k_i}(ctr_i \| pad_1)$.
$\mathcal{B}$ associates tag $\mathcal{T}_n$ with a unique ID_n. $\mathcal{T}_n$ keeps a counter ctr_n and a 1-bit flag s_n with initially values 1 and 0, respectively. *Note that, the key k_n of $\mathcal{T}_n$ is unknown to $\mathcal{B}$.* $\mathcal{B}$ queries its oracle on $ctr_n \| pad_1$ and gets the response of the oracle I_n. $\mathcal{B}$ stores the tuple $(I_n, *, ctr_n, \mathsf{ID}_n)$ in the database $\mathcal{DB}$.
In the following, we let $x = ctr_n \| pad_1$, $m'_1 \| I_n \| pad_2$ or $m'_1 \| I_n \| r_2$ depending on the context. The basic idea is that $\mathcal{B}$ queries its oracle g on x and gets either $F_{k_n}(x)$ or a random message as response .

Query phase 1 $\mathcal{A}$ issues **Launch**, **SendTag**, **SendReader** and **Reveal** queries to which $\mathcal{B}$ answers as follows:

- **Launch** query on $\mathcal{R}$: $\mathcal{B}$ responds according to the protocol.
- **SendTag** query on $(sid, m_1', \mathcal{T}_i)$: Respond according to the protocol. Note that $\mathcal{B}$ can do it for $i \in \{1, \ldots, n-1\}$ because $\mathcal{B}$ knows the keys and the internal information of $\mathcal{T}_i$. For $\mathcal{T}_n$, $\mathcal{B}$ also can do it by querying its oracle on x whenever it needs to compute $F_{k_n}(x)$.
- **SendReader** query on (sid, m_2'): Respond according to the protocol. Whenever $\mathcal{B}$ needs to compute $F_{k_n}(x)$, $\mathcal{B}$ queries its oracle on x.
- **Reveal** query on $\mathcal{T}_i$: If $\mathcal{T}_i = \mathcal{T}_n$, abort and randomly output a bit; else, forward the key k_i and internal state (ctr_i, s_i) of $\mathcal{T}_i$ to $\mathcal{A}$.

Challenge $\mathcal{A}$ submits a message $m_1 \in \{0,1\}^{l_c}$ and an uncorrupted challenge tag $\mathcal{T}_c$ to $\mathcal{B}$ which proceeds as follows:

- If $T_c \neq T_n$, abort and randomly output a bit.
- Else, randomly choose $r_2 \in \{0,1\}^{l_{p2}}$.
- Set $x = ctr_n \| pad_1$, query its oracle on x and get the response I_n.
- If $s_n = 0$, query its oracle on $x = m_1 \| I_n \| pad_2$, get the response y and set $r_1 = y \oplus ctr_n$; else query its oracle on $x = m_1 \| I_n \| r_2$, get the response y and set $r_1 = y \oplus ctr_n$.
- Set $m_2 = (r_1 \| I_n, r_2)$, update $ctr_n = ctr_n + 1$ and $s_n = 0$.
- Query its oracle on $m_1 \| ctr_n \| r_2$, get the response m_3, and send (m_2, m_3) to $\mathcal{A}$.

Query phase 2 Let $\mathcal{T}'$ denote the tag set $\mathcal{T} - \mathcal{T}_c = \{\mathcal{T}_1, \ldots, \mathcal{T}_{n-1}\}$. $\mathcal{A}$ continues to issue **Launch**, **SendTag**, **SendReader** and **Reveal** queries. $\mathcal{B}$ answers them in as follows:

- **Launch** query on $\mathcal{R}$: Respond according to the protocol.
- **SendTag** query on $(sid, m_1', \mathcal{T}_i \in \mathcal{T}')$: Respond according to the protocol. $\mathcal{B}$ can do it because it knows the keys and the internal information of the tags in $\mathcal{T}'$.
- **SendReader** query on (sid, m_2'): Respond according to the protocol. $\mathcal{B}$ can do it because it knows $\mathcal{DB}$.
- **Reveal** query on $\mathcal{T}_i \in \mathcal{T}'$: Forward $\mathcal{T}_i$'s key k_i and internal state (ctr_i, s_i) to $\mathcal{A}$.

Guess $\mathcal{A}$ outputs a bit β' which $\mathcal{B}$ also takes as its output.

If $\mathcal{B}$ does not abort during the simulation, $\mathcal{B}$'s simulation is perfect and are identically distributed as the real one from the construction. It is obvious that the probability that $\mathcal{B}$ does not abort during the simulation is $1/n$. In the simulation, $\mathcal{B}$ needs to query its oracle in response to $\mathcal{A}$'s **SendTag** and **SendReader** queries. So, $q \approx q_{st} + q_{sr}$. The running time of $\mathcal{B}$ is approximately that of $\mathcal{A}$. This completes the proof.

4 Our Model and Results

The limitation in the definition of the *unp*′-privacy model, as shown in the above example, is due to the constraint imposed on the adversary $\mathcal{A}_2$, i. e., $\mathcal{A}_2$ only has access to m_2^* and m_3^* as supplied by the challenger and is not allowed to query oracles on the challenge tag $\mathcal{T}_c$. In this section, we propose a new RFID privacy model, denoted as *unp**-privacy, as a remedy to this problem.

The intuition of the *unp**-privacy model is that no adversary should be able to distinguish the output of a real tag from that of a virtual tag, which is defined as a tag without any secret information. This implies that no adversary can link a real tag and its behavior without learning its secret key. We emphasis that our *unp**-privacy model does not impose any restrictions on the number of rounds in an RFID protocol. In what follows, we introduce the *unp**-privacy model, investigate the relationship between this new model and the *ind*-privacy model. We also extend the 2-round RFID protocol in [13] to a 3-round mutual authentication protocol and show that it is of *unp**-privacy.

4.1 The *unp**-Privacy Model

Figure 6 illustrates the *unp**-privacy game $\mathbf{Game}_{\mathcal{A}}^{unp^*}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$ between an adversary $\mathcal{A}$ and a challenger, in which $\mathcal{A}$ consists of a pair of algorithms $(\mathcal{A}_1, \mathcal{A}_2)$. The challenger initializes the RFID system $\mathcal{S}$ by producing a reader $\mathcal{R}$ and a set of tags $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_n\}$ according to the security parameter κ. Then, $\mathcal{A}_1$ issues $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ and $\mathcal{O}_4$ oracle queries, and outputs an uncorrupted challenge tag $\mathcal{T}_c$. It also outputs a state information *st* which will be transmitted to algorithm $\mathcal{A}_2$. Next, the challenger selects a random bit β. Finally, $\mathcal{A}_2$ is asked to guess the value of the random bit. $\mathcal{A}_2$ is allowed to query $\mathcal{O}_1, \mathcal{O}_2$ and $\mathcal{O}_3$ oracles on $\mathcal{R}$ and $\mathcal{T}_c$. The challenger responds to $\mathcal{A}_2$ queries as follows:

– **Launch** query on $\mathcal{R}$: If $\beta = 0$, generate a new session identifier *sid*, randomly choose $m_1 \in \{0,1\}^{l_1}$ and forward (sid, m_1) to $\mathcal{A}_2$; else, run the algorithm $\mathsf{ReaderStart}$, and forward the result to $\mathcal{A}_2$.
– **SendTag** query on $(sid, m_1', \mathcal{T}_c)$: If $\beta = 0$, randomly choose $m_2 \in \{0,1\}^{l_2}$ and forward m_2 to $\mathcal{A}_2$; else, run the algorithm $\mathsf{TagCompute}(sid, m_1', \mathcal{T}_c)$ and forward the result to $\mathcal{A}_2$.
– **SendReader** query on input (sid, m_2'): If $\beta = 0$, randomly choose $m_3 \in \{0,1\}^{l_3}$ and forward m_3 to $\mathcal{A}_2$; else, run the algorithm $\mathsf{ReaderCompute}(sid, m_3)$ and forward the result to $\mathcal{A}_2$.

Definition 12. *The advantage of an adversary $\mathcal{A}$ in the above game is defined as*

$$Adv_{\mathcal{A}}^{unp^*}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}) = |Pr[\beta' = \beta] - \frac{1}{2}|,$$

where the probability is taken over the choice of tag set $\mathcal{T}$ and the coin tosses of $\mathcal{A}$.

Definition 13. *An adversary $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the unp*-privacy of an RFID system $\mathcal{S}$ if the advantage $Adv_{\mathcal{A}}^{unp^*}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$ of $\mathcal{A}$ in the above game is at least ϵ and the running time of $\mathcal{A}$ is at most t.*

Definition 14. *An RFID system $\mathcal{S}$ is said to be $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-unp*-privacy if there exists no adversary who can $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-break the unp*-privacy of $\mathcal{S}$.*

```
$\mathbf{Game}_{\mathcal{A}}^{unp^*}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$
  $\triangleright$ Setup the reader $\mathcal{R}$ and a set of tags $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$;
  $\triangleright$ $\{\mathcal{T}_c, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3,\mathcal{O}_4}(\mathcal{R}, \mathcal{T})$;
  $\triangleright$ $\beta \in_R \{0, 1\}$;
  $\triangleright$ $\beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathcal{R}, \mathcal{T}_c, st)$.
```

Fig. 6. The *unp**-Privacy Game

4.2 Relationship with *ind*-Privacy Model

In order to clarify the relationship between the *ind*-privacy and *unp**-privacy, we introduce another model, called *ind**-privacy model, as a "bridge" between the two models. We first show that *ind**-privacy is equivalent to *ind*-privacy and then prove that *unp**-privacy implies *ind**-privacy and hence *ind*-privacy.

Figure 7 shows the *ind**-privacy game $\mathbf{Game}_{\mathcal{A}}^{ind^*}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$. The *ind**-privacy game is identical to the *ind*-privacy game given in Figure 2 except that $\mathcal{A}_2$ in the former is only allowed to query $\mathcal{O}_1, \mathcal{O}_2$, and $\mathcal{O}_3$ oracles on $\mathcal{R}$ and $\mathcal{T}_c$.

```
$\mathbf{Game}_{\mathcal{A}}^{ind^*}[\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}]$
  $\triangleright$ Setup reader $\mathcal{R}$ and a set of tags $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$;
  $\triangleright$ $\{\mathcal{T}_i, \mathcal{T}_j, st\} \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3,\mathcal{O}_4}(\mathcal{R}, \mathcal{T})$;
  $\triangleright$ $\beta \in_R \{0, 1\}$;
  $\triangleright$ If $\beta = 0$ then $\mathcal{T}_c = \mathcal{T}_i$, else $\mathcal{T}_c = \mathcal{T}_j$;
  $\triangleright$ $\beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathcal{R}, \mathcal{T}_c, st)$.
```

Fig. 7. The *ind**-Privacy Game

Definition 15. *The advantage of an adversary $\mathcal{A}$ in the above game is defined as*

$$Adv_{\mathcal{A}}^{ind^*}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv}) = |Pr[\beta' = \beta] - \frac{1}{2}|,$$

where the probability is taken over the choice of tag set $\mathcal{T}$ and the coin tosses of the adversary $\mathcal{A}$.

Definition 16. *An adversary $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the ind*-privacy of RFID system $\mathcal{S}$ if the advantage $Adv_{\mathcal{A}}^{ind^*}(\kappa, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$ of $\mathcal{A}$ in the above game is at least ϵ and the running time of $\mathcal{A}$ is at most t.*

Definition 17. *An RFID system $\mathcal{S}$ is said to be $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-ind*-privacy if there exists no adversary who can $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-break the ind*-privacy of $\mathcal{S}$.*

Theorem 2. *For an RFID system $\mathcal{S}$, the ind-privacy and the ind*-privacy are equivalent.*

Proof. It is obvious that *ind*-privacy $\Longrightarrow$ *ind**-privacy holds. Now we prove that *ind*-privacy $\Longleftarrow$ *ind**-privacy also holds.

Suppose there exists an adversary $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the *ind*-privacy of the RFID system $\mathcal{S}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv} + n - 1)$-breaks the *ind**-privacy of the RFID system $\mathcal{S}$. Let $\mathcal{C}$ denote an *ind**-privacy challenger against $\mathcal{B}$. $\mathcal{B}$ runs $\mathcal{A}$ executing the following steps.

Setup $\mathcal{B}$ maintains a list KS-List. Initially the list is empty.

Query phase 1 $\mathcal{A}$ issues **Launch**, **SendTag**, **SendReader** and **Reveal** queries. $\mathcal{B}$ answers them in the following way:
- **Launch** query on $\mathcal{R}$: Issue a **Launch** query on $\mathcal{R}$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **SendTag** query on $(sid, m_1', \mathcal{T}_i \in \mathcal{T})$: Issue a **SendTag** query on $(sid, m_1', \mathcal{T}_i)$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **SendReader** query on (sid, m_2'): Issue a **SendReader** query on (sid, m_2') to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **Reveal** query on $\mathcal{T}_i \in \mathcal{T}$: Issue a **Reveal** query on $\mathcal{T}_i$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.

Challenge Adversary $\mathcal{A}$ submits two uncorrupted tags $\mathcal{T}_{c0}, \mathcal{T}_{c1} \in \mathcal{T}$. $\mathcal{B}$ submits the same two tags $\mathcal{T}_{c0}$ and $\mathcal{T}_{c1}$ to $\mathcal{C}$ which responds with a challenge tag $\mathcal{T}_c \in \{\mathcal{T}_{c0}, \mathcal{T}_{c1}\}$. Then $\mathcal{B}$ *issues Reveal queries on the tag set* $\mathcal{T} - \{\mathcal{T}_{c0}, \mathcal{T}_{c1}\}$ *and stores the results in the list KS-List.* $\mathcal{B}$ forwards $\mathcal{T}_c$ to $\mathcal{A}$. Let $\mathcal{T}'$ denote the tag set $\mathcal{T} - \{\mathcal{T}_{c0}, \mathcal{T}_{c1}\} + \mathcal{T}_c$.

Query phase 2 $\mathcal{A}$ continues to issue **Launch**, **SendTag**, **SendReader** and **Reveal** queries. $\mathcal{B}$ answers them in the following way:
- **Launch** query on $\mathcal{R}$: Issue a **Launch** query on $\mathcal{R}$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **SendTag** query on $(sid, m_1', \mathcal{T}_i \in \mathcal{T}')$: If $\mathcal{T}_i = \mathcal{T}_c$, issue a **SendTag** query on $(sid, m_1', \mathcal{T}_i)$ and forward the result to $\mathcal{A}$; else, use the list KS-List to respond.
- **SendReader** query on (sid, m_2'): Use **SendReader** oracle and the list KS-List to respond.
- **Reveal** query on $\mathcal{T}_i \in \{\mathcal{T} - \{\mathcal{T}_{c0}, \mathcal{T}_{c1}\}\}$: Use the list KS-List to respond.

Guess $\mathcal{A}$ outputs a bit β' which $\mathcal{B}$ also takes as its output.

It is obvious that the simulation is perfect. Thus we have shown an adversary $\mathcal{A}$ against the *ind*-privacy of the RFID system $\mathcal{S}$ with advantage ϵ can be used to construct another adversary $\mathcal{B}$ against the *ind**-privacy of the same RFID system with an identical advantage. Note that, the number of times that $\mathcal{B}$ queries the **Reveal** oracle is $q_{rv} + n - 1$. The running time of $\mathcal{B}$ is approximate to that of $\mathcal{A}$. This completes the proof.

Theorem 3. *Assume that an RFID system $\mathcal{S}$ is of $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-unp*-privacy, then it is also of $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-ind*-privacy.*

Proof. Suppose there exists an adversary $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the *ind**-privacy of the RFID system $\mathcal{S}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ $(\epsilon, t, n, q_{ini}, q_{st}, q_{sr}, q_{rv})$-breaks the *unp**-privacy of the same RFID system $\mathcal{S}$. Let $\mathcal{C}$ denote an *unp**-privacy challenger against $\mathcal{B}$. $\mathcal{B}$ runs $\mathcal{A}$ executing the following steps.

Setup $\mathcal{B}$ does nothing.

Query phase 1 $\mathcal{A}$ issues **Launch**, **SendTag**, **SendReader** and **Reveal** queries. $\mathcal{B}$ answers them in the following way:
- **Launch** query on $\mathcal{R}$: Issue a **Launch** query on $\mathcal{R}$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **SendTag** query on $(sid, m_1', \mathcal{T}_i \in \mathcal{T})$: Issue a **SendTag** query on $(sid, m_1', \mathcal{T}_i)$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **SendReader** query on (sid, m_2'): Issue a **SendReader** query on (sid, m_2') to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **Reveal** query on $\mathcal{T}_i \in \mathcal{T}$: Issue a **Reveal** query on $\mathcal{T}_i$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.

Challenge $\mathcal{A}$ submits two uncorrupted tags $\mathcal{T}_{c0}, \mathcal{T}_{c1} \in \mathcal{T}$. $\mathcal{B}$ selects a random bit $\beta \in \{0,1\}$ and sets the challenge tag $\mathcal{T}_c = \mathcal{T}_{c0}$ if $\beta = 0$ and $\mathcal{T}_c = \mathcal{T}_{c1}$ otherwise. $\mathcal{B}$ submits $\mathcal{T}_c$ to $\mathcal{C}$.

Query phase 2 The adversary continues to issue **Launch**, **SendTag** and **Send Reader** queries. $\mathcal{B}$ answers them as follows:
- **Launch** query on $\mathcal{R}$: Issue a **Launch** query on $\mathcal{R}$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **SendTag** query on $(sid, m_1', \mathcal{T}_c)$: Issue a **SendTag** query on $(sid, m_1', \mathcal{T}_c)$ to $\mathcal{C}$ and forward the result to $\mathcal{A}$.
- **SendReader** query on (sid, m_2'): Issue a **SendReader** query on (sid, m_2') to $\mathcal{C}$ and forward the result to $\mathcal{A}$.

Guess $\mathcal{A}$ outputs a bit β'. If $\beta = \beta'$, $\mathcal{B}$ outputs 1; else, $\mathcal{B}$ outputs 0.

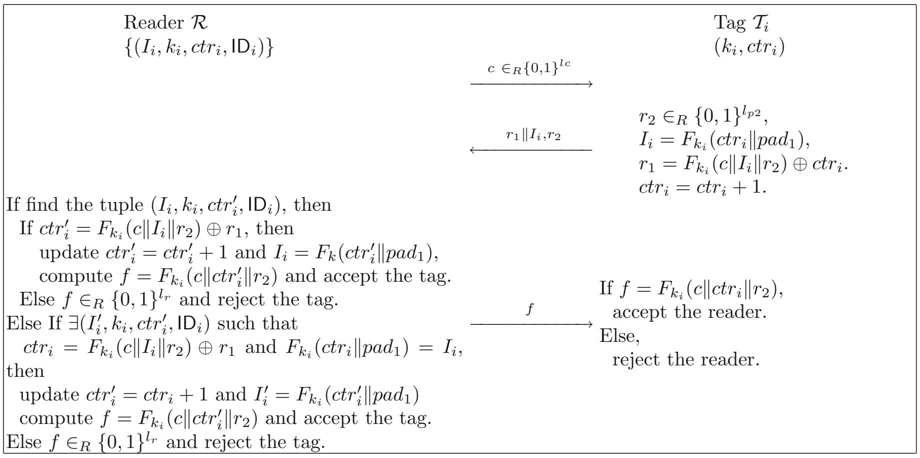
The simulation of $\mathcal{B}$ is perfect. When the binary coin flipped by the *unp**-privacy challenger $\mathcal{C}$ is equal to 1, the probability of $\beta = \beta'$ is equal to $1/2 \pm \epsilon$; otherwise, the probability of $\beta = \beta'$ is equal to $1/2$, because in this case the challenge tag $\mathcal{T}_c$ is in fact a virtual tag in adversary $\mathcal{A}$'s view. Hence, the advantage of $\mathcal{B}$ is equal to that of $\mathcal{A}$ (i. e., ϵ). The running time of $\mathcal{B}$ is exactly the same as that of $\mathcal{A}$. This completes the proof.

Theorem 4. *There exists an RFID system that is of ind*-privacy but is not of unp*-privacy.*

Proof. Suppose an RFID system $\mathcal{S} = (\mathcal{R}, \mathcal{T}, \mathcal{DB}, \mathsf{Initialize}, \mathsf{Execute})$ is of *ind**-privacy, and the output of the algorithm $\mathsf{Execute}$ is (c, r, f). We construct a new RFID system $\mathcal{S}' = (\mathcal{R}, \mathcal{T}, \mathcal{DB}, \mathsf{Initialize}, \mathsf{Execute}')$ such that $(c, r\|r, f) \leftarrow \mathsf{Execute}'$. It is easy to see that $\mathcal{S}'$ is also of *ind**-privacy. Since every protocol transcript of $\mathcal{S}'$ is of the form $(c, r\|r, f)$, an adversary can easily distinguish it from a random tuple $(c', r_1\|r_2, f')$ by checking whether $r_1 = r_2$. Therefore, $\mathcal{S}'$ is not of *unp**-privacy.

4.3 A Protocol with *unp**-Privacy

We now present a 3-round mutual authentication protocol with unp^*-privacy by modifying the 2-round protocol in [13]. This RFID protocol is shown in Figure 8. It is important to note that when the reader $\mathcal{R}$ fails to identify a tag, it does not simply abort, but responds with a random message. A detailed description of the protocol and its proof of unp^*-privacy are given in the full version.



Reader $\mathcal{R}$
$\{(I_i, k_i, ctr_i, \mathsf{ID}_i)\}$

Tag $\mathcal{T}_i$
(k_i, ctr_i)

$\xrightarrow{c \in_R \{0,1\}^{lc}}$

$r_2 \in_R \{0,1\}^{l_{p2}}$,
$I_i = F_{k_i}(ctr_i \| pad_1)$,
$r_1 = F_{k_i}(c \| I_i \| r_2) \oplus ctr_i$.
$ctr_i = ctr_i + 1$.

$\xleftarrow{r_1 \| I_i, r_2}$

If find the tuple $(I_i, k_i, ctr_i', \mathsf{ID}_i)$, then
  If $ctr_i' = F_{k_i}(c \| I_i \| r_2) \oplus r_1$, then
    update $ctr_i' = ctr_i' + 1$ and $I_i = F_k(ctr_i' \| pad_1)$,
    compute $f = F_{k_i}(c \| ctr_i' \| r_2)$ and accept the tag.
  Else $f \in_R \{0,1\}^{l_r}$ and reject the tag.
Else If $\exists (I_i', k_i, ctr_i', \mathsf{ID}_i)$ such that
  $ctr_i = F_{k_i}(c \| I_i \| r_2) \oplus r_1$ and $F_{k_i}(ctr_i \| pad_1) = I_i$, then
  update $ctr_i' = ctr_i + 1$ and $I_i' = F_{k_i}(ctr_i' \| pad_1)$
  compute $f = F_{k_i}(c \| ctr_i' \| r_2)$ and accept the tag.
Else $f \in_R \{0,1\}^{l_r}$ and reject the tag.

$\xrightarrow{f}$

If $f = F_{k_i}(c \| ctr_i \| r_2)$,
  accept the reader.
Else,
  reject the reader.

Fig. 8. The Mutual Authentication Protocol with unp^*-Privacy

5 Conclusions

In this paper we first revisited the formal privacy models for RFID systems existing in the literature, including the *ind*-privacy model [12], the *unp*-privacy model [8] and the newly proposed unp'-privacy model [13]. In doing so, we have highlighted their potential limitations or flaws. In particular, for the first time, we pointed out that though the unp'-privacy model is robust when applied to 2-round RFID protocols but has a deficiency in dealing with 3-round mutual authentication RFID protocols. This deficiency arises from the constraint that the adversary in the guessing stage of the unp'-privacy game is not given any oracle access to the challenge tag. We demonstrated this through a counterexample protocol which has problem with respect to privacy but can be proven to be of unp'-privacy.

We proposed a new privacy model, denoted as unp^*-privacy, based on the indistinguishability of the output of a real tag from that of a virtual tag (e. g., a tag without any secret key). The adversary in the unp^*-privacy game is given multiple oracle accesses to the challenge tag in the guessing stage. The new model does not suffer from the limitations of the *unp*-privacy and the unp'-privacy models. Furthermore, we formally established the relationship between the *ind*-privacy and the unp^*-privacy notions by proving that the former is weaker than the latter. Finally, we extended the 2-round RFID protocol in [13] to a 3-round mutual authentication RFID protocol and showed that it is of *unp**-privacy.

Acknowledgement

We are grateful to the anonymous reviewers for their helpful comments. This work is partly supported by the Office of Research, Singapore Management University, and also supported in part by A*Star SERC Grant No. 082 101 0022 in Singapore.

References

1. Ateniese, G., Camenisch, J., de Medeiros, B.: Untraceable RFID Tags via Insubvertible Encryption. In: Conference on Computer and Communications Security, CCS '05, pp. 92–101 (2005)
2. Avoine, G.: Security and privacy in RFID systems (2006), `http://lasecwww.epfl.ch/~gavoine/rfid/`
3. Avoine, G.: Adversary Model for Radio Frequency Identification. Technical Report LASEC-REPORT-2005-001, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC) (2005)
4. van Deursen, T., Radomirović, S.: On a New Formal Proof Model for RFID Location Privacy. Cryptology ePrint Archive, Report 2008/477.
5. Fishkin, K., Roy, S., Jiang, B.: Some methods for privacy in RFID communication. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 42–53. Springer, Heidelberg (2005)
6. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal re-encryption for mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004)
7. Henrici, D., Müller, P.: Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In: Workshop on Pervasive Computing and Communications Security-PerSec 2004, pp. 149–153. IEEE Computer Society, Los Alamitos (2004)
8. Ha, J., Moon, S., Zhou, J., Ha, J.: A new formal proof model for RFID location privacy. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 267–281. Springer, Heidelberg (2008)
9. Juels, A.: RFID Security and Privacy: A Research Survey. IEEE Journal on Selected Areas in Communications 24(2), 381–394 (2006)
10. Juels, A., Pappu, R.: Squealing euros: Privacy protection in RFID-enabled banknotes. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 103–121. Springer, Heidelberg (2003)
11. Juels, A., Rivest, R., Szydlo, M.: The blocker tag: Selective blocking of RFID tags for consumer privacy. In: CCS '03, pp. 103–111. ACM Press, New York (2003)
12. Juels, A., Weis, S.A.: Defining strong privacy for RFID. ePrint, Report 2006/137, 2006, PerCom 2007
13. Ma, C., Li, Y., Deng, R.H., Li, T.: RFID Privacy: Relation Between Two Notions, Minimal Condition, and Efficient Construction. In: ACM CCS 2009 (2009)

14. Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient Hash-Chain Based RFID Privacy Protection Scheme. In: International Conference on Ubiquitous Computing-Ubicomp, Workshop Privacy: Current Status and Future Directions (2004)
15. Pedro, P.L., Cesar, H.C.J., Juan, M.E.T., Arturo, R.: RFID Systems: A Survey on Security Threats and Proposed Solutions. In: Cuenca, P., Orozco-Barbosa, L. (eds.) PWC 2006. LNCS, vol. 4217, pp. 159–170. Springer, Heidelberg (2006)
16. Paise, R.-I., Vaudenay, S.: Mutual authentication in RFID: Security and privacy. In: Proc. of ASIACCS, pp. 292–299. ACM Press, New York (2008)
17. Sadeghi, A.-R., Visconti, I., Wachsmann, C.: Anonymizer-enabled security and privacy for RFID. In: Miyaji, A., Echizen, I., Okamoto, T. (eds.) CANS 2009. LNCS, vol. 5888, pp. 134–153. Springer, Heidelberg (2009)
18. Saito, J., Ryou, J.-C., Sakurai, K.: Enhancing privacy of universal re-encryption scheme for RFID tags. In: Yang, L.T., Guo, M., Gao, G.R., Jha, N.K. (eds.) EUC 2004. LNCS, vol. 3207, pp. 879–890. Springer, Heidelberg (2004)
19. Tsudik, G.: YA-TRAP: Yet Another Trivial RFID Authentication Protocol. In: International Conference on Pervasive Computing and Communications, PerCom 2006, pp. 640–643 (2006)
20. Vaudenay, S.: On privacy models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)
21. Weis, S., Sarma, S., Rivest, R., Engels, D.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In: International Conference on Security in Pervasive Computing-SPC 2003 (2003)
22. Yu Ng, C., Susilo, W., Mu, Y., Safavi-Naini, R.: RFID privacy models revisited. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 251–266. Springer, Heidelberg (2008)

On RFID Privacy with Mutual Authentication and Tag Corruption

Frederik Armknecht[1], Ahmad-Reza Sadeghi[2], Ivan Visconti[3], and Christian Wachsmann[2]

[1] University of Mannheim, Germany
armknecht@math.uni-mannheim.de
[2] Horst Görtz Institute for IT-Security (HGI), Ruhr-University Bochum, Germany
{ahmad.sadeghi,christian.wachsmann}@trust.rub.de
[3] Dipartimento di Informatica ed Applicazioni, University of Salerno, Italy
visconti@dia.unisa.it

Abstract. RFID systems have become increasingly popular and are already used in many real-life applications. Although very useful, RFIDs also introduce privacy risks since they carry identifying information that can be traced. Hence, several RFID privacy models have been proposed. However, they are often incomparable and in part do not reflect the capabilities of real-world adversaries. Recently, Paise and Vaudenay presented a general RFID security and privacy model that abstracts and unifies most previous approaches. This model defines mutual authentication (between the RFID tag and reader) and several privacy notions that capture adversaries with different tag corruption behavior and capabilities.

In this paper, we revisit the model proposed by Paise and Vaudenay and investigate some subtle issues such as tag corruption aspects. We show that in their formal definitions tag corruption discloses the temporary memory of tags and leads to the impossibility of achieving both mutual authentication and any reasonable notion of RFID privacy in their model. Moreover, we show that the strongest privacy notion (narrow-strong privacy) cannot be achieved simultaneously with reader authentication even if the adversary is not capable of corrupting a tag during the protocol execution.

Although our results are shown on the privacy definition by Paise and Vaudenay, they give insight to the difficulties of setting up a mature security and privacy model for RFID systems that aims at fulfilling the sophisticated requirements of real-life applications.

Keywords: RFID, Security Model, Privacy, Mutual Authentication.

1 Introduction

Radio Frequency Identification (RFID) enables RFID *readers* to perform fully automatic wireless identification of objects that are labeled with RFID *tags*, and is widely deployed to many applications (e.g., access control [1,2], electronic tickets [2,3], e-passports [4]). As pointed out in previous publications (see, e.g., [5,6,7]), this prevalence of RFID technology introduces various risks, in particular concerning the privacy of its users and holders. The most deterrent privacy risk concerns the tracking of users, which allows the creation and misuse of detailed user profiles. Thus, it is desired that an RFID system provides *anonymity* (confidentiality of the tag identity) as well as *untraceability* (unlinkability of the communication of a tag), even in case the state (e.g., the secret) of a tag has been disclosed.

The design of a secure privacy-preserving RFID scheme requires a careful analysis in an appropriate formal model. There is a large body of literature on security and privacy models for RFID (see, e.g., [8,9,10,11,12,13]). Existing solutions often do not consider important aspects like adversaries with access to auxiliary information (e.g., on whether the identification of a tag was successful), or the privacy of corrupted tags whose state has been disclosed. In particular, tag corruption is usually considered to happen only *before* and *after*, but *not during* a protocol execution. However, in practice there are a variety of side-channel attacks (see., e.g., [14,15,16]) that extract the state of a powered tag based on the observation of (e.g., the power consumption of) the tag *while* it is executing a protocol with the reader. Since RFID tags are usually cost-effective devices without expensive tamper-proof mechanisms [1,2], tag corruption is an important aspect to be covered by the underlying (formal) model. Though in literature, tag corruption during protocol execution is rarely considered. To the best of our knowledge, the security and privacy model in [10] is the only one that considers corruption of tags during protocol executions and proposes a protocol in this model. However, this model does not consider issues like the privacy of tags *after* they have been corrupted and privacy against adversaries with access to auxiliary information. Moreover, [10] only provides an informal security analysis of the proposed protocol. Recently, tag corruption during protocol executions has been informally discussed in [13]. However, the formal RFID security and privacy model proposed in [13] assumes that such attacks cannot occur. Moreover, [13] indicates informally (without giving formal arguments and proofs) that tag corruption during protocol execution may have an impact on the formal definitions of [9] and [11,12], which is basis for many subsequent works (see, e.g., [19,20,21,22,23,24,25,26]). The first papers addressing tag corruption during protocol execution in the model of [11] are [24,25], where it is shown that privacy can be achieved under the assumption that tag corruption during protocol execution can be detected by the tag.

In this paper, we focus on the security and privacy model by Paise and Vaudenay [12] (that is based on [11]), which we call the *PV-Model* (Paise-Vaudenay Model) in the following. The PV-Model is one of the most comprehensive RFID security and privacy models up to date since it captures many aspects of real world RFID systems and aims at abstracting most previous works in a single concise framework. It defines mutual authentication between RFID tags and readers and several privacy notions that correspond to adversaries with different tag corruption abilities. However, as we show in this paper, the PV-Model suffers from subtle deficiencies and weaknesses that are mainly caused by tag corruption aspects: in the PV-Model, each tag maintains a state that can be divided into a persistent and a temporary part.[1] The *persistent state* subsumes all information that must be available to the tag in more than one interaction with the reader (e.g., the authentication secret of the tag) and can be updated during the interaction with the reader. The *temporary state* consists of all ephemeral information that is discarded by the tag after each interaction with the reader (e.g., the randomness used by the tag). The PV-Model shows that if the adversary can obtain *both* the persistent *and* the temporary tag state by tag corruption, then it is impossible to achieve any notion of privacy that allows tag corruption in their model. They address this issue by assuming that each tag erases its temporary state each time it gets out of the reading range of the adversary. However, this assumption leaves open the possibility to corrupt a tag *while* it is in the reading range of the adversary, i.e., *before* its temporary state is erased. In particular, the PV-Model allows the adversary to corrupt a tag *while* it is executing the authentication protocol with the reader.

Contribution. In this paper, we point out subtle weaknesses and deficiencies in the PV-Model. First, we show that the assumption of erasing temporary tag states whenever a tag gets out of the reading range of the adversary made by the PV-Model is not strong enough. We prove that, even under this assumption, it is *impossible* to achieve reader authentication and simultaneously *any* notion of privacy that allows tag corruption. This implies that the PV-Model cannot provide privacy along with mutual authentication without relying on tamper-proof hardware, which is unrealistic for low-cost RFID tags. Consequently, we show attacks on the privacy goals of two of the three schemes presented in [12].

Our second contribution is to show that even under the strong assumption that the temporary tag state is not subject to corruption attacks, some privacy notions still remain impossible in the PV-Model. This implies that the third protocol of [12] does not achieve its claimed privacy goal. On the good side, we prove that some privacy notions can be met under the assumption that temporary tag states are not disclosed by tag corruption.

Although our results are shown on the privacy definition by Paise and Vaudenay, we believe that our work is helpful for developing a mature security and privacy model for RFID systems that fulfills the sophisticated requirements of real-life applications.

2 RFID System and Requirement Analysis

System model. An RFID system consists of at least an operator $\mathcal{I}$, a reader $\mathcal{R}$ and a tag $\mathcal{T}$. $\mathcal{I}$ is the entity that enrolls and maintains the RFID system. Hence, $\mathcal{I}$ initializes $\mathcal{T}$ and $\mathcal{R}$ before they are deployed in the system. $\mathcal{T}$ and $\mathcal{R}$ are called *legitimate* if they have been initialized by $\mathcal{I}$. In many applications $\mathcal{T}$

[1] During a protocol execution, tags could store some temporary information that allows them to verify the response of the reader.

is a hardware token with constrained computing and memory capabilities that is equipped with a radio interface [1,2]. All information (e.g., secrets and data) stored on $\mathcal{T}$ is denoted as the *state* of $\mathcal{T}$. Usually $\mathcal{T}$ is attached to some object or carried by a user of the RFID system. $\mathcal{R}$ is a stationary or mobile computing device that interacts with $\mathcal{T}$ when $\mathcal{T}$ gets into the reading range of $\mathcal{R}$. The main purpose of this interaction usually is the authentication of $\mathcal{T}$ to $\mathcal{R}$. Depending on the use case, $\mathcal{R}$ may also authenticate to $\mathcal{T}$ and/or obtain additional information like the identity of $\mathcal{T}$. $\mathcal{R}$ can have a sporadic or permanent online connection to some backend system $\mathcal{D}$, which typically is a database maintaining detailed information on all tags in the system. $\mathcal{D}$ is initialized and maintained by $\mathcal{I}$ and can be read and updated by $\mathcal{R}$.

Trust and adversary model. The operator $\mathcal{I}$ maintains the RFID system, and thus is considered to behave correctly. However, $\mathcal{I}$ may be curious since he may collect user information (see, e.g., [5,6]). Since $\mathcal{T}$ and $\mathcal{R}$ communicate over a radio link, any entity can eavesdrop and manipulate this communication, even from outside the nominal reading range of $\mathcal{R}$ and $\mathcal{T}$ [27]. Thus, the adversary $\mathcal{A}$ can be every (potentially unknown) entity. Besides the communication between $\mathcal{T}$ and $\mathcal{R}$, $\mathcal{A}$ can also obtain useful auxiliary information (e.g., by visual observation) on whether $\mathcal{R}$ accepted $\mathcal{T}$ as a legitimate tag [9,12]. Most commercial RFID tags are cost-efficient devices without expensive protection mechanisms against physical tampering [1,2]. Hence, $\mathcal{A}$ can physically attack (*corrupt*) $\mathcal{T}$ and obtain its state. In practice, RFID readers are embedded devices that can be integrated into mobile devices (e.g., mobile phones or PDAs) or computers. The resulting complexity exposes them to sophisticated hard- and software attacks (e.g., viruses and Trojans). This problem aggravates for mobile readers that can easily be lost or stolen. Hence, $\mathcal{A}$ can get full control over $\mathcal{R}$ [28,29,30].

Security and privacy objectives. The most deterrent privacy risk concerns the *tracking* of tag users, which allows the creation and misuse of detailed user profiles in an RFID system [6]. For instance, detailed movement profiles can leak sensitive information on the personal habits and interests of the tag user. The major security threats are to create illegitimate (*forge*) tags that are accepted by honest readers, to simulate (*impersonate*) or to copy (*clone*) legitimate tags, and to permanently prevent users from using the RFID system (*denial-of-service*) [10]. Thus, an RFID system should provide *anonymity* (confidentiality of the tag identity) as well as *untraceability* (unlinkability of the communication of a tag) even if the state of a tag has been disclosed. The main security objective is to ensure that only legitimate tags are accepted by honest readers (*tag authentication*). Most use cases additionally require $\mathcal{R}$ to determine the authentic tag identity (*tag identification*). Moreover, there are several applications (e.g., electronic tickets) where reader authentication is a fundamental security property. However, there are also use cases (e.g., electronic product labels) that do not require reader authentication.

3 Notation

For a finite set S, $|S|$ denotes the size of S whereas for an integer n the term $|n|$ means the bit-length of n. The term $s \in_R S$ means the assignment of a uniformly chosen element of S to s. Let A be a probabilistic algorithm. Then $y \leftarrow \mathsf{A}(x)$ means that on input x, algorithm A assigns its output to variable y. The term $[\mathsf{A}(x)]$ denotes the set of all possible outputs of A on input x. $\mathsf{A}_K(x)$ means that the output of A depends on x and some additional parameter K (e.g., a secret key). The term $\mathsf{Prot}[\mathsf{A}:x_\mathsf{A};\ \mathsf{B}:x_\mathsf{B};\ *:x_{pub}] \rightarrow [\mathsf{A}:y_\mathsf{A};\ \mathsf{B}:y_\mathsf{B}]$ denotes an interactive protocol Prot between two algorithms A and B. Hereby, A (resp. B) gets a private input x_A (resp. x_B) and a public input x_{pub}. While A (resp. B) is operating, it can interact with B (resp. A). After the protocol terminates, A (resp. B) returns y_A (resp. y_B). Let E be some event (e.g., the result of a security experiment), then $\Pr[E]$ denotes the probability that E occurs. Probability $\epsilon(l)$ is called *negligible* if for all polynomials f it holds that $\epsilon(l) \leq 1/f(l)$ for all sufficiently large l. Probability $1-\epsilon(l)$ is called *overwhelming* if $\epsilon(l)$ is negligible.

4 The PV-Model

In this section, we recall Paise and Vaudenay's model (PV-Model) [12], which is one of the most comprehensive RFID security and privacy models up to date.

4.1 System Model

The PV-Model considers RFID systems that consist of a single operator $\mathcal{I}$, a single reader $\mathcal{R}$ and a polynomial number of tags $\mathcal{T}$. $\mathcal{R}$ is assumed to be capable of performing public-key cryptography and of handling multiple instances of the mutual authentication protocol with different tags in parallel. Each tag $\mathcal{T}$ is a passive device, i.e., it does not have its own power supply but is powered by the electromagnetic field of $\mathcal{R}$. Hence, $\mathcal{T}$ cannot initiate communication, has a narrow communication range (i.e., a few centimeters to meters) and erases its temporary state (i.e., all session-specific information and randomness) after it gets out of the reading range of $\mathcal{R}$. Each $\mathcal{T}$ is assumed to be capable of performing basic cryptographic functions like random number generation, hashing and symmetric-key encryption. The authors of [12] also use public-key encryption, although it exceeds the capabilities of most currently available RFID tags.

The operator $\mathcal{I}$ sets up the reader $\mathcal{R}$ and all tags $\mathcal{T}$. Hence, there are two setup algorithms to generate the system parameters (e.g., keys) of $\mathcal{R}$ and $\mathcal{T}$. Mutual authentication is covered by a third protocol between $\mathcal{T}$ and $\mathcal{R}$.

Definition 1 (RFID System [12]). *An RFID system is a tuple of probabilistic polynomial time (p.p.t.) algorithms* $(\mathcal{R}, \mathcal{T}, \mathsf{SetupReader}, \mathsf{SetupTag}, \mathsf{Ident})$ *that are defined as follows:*

$\mathsf{SetupReader}(1^l) \rightarrow (sk_\mathcal{R}, pk_\mathcal{R}, \mathtt{DB})$ *On input of a security parameter l, this algorithm creates the public system parameters $pk_\mathcal{R}$ that are known to all entities.*

Moreover, it creates the secret system parameters $sk_{\mathcal{R}}$ *and a database* $\mathtt{DB}$ *that can only be accessed by* $\mathcal{R}$*.*

$\mathsf{SetupTag}_{pk_{\mathcal{R}}}(\mathtt{ID}) \to (K, S)$ *uses* $pk_{\mathcal{R}}$ *to generate a tag secret* K *and tag state* S*, initializes* $\mathcal{T}_{\mathtt{ID}}$ *with* S*, and stores* $(\mathtt{ID}, K)$ *in* $\mathtt{DB}$*.*

$\mathsf{Ident}[\mathcal{T}_{\mathtt{ID}}\!:\!S;\ \mathcal{R}\!:\!sk_{\mathcal{R}}, \mathtt{DB};\ *\!:\!pk_{\mathcal{R}}] \to [\mathcal{T}_{\mathtt{ID}}\!:\!out_{\mathcal{T}_{\mathtt{ID}}};\ \mathcal{R}\!:\!out_{\mathcal{R}}]$ *is an interactive protocol between* $\mathcal{T}_{\mathtt{ID}}$ *and* $\mathcal{R}$*.* $\mathcal{T}_{\mathtt{ID}}$ *takes as input its current state* S *while* $\mathcal{R}$ *has input* $sk_{\mathcal{R}}$ *and* $\mathtt{DB}$*. The common input to all parties is* $pk_{\mathcal{R}}$*. After the protocol terminates,* $\mathcal{R}$ *returns either the identity* $\mathtt{ID}$ *of* $\mathcal{T}_{\mathtt{ID}}$ *or* $\perp$ *to indicate that* $\mathcal{T}_{\mathtt{ID}}$ *is not a legitimate tag.* $\mathcal{T}_{\mathtt{ID}}$ *returns either* $\mathtt{ok}$ *to indicate that* $\mathcal{R}$ *is legitimate or* $\perp$ *otherwise.*

Correctness describes the honest behavior of legitimate tags and the reader $\mathcal{R}$.

Definition 2 (Correctness [12]). *An RFID system (Definition 1) is correct if* $\forall\ l$*,* $\forall\ (sk_{\mathcal{R}}, pk_{\mathcal{R}}, \mathtt{DB}) \in [\mathsf{SetupReader}(1^l)]$*, and* $\forall\ (K, S) \in [\mathsf{SetupTag}_{pk_{\mathcal{R}}}(\mathtt{ID})]$ $\Pr\left[\mathsf{Ident}[\mathcal{T}_{\mathtt{ID}}\!:\!S;\ \mathcal{R}\!:\!sk_{\mathcal{R}}, \mathtt{DB};\ *\!:\!pk_{\mathcal{R}}] \to [\mathcal{T}_{\mathtt{ID}}\!:\!\mathtt{ok};\ \mathcal{R}\!:\!\mathtt{ID}]\right]$ *is overwhelming.*

4.2 Trust and Adversary Model

The PV-Model assumes the issuer $\mathcal{I}$, the backend database $\mathcal{D}$ and the readers to be trusted, whereas a tag $\mathcal{T}$ can be compromised. All readers and $\mathcal{D}$ are subsumed to *one single* reader entity $\mathcal{R}$ that cannot be corrupted. The PV-Model defines privacy and security as experiments, where an adversary $\mathcal{A}$ interacts with a set of oracles that model the capabilities of $\mathcal{A}$. These oracles are:

$\mathsf{CreateTag}^b(\mathtt{ID})$ Allows $\mathcal{A}$ to set up a tag $\mathcal{T}_{\mathtt{ID}}$ with identifier $\mathtt{ID}$ by internally calling $\mathsf{SetupTag}_{pk_{\mathcal{R}}}(\mathtt{ID})$ to create (K, S) for $\mathcal{T}_{\mathtt{ID}}$. If input $b = 1$, then $(\mathtt{ID}, K)$ is added to $\mathtt{DB}$. If $b = 0$, then $(\mathtt{ID}, K)$ is *not* added to $\mathtt{DB}$.

$\mathsf{Draw}(\delta) \to (vtag_1, b_1, \ldots, vtag_n, b_n)$ Initially, $\mathcal{A}$ cannot interact with any tag but must query Draw to get access to a set of tags that has been chosen according to a given probability distribution δ. $\mathcal{A}$ knows the tags he can interact with by temporary tag identifiers $vtag_1, \ldots, vtag_n$. Draw manages a secret table Γ that links each temporary tag identifier $vtag_i$ to the corresponding real tag identifier $\mathtt{ID}_i$ (i.e., $\Gamma[vtag_i] = \mathtt{ID}_i$). Moreover, Draw provides $\mathcal{A}$ with information on whether the tags are legitimate ($b_i = 1$) or not ($b_i = 0$).

$\mathsf{Free}(vtag)$ Makes $vtag$ inaccessible to $\mathcal{A}$ such that $\mathcal{A}$ can no longer interact with $vtag$ until it is made accessible again (under a new temporary identifier $vtag'$) by another Draw query.

$\mathsf{Launch}() \to \pi$ Makes $\mathcal{R}$ to start a new instance π of the Ident protocol.

$\mathsf{SendReader}(m, \pi) \to m'$ Sends a message m to instance π of the Ident protocol that is running on $\mathcal{R}$. $\mathcal{R}$ interprets m as a protocol message of instance π of the Ident protocol and responds with a message m'.

$\mathsf{SendTag}(m, vtag) \to m'$ Sends a message m to the tag $\mathcal{T}_{\mathtt{ID}}$ that is known as $vtag$ to $\mathcal{A}$. $\mathcal{T}_{\mathtt{ID}}$ interprets m as a protocol message of the Ident protocol and responds with a message m'.

$\mathsf{Result}(\pi)$ Returns 1 if instance π of the Ident protocol has been completed and the tag that participated in instance π has been accepted by $\mathcal{R}$. Otherwise Result returns 0.

Corrupt($vtag$) $\rightarrow S$ Returns the current state S (i.e., all information stored in the memory) of the tag $\mathcal{T}_{\mathtt{ID}}$ that is known as $vtag$ to $\mathcal{A}$.

The PV-Model distinguishes eight adversary classes, which differ in (i) their ability to corrupt tags and (ii) the availability of auxiliary information (i.e., the ability to access the Corrupt and Result oracle, respectively).

Definition 3 (Adversary Classes [12]). *An adversary is a p.p.t. algorithm that has arbitrary access to all oracles described in Section 4.2. Weak adversaries cannot access the* Corrupt *oracle. Forward adversaries can no longer query any other oracle than* Corrupt *after they made the first* Corrupt *query. Destructive adversaries cannot query any oracle for vtag again after they made a* Corrupt($vtag$) *query. Strong adversaries have no restrictions on the use of the* Corrupt *oracle. Narrow adversaries cannot access the* Result *oracle.*

Tag corruption aspects. Depending on the concrete scenario one could have that the temporary tag state is disclosed under tag corruption. In general, any concrete scenario will range between the following two extremes: (i) corruption discloses the full temporary tag state or (ii) corruption does not disclose any information on the temporary tag state. In Sections 5 and 6, we will prove that in both cases some privacy notions are impossible to achieve in the PV-Model. Thus, *independently* of any possible interpretation of tag corruption, impossibility results exist that revisit the claims of [12].

4.3 Security Definition

The security definition of the PV-Model focuses on attacks where the adversary aims to impersonate or forge a legitimate tag $\mathcal{T}$ or the reader $\mathcal{R}$. It does *not* capture availability and security against cloning.

Tag authentication. The definition of tag authentication is based on a security experiment $\mathbf{Exp}_{\mathcal{A}_{\text{sec}}}^{\mathcal{T}\text{-aut}}$, where a strong adversary $\mathcal{A}_{\text{sec}}$ (Definition 3) must make the reader $\mathcal{R}$ to identify some tag $\mathcal{T}_{\mathtt{ID}}$ in some instance π of the Ident protocol. To exclude trivial attacks (e.g., relay attacks), $\mathcal{A}_{\text{sec}}$ is not allowed to simply forward all the messages from $\mathcal{T}_{\mathtt{ID}}$ to $\mathcal{R}$ in instance π nor to corrupt $\mathcal{T}_{\mathtt{ID}}$. This means that at least some of the protocol messages that made $\mathcal{R}$ to return ID must have been computed by $\mathcal{A}_{\text{sec}}$ without knowing the secrets of $\mathcal{T}_{\mathtt{ID}}$. With $\mathbf{Exp}_{\mathcal{A}_{\text{sec}}}^{\mathcal{T}\text{-aut}} = 1$ we denote the case where $\mathcal{A}_{\text{sec}}$ wins the security experiment.

Definition 4 (Tag Authentication [12]). *An RFID system (Definition 1) achieves tag authentication if for every strong adversary $\mathcal{A}_{\text{sec}}$ (Definition 3)* $\Pr[\mathbf{Exp}_{\mathcal{A}_{\text{sec}}}^{\mathcal{T}\text{-aut}} = 1]$ *is negligible.*

Note that tag authentication is a critical property and hence must be preserved even against strong adversaries.

Reader authentication. The definition of reader authentication is based on a security experiment $\mathbf{Exp}_{\mathcal{A}_{\text{sec}}}^{\mathcal{R}\text{-aut}}$, where a strong adversary $\mathcal{A}_{\text{sec}}$ (Definition 3) must successfully impersonate the reader $\mathcal{R}$ to a legitimate tag $\mathcal{T}_{\mathtt{ID}}$. Also here, to exclude trivial attacks, $\mathcal{A}_{\text{sec}}$ must achieve this without simply forwarding the protocol messages from $\mathcal{R}$ to $\mathcal{T}_{\mathtt{ID}}$. With $\mathbf{Exp}_{\mathcal{A}_{\text{sec}}}^{\mathcal{R}\text{-aut}} = 1$ we denote the case where $\mathcal{A}_{\text{sec}}$ wins the security experiment.

Definition 5 (Reader Authentication [12]). *An RFID system (Definition 1) achieves reader authentication if for every strong adversary $\mathcal{A}_{\text{sec}}$ (Definition 3) $\Pr[\mathbf{Exp}_{\mathcal{A}_{\text{sec}}}^{\mathcal{R}\text{-aut}} = 1]$ is negligible.*

4.4 Privacy Definition

The privacy definition of the PV-Model is very flexible and, dependent on the adversary class (see Definition 3), it covers different notions of privacy. It captures anonymity and unlinkability and focuses on the privacy leakage of the communication of tags with the reader. It is based on the existence of a simulator $\mathcal{B}$, called *blinder*, that can simulate every tag $\mathcal{T}$ and the reader $\mathcal{R}$ without knowing their secrets such that an adversary $\mathcal{A}_{\text{prv}}$ cannot distinguish whether it is interacting with the real or the simulated RFID system.

The privacy definition can be formalized by the following experiment $\mathbf{Exp}_{\mathcal{A}_{\text{prv}}}^{\text{prv-}b}$: Let $\mathcal{A}_{\text{prv}}$ be an adversary according to Definition 3, l be a given security parameter and $b \in_R \{0,1\}$. In the first phase of the experiment, $\mathcal{R}$ is initialized with $(sk_{\mathcal{R}}, pk_{\mathcal{R}}, \mathtt{DB}) \leftarrow \mathsf{SetupReader}(1^l)$. The public key $pk_{\mathcal{R}}$ is given to $\mathcal{A}_{\text{prv}}$ and $\mathcal{B}$. Now, $\mathcal{A}_{\text{prv}}$ is allowed to arbitrarily interact with all oracles defined in Section 4.2. Hereby, $\mathcal{A}_{\text{prv}}$ is subject to the restrictions of its corresponding adversary class (see Definition 3). If $b = 1$, all queries to the Launch, SendReader, SendTag and Result oracles are redirected to and answered by $\mathcal{B}$. Hereby, $\mathcal{B}$ can observe all queries $\mathcal{A}_{\text{prv}}$ makes to all other oracles that are not simulated by $\mathcal{B}$ and the corresponding responses ("$\mathcal{B}$ sees what $\mathcal{A}_{\text{prv}}$ sees"). After a polynomial number of oracle queries, the second phase of the experiment starts, where $\mathcal{A}_{\text{prv}}$ can no longer interact with the oracles but is given the secret table Γ of the Draw oracle. Finally, $\mathcal{A}_{\text{prv}}$ returns a bit b', which we denote with $\mathbf{Exp}_{\mathcal{A}_{\text{prv}}}^{\text{prv-}b} = b'$.

Definition 6 (Privacy [11]). *Let C be an adversary class according to Definition 3. An RFID system (Definition 1) is C-private if for every adversary $\mathcal{A}_{\text{prv}}$ of C there exists a p.p.t. algorithm $\mathcal{B}$ (blinder) such that the advantage*

$$\mathbf{Adv}_{\mathcal{A}_{\text{prv}}}^{\text{prv}} = \left| \Pr\left[\mathbf{Exp}_{\mathcal{A}_{\text{prv}}}^{\text{prv-0}} = 1\right] - \Pr\left[\mathbf{Exp}_{\mathcal{A}_{\text{prv}}}^{\text{prv-1}} = 1\right] \right|$$

of $\mathcal{A}_{\text{prv}}$ is negligible. $\mathcal{B}$ simulates the Launch, SendReader, SendTag *and* Result *oracles to $\mathcal{A}_{\text{prv}}$ without having access to $sk_{\mathcal{R}}$ and* DB. *Hereby, all oracle queries $\mathcal{A}_{\text{prv}}$ makes and their corresponding responses are also sent to $\mathcal{B}$.*

Figure 1 summarizes all privacy notions defined in the PV-Model and shows the relations among them. It has been shown that strong privacy is impossible while the technical feasibility of destructive privacy currently is an open problem [11].

$$\begin{tikzcd}
\text{Strong} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{Destructive} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{Forward} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{Weak} \arrow[d, Rightarrow] \\
\text{Narrow-Strong} \arrow[r, Rightarrow] & \text{Narrow-Destructive} \arrow[r, Rightarrow] & \text{Narrow-Forward} \arrow[r, Rightarrow] & \text{Narrow-Weak}
\end{tikzcd}$$

Fig. 1. Privacy notions defined in the PV-Model and their relations

5 Corruption with Temporary State Disclosure

We now point out a subtle conceptual weakness of the PV-Model and revisit two of the claims given in [12] about their protocols, that do not achieve the claimed privacy properties. We first illustrate our adversarial strategy by showing how tag corruption (as defined in the PV-Model) can be exploited to attack one of the protocols proposed in [12]. Then we generalize our attack to the class of narrow-forward private protocols, which finally leads to our first impossibility result: in the PV-Model it is *impossible* to achieve any notion of privacy simultaneously with reader authentication (under temporary state disclosure) except for the weak and narrow-weak privacy notions.

We stress that this impossibility result is due to the fact that, according to the formal definitions of the PV-Model, the adversary can obtain the full state (including the temporary memory) of a tag by corrupting the tag *while* it is executing a protocol with the reader. In face of side-channel attacks (see, e.g., [15,16]), such attacks can be feasible in practice (in particular against low-cost RFID tags) and hence, must be formally considered. Although [12] informally discusses an issue related to tag corruption during protocol execution, we show that such attacks are *not* adequately captured by the formal definitions of the PV-Model. Hence, the only achievable privacy notions are those where the adversary is not allowed to corrupt tags at all. Since in practice tag corruption is realistic, this implies that using the PV-Model is not helpful when reader authentication and a reasonable notion of privacy are needed.

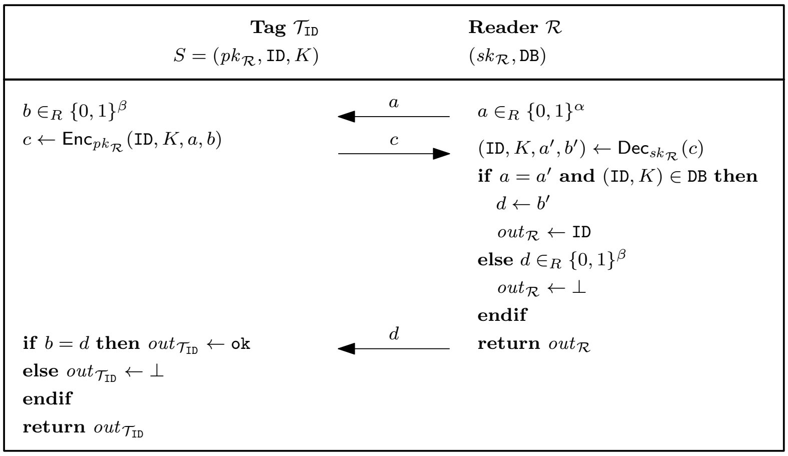
5.1 Illustrative Example

We first show our attack on protocol 3 of [12], which is claimed to provide reader authentication while being narrow-strong private.[2]

Protocol description. Let $l, k, \alpha, \beta \in \mathbb{N}$ be given security parameters. $\mathcal{R}$ is initialized with the credentials database $\mathtt{DB}$ and the secret key $sk_{\mathcal{R}}$ of a CCA-secure public-key encryption scheme, while $\mathcal{T}_{\mathtt{ID}}$ is initialized with the corresponding public key $pk_{\mathcal{R}}$, a given tag identifier $\mathtt{ID}$ and $K \in_R \{0,1\}^k$. The Ident protocol is illustrated in Figure 2 and works as follows: Upon receipt of a, $\mathcal{T}_{\mathtt{ID}}$ stores b in its temporary memory and sends c to $\mathcal{R}$.[3] If $\mathcal{T}_{\mathtt{ID}}$ is legitimate, $\mathcal{R}$ can identify $\mathcal{T}_{\mathtt{ID}}$

[2] In the full version of this paper [31], we show that the same attack can be launched on the second protocol of [12], which is claimed to be narrow-destructive private.

[3] Note that $\mathcal{T}_{\mathtt{ID}}$ interprets the protocol messages sent by $\mathcal{R}$ based on the value of b. If b is empty (i.e., has been erased), then $\mathcal{T}_{\mathtt{ID}}$ considers the message sent by $\mathcal{R}$ as the first, and as the third protocol message otherwise.



```
Tag $\mathcal{T}_{\texttt{ID}}$                                  Reader $\mathcal{R}$
$S = (pk_{\mathcal{R}}, \texttt{ID}, K)$                          $(sk_{\mathcal{R}}, \texttt{DB})$

$b \in_R \{0,1\}^\beta$                   <--- a ---    $a \in_R \{0,1\}^\alpha$
$c \leftarrow \mathsf{Enc}_{pk_{\mathcal{R}}}(\texttt{ID}, K, a, b)$   --- c --->    $(\texttt{ID}, K, a', b') \leftarrow \mathsf{Dec}_{sk_{\mathcal{R}}}(c)$
                                                        if $a = a'$ and $(\texttt{ID}, K) \in \texttt{DB}$ then
                                                          $d \leftarrow b'$
                                                          $out_{\mathcal{R}} \leftarrow \texttt{ID}$
                                                        else $d \in_R \{0,1\}^\beta$
                                                          $out_{\mathcal{R}} \leftarrow \perp$
                                                        endif
if $b = d$ then $out_{\mathcal{T}_{\texttt{ID}}} \leftarrow \texttt{ok}$   <--- d ---    return $out_{\mathcal{R}}$
else $out_{\mathcal{T}_{\texttt{ID}}} \leftarrow \perp$
endif
return $out_{\mathcal{T}_{\texttt{ID}}}$
```

Fig. 2. Protocol 3 of [12]

by verifying that $a = a'$ and checking if $\texttt{DB}$ contains$(\texttt{ID}, K)$. If this is the case, $\mathcal{R}$ sends $d \leftarrow b'$ to $\mathcal{T}_{\texttt{ID}}$ and outputs $\texttt{ID}$. Otherwise, $\mathcal{R}$ sends a random d and returns $\perp$. $\mathcal{T}_{\texttt{ID}}$ checks if $b = d$ and returns $\texttt{ok}$ if this is the case and $\perp$ otherwise.

Attacking the protocol. Theorem 4 of [12] claims that the protocol shown in Figure 2 provides mutual authentication (Definitions 4 and 5) and narrow-strong privacy (Definition 6). Note that the proof of reader authentication given in [12] (which we believe to be correct) requires the encryption scheme to be CCA-secure and $2^{-\beta}$ to be negligible. In the following we show that the protocol in Figure 2 is not narrow-strong private.

Theorem 1. *The RFID protocol shown in Figure 2 does not achieve narrow-strong privacy (Definition 6), reader authentication (Definition 5) and correctness (Definition 2) at the same time.*

The full proof of Theorem 1 can be found in the full version of this paper [31].

Proof (Theorem 1, Sketch). The idea of the proof is to construct a narrow-strong adversary $\mathcal{A}_{\text{prv}}$ that violates Definition 6 by corrupting a legitimate tag $\mathcal{T}_{\texttt{ID}}$ during the Ident protocol. More precisely, $\mathcal{A}_{\text{prv}}$ corrupts $\mathcal{T}_{\texttt{ID}}$ right *before* it receives d, which authenticates $\mathcal{R}$ to $\mathcal{T}_{\texttt{ID}}$. This allows $\mathcal{A}_{\text{prv}}$ to obtain the complete state $S = (pk_{\mathcal{R}}, \texttt{ID}, K, b)$ of $\mathcal{T}_{\texttt{ID}}$, including its temporary state b. Hence, $\mathcal{A}_{\text{prv}}$ can perform the computation $\mathcal{T}_{\texttt{ID}}$ would have done on receipt of d (i.e., $\mathcal{A}_{\text{prv}}$ can check whether $b = d$). In case $\mathcal{A}_{\text{prv}}$ interacted with the real oracles, then (due to correctness) with overwhelming probability this computation must result in acceptance of $\mathcal{R}$ (i.e., it must hold that $b = d$). However, if $\mathcal{A}_{\text{prv}}$ interacted with the blinder $\mathcal{B}$, then the computation done by $\mathcal{A}_{\text{prv}}$ leads to rejection of $\mathcal{R}$ (i.e., it holds that $b \neq d$) with overwhelming probability. This is due to reader authentication (Definition 5). Hence, $\mathcal{A}_{\text{prv}}$ can distinguish the real oracles from the simulation by $\mathcal{B}$ with non-negligible advantage, which violates narrow-strong privacy (Definition 6). □

5.2 Impossibility of Narrow-Forward-Privacy

Now we generalize the attack shown in Section 5.1. To prove our first impossibility result, we need the following lemma, which we will prove further below.

Lemma 1. *If for every narrow-forward adversary $\mathcal{A}_{\text{prv}}$ there is a blinder $\mathcal{B}$ such that $\mathbf{Adv}_{\mathcal{A}_{\text{prv}}}^{\text{prv}}$ is negligible (Definition 6), then $\mathcal{B}$ can be used to construct an adversary $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ such that* $\Pr[\mathbf{Exp}_{\mathcal{A}_{\text{sec}}^{\mathcal{B}}}^{\mathcal{R}\text{-aut}} = 1]$ *is non-negligible (Definition 5).*

Based on this lemma, we set up the following theorem, which we need later to prove our main impossibility result:

Theorem 2. *There is no RFID system (Definition 1) that achieves both reader authentication (Definition 5) and narrow-forward privacy (Definition 6).*

Proof (Theorem 2). Let $\mathcal{A}_{\text{prv}}$ be a narrow-forward adversary (Definition 3). Definition 6 requires the existence of a blinder $\mathcal{B}$ such that $\mathcal{A}_{\text{prv}}$ cannot distinguish between $\mathcal{B}$ and the real oracles. From Lemma 1 it follows that $\mathcal{B}$ can be used to impersonate $\mathcal{R}$ to any legitimate tag $\mathcal{T}$ with non-negligible probability. Hence, the existence of $\mathcal{B}$ contradicts reader authentication (Definition 5). □

Proof (Lemma 1). First, we show how to construct $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ from $\mathcal{B}$. Second, we prove that $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ violates reader authentication (Definition 5) if $\mathbf{Adv}_{\mathcal{A}_{\text{prv}}}^{\text{prv}}$ is negligible for every narrow-forward $\mathcal{A}_{\text{prv}}$ (Definition 3).

Let $q_{\mathcal{R}} \in \mathbb{N}$ with $q_{\mathcal{R}} > 0$ be the (expected) number of $\mathsf{SendReader}$ queries as specified by the Ident protocol and let $S_i^{\mathcal{R}}$ be the state of $\mathcal{R}$ after processing the i-th $\mathsf{SendReader}$ query. The initial reader state $S_0^{\mathcal{R}}$ includes the public key $pk_{\mathcal{R}}$ and the secret key $sk_{\mathcal{R}}$ of $\mathcal{R}$ as well as a pointer to the credentials database $\mathtt{DB}$. Note that during the processing of a $\mathsf{SendReader}$ query, $\mathcal{R}$ can update $\mathtt{DB}$. $\mathcal{R}$ can be considered as a tuple of algorithms $(\mathcal{R}_\pi^{(1)}, \ldots, \mathcal{R}_\pi^{(q_{\mathcal{R}})})$, where $\mathcal{R}_\pi^{(i)}$ represents the computation done by $\mathcal{R}$ when processing the i-th $\mathsf{SendReader}$ query in instance π of the Ident protocol. More formally: $(S_1^{\mathcal{R}}, m_1) \leftarrow \mathcal{R}_\pi^{(0)}(S_0^{\mathcal{R}})$ and $(S_{i+1}^{\mathcal{R}}, m_{2i+1}) \leftarrow \mathcal{R}_\pi^{(i)}(S_i^{\mathcal{R}}, m_{2i})$ for $1 \leq i \leq q_{\mathcal{R}}$. Since tags are passive devices that cannot initiate communication $\mathcal{R}$ must send the first protocol message. Thus, $\mathcal{R}$ generates all protocol messages with odd indices whereas the tag $\mathcal{T}$ generates all messages with even indices. In case the Ident protocol specifies that $\mathcal{T}$ sends the last protocol message, then $m_{2q_{\mathcal{R}}+1}$ is the empty string. Let $q_{\mathcal{T}} \in \mathbb{N}$ with $q_{\mathcal{T}} > 0$ be the (expected) number of $\mathsf{SendTag}$ queries as specified by the Ident protocol and let $S_i^{\mathcal{T}}$ be the state of $\mathcal{T}$ after processing the i-th $\mathsf{SendTag}$ query. $\mathcal{T}$ can be represented as a tuple of algorithms $(\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(q_{\mathcal{T}})})$ where $\mathcal{T}^{(i)}$ means the computation done by $\mathcal{T}$ when processing the i-th $\mathsf{SendTag}$ query in an instance of the Ident protocol that involves $\mathcal{T}$. More formally: $(S_{i+1}^{\mathcal{T}}, m_{2i}) \leftarrow \mathcal{T}^{(i)}(S_i^{\mathcal{T}}, m_{2i-1})$ for $1 \leq i \leq q_{\mathcal{T}}$. Note that $m_{2q_{\mathcal{T}}}$ is the empty string if Ident specifies that $\mathcal{R}$ must send the last protocol message.

Alg. 1. Adversary $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ violating reader authentication

```
1: $\mathsf{CreateTag}(\mathtt{ID})$
2: $vtag \leftarrow \mathsf{Draw}(\Pr[\mathtt{ID}] = 1)$
3: $\pi \leftarrow \mathsf{Launch}()$                                          $\triangleright$ simulated by $\mathcal{B}$
4: $m_1 \leftarrow \mathsf{SendReader}(-, \pi)$                                $\triangleright$ simulated by $\mathcal{B}$
5: $i \leftarrow 1$
6: while $i < q_{\mathcal{R}}$ do
7:     if $i \leq q_{\mathcal{T}}$ then $m_{2i} \leftarrow \mathsf{SendTag}(m_{2i-1}, vtag)$    $\triangleright$ simulated by $\mathcal{B}$
8:     end if
9:     $m_{2i+1} \leftarrow \mathsf{SendReader}(m_{2i}, \pi)$                  $\triangleright$ simulated by $\mathcal{B}$
10:    $i \leftarrow i + 1$
11: end while
12: $out_{\mathcal{T}_{\mathtt{ID}}} \leftarrow \mathsf{SendTag}(m_{2q_{\mathcal{R}}-1}, vtag)$    $\triangleright$ computed by $\mathcal{T}_{\mathtt{ID}}$
```

The idea of $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ is to internally use $\mathcal{B}$ as a black-box to simulate the final response of $\mathcal{R}$ that makes a legitimate tag $\mathcal{T}_{\mathtt{ID}}$ to accept $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ as $\mathcal{R}$. $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ is defined in Algorithm 1 and works as follows: First, $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ creates a legitimate tag $\mathcal{T}_{\mathtt{ID}}$ (step 1) and makes it accessible (step 2). Both steps are also shown to $\mathcal{B}$, which expects to observe all oracle queries. Then, $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ makes $\mathcal{B}$ to start a new instance π of the Ident protocol with $\mathcal{T}_{\mathtt{ID}}$ (step 3) and obtains the first protocol message m_1 generated by $\mathcal{B}$ (step 4). Now, $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ internally runs $\mathcal{B}$ that simulates *vtag* and $\mathcal{R}$ until $\mathcal{B}$ returns the final reader message $m_{2q_{\mathcal{R}}-1}$ (steps 5–11). Finally, $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ sends $m_{2q_{\mathcal{R}}-1}$ to the real tag $\mathcal{T}_{\mathtt{ID}}$ (step 12). $\mathcal{A}_{\text{sec}}^{\mathcal{B}}$ succeeds if $\mathcal{T}_{\mathtt{ID}}$ accepts $m_{2q_{\mathcal{R}}-1}$ and returns $out_{\mathcal{T}_{\mathtt{ID}}} = \mathtt{ok}$, which means that $\mathcal{T}_{\mathtt{ID}}$ accepts $\mathcal{B}$ as $\mathcal{R}$. More formally, this means that:

$$\Pr\left[\mathbf{Exp}_{\mathcal{A}_{\text{sec}}^{\mathcal{B}}}^{\mathcal{R}\text{-aut}} = 1\right] = \Pr\left[\mathsf{Ident}\left[\mathcal{T}_{\mathtt{ID}} : S_0^{\mathcal{T}_{\mathtt{ID}}};\ \mathcal{A}_{\text{sec}}^{\mathcal{B}} : -;\ * : pk_{\mathcal{R}}\right] \rightarrow \left[\mathcal{T}_{\mathtt{ID}} : \mathtt{ok};\ \mathcal{A}_{\text{sec}}^{\mathcal{B}} : \cdot\right]\right] \quad (1)$$

We stress that this indeed is a valid attack w.r.t. Definition 5 since $\mathcal{A}_{\text{sec}}$ neither corrupts $\mathcal{T}_{\mathtt{ID}}$ nor just forwards the protocol messages between $\mathcal{R}$ and $\mathcal{T}_{\mathtt{ID}}$.

Next, we show that narrow-forward privacy (Definition 6) ensures that Eq. 1 is non-negligible. Therefore, we assume by contradiction that Eq. 1 is negligible, which implies that with overwhelming probability $p_\perp$ message $m_{2q_{\mathcal{R}}-1}$ generated by $\mathcal{B}$ makes $\mathcal{T}_{\mathtt{ID}}$ to return $out_{\mathcal{T}_{\mathtt{ID}}} = \perp$. In the following, we show that if $p_\perp$ is non-negligible, then there is a narrow-forward adversary $\mathcal{A}_{\text{prv}}$ that has non-negligible advantage $\mathbf{Adv}_{\mathcal{A}_{\text{prv}}}^{\text{prv}}$, which contradicts narrow-forward privacy (Definition 6). $\mathcal{A}_{\text{prv}}$ is defined in Algorithm 2 and works as follows: First, $\mathcal{A}_{\text{prv}}$ creates a legitimate tag $\mathcal{T}_{\mathtt{ID}}$ (step 1) and makes it accessible (step 2). Then, $\mathcal{A}_{\text{prv}}$ makes $\mathcal{R}$ to start a new instance π of the Ident protocol with $\mathcal{T}_{\mathtt{ID}}$ (step 3) and obtains the first protocol message m_1 from $\mathcal{R}$ (step 4). Now, $\mathcal{A}_{\text{prv}}$ eavesdrops on the execution of the Ident protocol up to to the point *after* $\mathcal{R}$ has sent its last protocol message $m_{2q_{\mathcal{R}}-1}$ (steps 5–11) and corrupts $\mathcal{T}_{\mathtt{ID}}$ just *before* $\mathcal{T}_{\mathtt{ID}}$ received $m_{2q_{\mathcal{R}}-1}$ (step 12). Next, $\mathcal{A}_{\text{prv}}$ performs the computation $\mathcal{T}_{\mathtt{ID}}$ would have done on receipt of $m_{2q_{\mathcal{R}}-1}$ (step 13). If this computation results in $out_{\mathcal{T}_{\mathtt{ID}}} = \mathtt{ok}$, $\mathcal{A}_{\text{prv}}$ returns 0 to indicate that he interacted with the real oracles (step 14). Otherwise, $\mathcal{A}_{\text{prv}}$ indicates the presence of $\mathcal{B}$ by returning 1 (step 15). Note that $\mathcal{A}_{\text{prv}}$ indeed

Alg. 2. Narrow-forward adversary $\mathcal{A}_{\text{prv}}$

```
1: $\mathsf{CreateTag}(\mathtt{ID})$
2: $vtag \leftarrow \mathsf{Draw}(\Pr[\mathtt{ID}] = 1)$
3: $\pi \leftarrow \mathsf{Launch}()$
4: $m_1 \leftarrow \mathsf{SendReader}(-, \pi)$
5: $i \leftarrow 1$
6: while $i < q_{\mathcal{R}}$ do
7:     if $i \leq q_{\mathcal{T}}$ then $m_{2i} \leftarrow \mathsf{SendTag}(m_{2i-1}, vtag)$
8:     end if
9:     $m_{2i+1} \leftarrow \mathsf{SendReader}(m_{2i}, \pi)$
10:    $i \leftarrow i + 1$
11: end while
12: $S_{q_{\mathcal{R}}}^{\mathcal{T}_{\mathtt{ID}}} \leftarrow \mathsf{Corrupt}(vtag)$
13: $out_{\mathcal{T}_{\mathtt{ID}}} \leftarrow \mathcal{T}_{\mathtt{ID}}{}^{(q_{\mathcal{R}})}(S_{q_{\mathcal{R}}}^{\mathcal{T}_{\mathtt{ID}}}, m_{2q_{\mathcal{R}}-1})$
14: if $out_{\mathcal{T}_{\mathtt{ID}}} = \mathtt{ok}$ then return 0
15: else return 1
16: end if
```

is a narrow-forward adversary (Definition 3) since $\mathcal{A}_{\text{prv}}$ never queries Result and none of the oracles defined in Section 4.2 after corrupting $\mathcal{T}_{\mathtt{ID}}$.

Next, we determine $\mathbf{Adv}_{\mathcal{A}_{\text{prv}}}^{\text{prv}}$. Therefore, we first consider the case where $\mathcal{A}_{\text{prv}}$ interacts with the real oracles. Since $\mathcal{T}_{\mathtt{ID}}$ is legitimate, it follows from correctness (Definition 2) that $out_{\mathcal{T}_{\mathtt{ID}}} = \mathtt{ok}$ with overwhelming probability $p_{\mathtt{ok}}$. Hence,

$$\Pr\left[\mathbf{Exp}_{\mathcal{A}_{\text{prv}}}^{\text{prv-0}} = 1\right] = 1 - p_{\mathtt{ok}} \tag{2}$$

is negligible. Now, consider the case where $\mathcal{A}_{\text{prv}}$ interacts with $\mathcal{B}$. Note that by the contradicting hypothesis, $\mathcal{B}$ generates a protocol message $m_{2q_{\mathcal{R}}-1}$ that makes $\mathcal{T}_{\mathtt{ID}}$ to return $out_{\mathcal{T}_{\mathtt{ID}}} = \perp$ with overwhelming probability $p_{\perp}$. Thus, we have

$$\Pr\left[\mathbf{Exp}_{\mathcal{A}_{\text{prv}}}^{\text{prv-1}} = 1\right] = p_{\perp} \,. \tag{3}$$

From Eq. 2 and Eq. 3 it follows that $\mathbf{Adv}_{\mathcal{A}_{\text{prv}}}^{\text{prv}} = \left|1 - p_{\mathtt{ok}} - p_{\perp}\right|$. Note that both $p_{\mathtt{ok}}$ (due to correctness) and $p_{\perp}$ (by assumption) are overwhelming. Hence, $\mathbf{Adv}_{\mathcal{A}_{\text{prv}}}^{\text{prv}}$ is non-negligible, which contradicts narrow-forward privacy (Definition 6). In turn, this means that narrow-forward privacy ensures that Eq. 1 is non-negligible, which finishes the proof. □

Since the impossibility of narrow-forward privacy (see Theorem 2), implies the impossibility of all stronger privacy notions (see Figure 1), we have the following corollary, which corresponds to the first main claim of this paper.

Corollary 1. *In the PV-Model, there is no RFID system (Definition 1) that achieves both reader authentication (Definition 5) and any privacy notion that is different from weak and narrow-weak privacy (Definition 6) under temporary state disclosure.*

6 Corruption without Temporary State Disclosure

Our first impossibility result shows that the PV-Model requires further assumptions to evaluate the privacy properties of RFID systems where tag corruption is of concern. A natural question therefore is, whether one can achieve mutual authentication along with some form of privacy, if the temporary tag state is *not* disclosed. Hence, in this section we consider the case where corruption *only* reveals the persistent tag state but *no* information on the temporary tag state.

The attack and the impossibility result shown in Section 5 critically use the fact that in the PV-Model an adversary $\mathcal{A}_{\text{prv}}$ can learn the temporary state of a tag during the Ident protocol. This allows $\mathcal{A}_{\text{prv}}$ to verify the response of $\mathcal{R}$ (that may have been simulated by $\mathcal{B}$) and hence, due to reader authentication (Definition 5), $\mathcal{A}_{\text{prv}}$ can distinguish with non-negligible advantage between the real oracles and $\mathcal{B}$. However, if an adversary cannot obtain temporary tag states, he cannot perform this verification. Hence, the impossibility result we proved in Section 5 does not necessarily hold if the temporary state is safe to corruption.

6.1 Privacy under Corruption without Temporary State Disclosure

To show that it is possible to achieve a notion of privacy in the PV-Model that captures adversaries who can corrupt tags, we show that the protocol depicted in Figure 2 achieves narrow-forward privacy if corruption *only* reveals the persistent tag state but *no* information on the temporary tag state. Note that the attack presented in Section 5.1 cannot be applied since we now consider the case that the adversary cannot obtain the temporary tag state b.

Theorem 3. *The RFID protocol shown in Figure 2 achieves narrow-forward privacy (Definition 6) if the underlying encryption scheme is CCA-secure and* Corrupt *does not reveal the temporary tag state b.*

The full proof of Theorem 3 is given in the full version of this paper [31].

Proof (Theorem 3, Sketch). We construct a blinder $\mathcal{B}$ and show that $\mathbf{Adv}^{\text{prv}}_{\mathcal{A}_{\text{prv}}}$ is negligible for any narrow-forward adversary $\mathcal{A}_{\text{prv}}$, as required by Definition 6. More precisely, we prove that if there is an $\mathcal{A}_{\text{prv}}$ such that $\mathbf{Adv}^{\text{prv}}_{\mathcal{A}_{\text{prv}}}$ is non-negligible, then $\mathcal{A}_{\text{prv}}$ can be used as a black-box to construct an adversary $\mathcal{A}_{\text{cca}}$ that violates the CCA-security of the underlying encryption scheme. □

6.2 Impossibility of Narrow-Strong Privacy

We now point out a second, conceptually different weakness of the claimed narrow-strong private protocol of [12] (which is depicted in Figure 2). More precisely, we show an attack on this protocol that does not require the adversary to obtain temporary tag states. Moreover, we generalize this attack to prove our second impossibility result: in the PV-Model, it is *impossible* to achieve narrow-strong privacy along with reader authentication. This means that even in case the adversary cannot obtain temporary tag states, the most challenging privacy notion defined in [12] (narrow-strong privacy) remains unachievable.[4] We stress

[4] The impossibility of strong privacy has been shown in [11].

Alg. 3. Narrow-strong adversary $\mathcal{A}_{\text{prv}}$ on the protocol in Figure 2

```
1: $\mathsf{CreateTag}(\mathtt{ID})$
2: $vtag \leftarrow \mathsf{Draw}(\Pr[\mathtt{ID}] = 1)$
3: $(pk_{\mathcal{R}}, \mathtt{ID}, K) \leftarrow \mathsf{Corrupt}(vtag)$
4: $\mathsf{Free}(vtag)$
5: $\pi \leftarrow \mathsf{Launch}(\,)$
6: $a \leftarrow \mathsf{SendReader}(-, \pi)$
7: $b \in_R \{0,1\}^{\beta}$
8: $c \leftarrow \mathsf{Enc}_{pk_{\mathcal{R}}}(\mathtt{ID}, K, a, b)$
9: $d \leftarrow \mathsf{SendReader}(c, \pi)$
10: if $b = d$ then return 0
11: else return 1
12: end if
```

that introducing even stronger hardware assumptions to further restrict the ability of the adversary to corrupt tags would deviate from the capabilities of real tags. Indeed, most RFID tags are low-cost devices that usually are not equipped with mechanisms that ensure tamper-evidence or tamper-resistance.

Theorem 4. *The RFID protocol shown in Figure 2 does not achieve narrow-strong privacy (Definition 6), reader authentication (Definition 5) and correctness (Definition 2) at the same time.*

Note that the proof of reader authentication given in [12] (which we believe to be correct) requires the underlying public-key encryption scheme to be CCA-secure.

Proof (Theorem 4, Sketch). The idea is that a narrow-*strong* adversary $\mathcal{A}_{\text{prv}}$ can detect the presence of a blinder $\mathcal{B}$ by simulating a corrupt legitimate tag $\mathcal{T}_{\mathtt{ID}}$ to $\mathcal{R}$. In contrast to $\mathcal{R}$, $\mathcal{B}$ does not know the secret decryption key $sk_{\mathcal{R}}$ and thus will fail to generate a correct response d with overwhelming probability. This allows $\mathcal{A}_{\text{prv}}$ to distinguish between $\mathcal{B}$ and the real oracles. Note that in the following attack $\mathcal{A}_{\text{prv}}$ corrupts $\mathcal{T}_{\mathtt{ID}}$ *before* executing Ident with $\mathcal{R}$ and hence, the attack is *independent* of the temporary state of $\mathcal{T}_{\mathtt{ID}}$. $\mathcal{A}_{\text{prv}}$ is defined in Algorithm 3: First, $\mathcal{A}_{\text{prv}}$ creates a legitimate tag $\mathcal{T}_{\mathtt{ID}}$ (step 1), makes it accessible (step 2), corrupts it (step 3), and makes it inaccessible again (step 4). Then, $\mathcal{A}_{\text{prv}}$ makes $\mathcal{R}$ to start Ident with $\mathcal{T}_{\mathtt{ID}}$ (step 5) and obtains a from $\mathcal{R}$ (step 6). Now, $\mathcal{A}_{\text{prv}}$ simulates $\mathcal{T}_{\mathtt{ID}}$ to compute c (step 7–8) and sends c to $\mathcal{R}$ to obtain d (step 9). Next, $\mathcal{A}_{\text{prv}}$ performs the computation $\mathcal{T}_{\mathtt{ID}}$ would have done on receipt of d (step 10), i.e., $\mathcal{A}_{\text{prv}}$ checks if $b = d$. Finally, $\mathcal{A}_{\text{prv}}$ returns 0 to indicate that it interacted with the real oracles (step 10), or 1 to indicate the presence of $\mathcal{B}$ (step 11).

Since $\mathcal{T}_{\mathtt{ID}}$ is legitimate, due to correctness (Definition 2), $\mathsf{SendReader}$ must respond with $d = b$. The real $\mathsf{SendReader}$ oracle can compute b with probability 1 by decrypting c with $sk_{\mathcal{R}}$. Since, due to reader authentication (Definition 5), the public-key encryption scheme is assumed to be CCA-secure and $\mathcal{B}$ does not know $sk_{\mathcal{R}}$, $\mathcal{B}$ can at most guess b with negligible probability. Hence, $\mathcal{A}_{\text{prv}}$ has non-negligible advantage of distinguishing between $\mathcal{B}$ and the real oracles, which violates narrow-strong privacy (Definition 6). □

Now, we generalize this attack to our second impossibility result:

Theorem 5. *In the PV-Model there is no RFID system (Definition 1) that fulfills both reader authentication (Definition 5) and narrow-strong privacy (Definition 6).*

The full proof of Theorem 5 can be found in the full version of this paper [31].

Proof (Theorem 5, Sketch). We show that if there is a blinder $\mathcal{B}$ such that advantage $\mathbf{Adv}^{\text{prv}}_{\mathcal{A}_{\text{prv}}}$ is negligible for every narrow-strong adversary $\mathcal{A}_{\text{prv}}$ (as required by Definition 6), then we can use $\mathcal{B}$ as a black-box to construct an adversary $\mathcal{A}_{\text{sec}}$, who violates reader authentication (Definition 5). Note that $\mathcal{A}_{\text{prv}}$ can interact with tags after corrupting them since $\mathcal{A}_{\text{prv}}$ is a narrow-*strong* adversary. Hence, $\mathcal{A}_{\text{prv}}$ can corrupt a tag $\mathcal{T}$ after its creation and simulate it to $\mathcal{R}$ (that might be simulated by $\mathcal{B}$). This allows $\mathcal{A}_{\text{prv}}$ to verify the authentication messages of $\mathcal{R}$. Hence, $\mathcal{A}_{\text{prv}}$ can detect $\mathcal{B}$ since, due to reader authentication (Definition 5), $\mathcal{B}$ should not be able to successfully authenticate to $\mathcal{T}$ as $\mathcal{R}$. □

7 Conclusion

In this work we proved impossibility results that show that the RFID model proposed by Paise and Vaudenay [12] cannot guarantee the most interesting privacy notions and simultaneously reader authentication (which is the goal of the model). Nevertheless, we pointed out that, by restricting the tag corruption ability of the adversary, at least some, although weak, privacy notions can be achieved.

Acknowledgments. We thank Paolo D'Arco and Alessandra Scafuro for several useful discussions about RFID privacy notions. This work has been supported in part by the European Commission through the FP7 programme under contract 216646 ECRYPT II, 238811 UNIQUE, and 215270 FRONTS, in part by the Ateneo Italo-Tedesco under Program Vigoni and by the MIUR Project PRIN 2008 "PEPPER: Privacy E Protezione di dati PERsonali" (prot. 2008SY2PH4).

References

1. Atmel Corporation: Innovative IDIC solutions (2007), http://www.atmel.com/dyn/resources/prod_documents/doc4602.pdf
2. NXP Semiconductors: MIFARE smartcard ICs (September 2008), http://www.mifare.net/products/smartcardics/
3. Sadeghi, A.R., Visconti, I., Wachsmann, C.: User privacy in transport systems based on RFID e-tickets. In: International Workshop on Privacy in Location-Based Applications, PiLBA (2008)
4. I.C.A. Organization: Machine Readable Travel Documents, Doc 9303, Part 1 Machine Readable Passports, 5th Edn. (2003)

5. Weis, S.A., Sarma, S.E., Rivest, R.L., Engels, D.W.: Security and privacy aspects of low-cost radio frequency identification systems. In: Hutter, D., Müller, G., Stephan, W., Ullmann, M. (eds.) Security in Pervasive Computing. LNCS, vol. 2802, pp. 50–59. Springer, Heidelberg (2004)
6. Juels, A.: RFID security and privacy: A research survey. Journal of Selected Areas in Communication 24(2), 381–395 (2006)
7. Sadeghi, A.R., Visconti, I., Wachsmann, C.: Location privacy in RFID applications. In: Bettini, C., Jajodia, S., Samarati, P., Wang, X.S. (eds.) Privacy in Location-Based Applications. LNCS, vol. 5599, pp. 127–150. Springer, Heidelberg (2009)
8. Avoine, G.: Adversarial model for radio frequency identification. ePrint, Report 2005/049 (2005)
9. Juels, A., Weis, S.A.: Defining strong privacy for RFID. In: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '07), pp. 342–347. ACM Press, New York (2007)
10. Burmester, M., van Le, T., de Medeiros, B.: Universally composable and forward-secure RFID authentication and authenticated key exchange. In: Proc. of ASIACCS, pp. 242–252. ACM Press, New York (2007)
11. Vaudenay, S.: On privacy models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)
12. Paise, R.I., Vaudenay, S.: Mutual authentication in RFID: Security and privacy. In: Proc. of ASIACCS, pp. 292–299. ACM Press, New York (2008)
13. Deng, R.H., Li, Y., Yao, A.C., Yung, M., Zhao, Y.: A new framework for RFID privacy. ePrint, Report 2010/059 (2010)
14. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)
15. Hutter, M., Schmidt, J.M., Plos, T.: RFID and its vulnerability to faults. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 363–379. Springer, Heidelberg (2008)
16. Kasper, T., Oswald, D., Paar, C.: New methods for cost-effective side-channel attacks on cryptographic RFIDs. In: Workshop on RFID Security, RFIDSec (2009)
17. D'Arco, P., Scafuro, A., Visconti, I.: Semi-destructive privacy in DoS-enabled RFID systems. In: Workshop on RFID Security, RFIDSec (2009)
18. D'Arco, P., Scafuro, A., Visconti, I.: Revisiting DoS Attacks and Privacy in RFIDEnabled Networks. In: Dolev, S. (ed.) ALGOSENSORS 2009. LNCS, vol. 5804, pp. 76–87. Springer, Heidelberg (2009)
19. Ng, C.Y., Susilo, W., Mu, Y., Safavi-Naini, R.: RFID privacy models revisited. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 251–266. Springer, Heidelberg (2008)
20. Ng, C.Y., Susilo, W., Mu, Y., Safavi-Naini, R.: New privacy results on synchronized RFID authentication protocols against tag tracing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 321–336. Springer, Heidelberg (2009)
21. Bringer, J., Chabanne, H., Icart, T.: Efficient zero-knowledge identification schemes which respect privacy. In: Proceedings of ASIACCS '09, pp. 195–205. ACM Press, New York (2009)
22. Sadeghi, A.R., Visconti, I., Wachsmann, C.: Efficient RFID security and privacy with anonymizers. In: Workshop on RFID Security, RFIDSec (2009)
23. Sadeghi, A.R., Visconti, I., Wachsmann, C.: Anonymizer-enabled security and privacy for RFID. In: Miyaji, A., Echizen, I., Okamoto, T. (eds.) CANS 2009. LNCS, vol. 5888, pp. 134–153. Springer, Heidelberg (2009)

24. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game—A Completeness Theorem for Protocols with Honest Majority. In: Proc. of ACMSTOC, pp. 218–229 (1987)
25. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. SIAM J. on Computing 18(6), 186–208 (1989)
26. Sadeghi, A.R., Visconti, I., Wachsmann, C.: PUF-enhanced RFID security and privacy. In: Workshop on Secure Component and System Identification (SECSI) (2010)
27. Kirschenbaum, I., Wool, A.: How to build a low-cost, extended-range RFID skimmer. ePrint, Report 2006/054 (2006)
28. Avoine, G., Lauradoux, C., Martin, T.: When compromised readers meet RFID. In: Workshop on RFID Security (RFIDSec) (2009)
29. Garcia, F.D., van Rossum, P.: Modeling privacy for off-line RFID systems. In: Workshop on RFID Security (RFIDSec) (2009)
30. Nithyanand, R., Tsudik, G., Uzun, E.: Readers behaving badly: Reader revocation in PKI-based RFID systems. Cryptology ePrint Archive, Report 2009/465 (2009)
31. Sadeghi, A.R., Visconti, I., Wachsmann, C.: On rfid privacy with mutual authentication and tag corruption — Extended Version. ePrint (2010)

Social Network-Based Botnet Command-and-Control: Emerging Threats and Countermeasures

Erhan J. Kartaltepe[1], Jose Andre Morales[1], Shouhuai Xu[2], and Ravi Sandhu[1]

[1] Institute for Cyber Security, University of Texas at San Antonio
{erhan.kartaltepe,jose.morales,ravi.sandhu}@utsa.edu
[2] Department of Computer Science, University of Texas at San Antonio
shxu@cs.utsa.edu

Abstract. Botnets have become a major threat in cyberspace. In order to effectively combat botnets, we need to understand a botnet's Command-and-Control (C&C), which is challenging because C&C strategies and methods evolve rapidly. Very recently, botmasters have begun to exploit social network websites (e.g., `Twitter.com`) as their C&C infrastructures, which turns out to be quite stealthy because it is hard to distinguish the C&C activities from the normal social networking traffic. In this paper, we study the problem of using social networks as botnet C&C infrastructures. Treating as a starting point the current generation of social network-based botnet C&C, we envision the evolution of such C&C methods and explore social networks-based countermeasures.

Keywords: Botnet, command-and-control, social networks, security.

1 Introduction

The critical difference between botnets and other malware is that botmasters use a Command-and-Control (C&C) to coordinate large numbers of individual bots (i.e., compromised computers) to launch potentially much more damaging attacks. Botmasters also evolve their C&C strategies and methods very rapidly to evade defenders' countermeasures. Therefore, from a defender's perspective, it is always important to understand the trends and practices of botnet C&C [8,15,19,23,29,11,26]. Previous studies have mainly focused on two approaches: *host-centric* [31,36] and *network-centric* [16,18,17,3,7,8,14,23,20,6]. The host-centric approach aims to detect suspicious host activities, such as the use of incoming network data as system call arguments. The network-centric approach attempts to detect suspicious network activities by (for example) identifying network traffic patterns. The fact that a social network-based botnet C&C on `Twitter.com` was detected by "digging around" [25] suggests that we need to pursue more detection approaches.

Our contributions. Through an *application-centric* approach, we study the problem of botnets that use social network websites as their C&C infrastructures. First, we characterize the current-generation of social network-based botnet C&C, describing their strengths and weaknesses. Our characterization, while inspired by [25], is broader and deeper. Second, we envision how current social network-based botnet C&C might evolve in the near future, which capitalize on their strengths while diminishing their weaknesses. Third, we explore countermeasures for dealing with both current and future generations of social network-based botnet C&C. Since social network providers as well as client machines are victims of a social network-based botnet C&C, both server-side and client-side countermeasures are demonstrated and tested for both effectiveness and performance. Fourth, we discuss the limitations of the application-centric approach demonstrated in this paper, which suggests the need to integrate it with the aforementioned host-centric and network-centric methods because the three approaches are complementary to each other.

Paper outline. Section 2 discusses related prior work. Section 3 presents a characterization of the current generation of social network-based botnet C&C. Section 4 envisions the next-generation of social network-based botnet C&C. Sections 5 and 6 investigate server-end and client-end solutions to detecting social network-based botnet C&C, respectively. Section 7 discusses how to integrate them and how they can benefit from host-centric and network-centric approaches. Section 8 describes future work and concludes the paper.

2 Related Work

Network-centric approach. This approach aims to detect botnets by correlating the network traffic of a computer population, including destination IP addresses, server names, packet content, event sequences, crowd responses, protocol graphs and spatial-temporal relationships [16,18,17,3,7,8,14,23,20,6]. This approach is especially useful when only network traffic data is available.

Host-centric approach. This approach aims to differentiate malicious from benign processes running on host computers by observing that bot processes often use data received from the network as parameters in system calls [31]. A detection technique based on a high rate of failed connection attempts to remote hosts was recently presented in [36], which does not necessarily apply to the type of botnets we consider in the present paper because the connections are to popular social networking sites and are generally successful. This approach often looks deeply into the software stack [36].

Application-centric approach. This approach looks into the application-specific interactions. Previously, the focus has been put on IRC-based botnets (see, e.g., [14]). Recently, the possibility of exploiting emails as a botnet C&C was investigated [30], and the feasibility of detecting such botnets through their resulting spam traffic was presented in [35,34,37,22]. In this paper we consider a specific class of applications, namely web-based social networking. The concept of social network-based botnet C&Cs can be dated back to 2007 [21,12,28,13], but such botnets became a reality only very recently [2,25]. In particular, [25] served as the starting point of the present study. It should be noted that the focus of the present paper is only remotely related to the abuse of social networks for other purposes [1].

3 Characterizing Current Social Network-Based Botnet

How Does Current Social Network-Based Botnet C&C Work? Jose Narario first reported the actual use of social network as a botnet C&C [25], although the concept had been proposed as early as 2007 [21,28]. We call such a bot Naz, after its discoverer. At a high-level, Naz's botnet used accounts with the name upd4t3 owned by the botmaster on social network sites Twitter.com and Jaiku.com. These bots received Base64-encoded commands from these accounts to download malicious payloads onto the victimized bot computers. Since then, other C&Cs were discovered with variations of the botnet's scheme, such as the Twitter.com account Botn3tControl, which was shutdown days later.

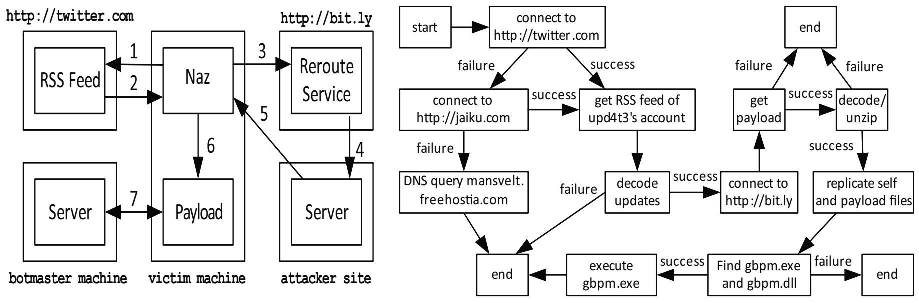


Fig. 1. Naz's C&C attack flow (left) and control flow (right)

To understand the behavior of Naz's botnet, we conducted two experiments. The first confirmed and extended Naz's C&C flow reported in [25]. Specifically, we obtained and ran a Naz sample on a machine by replacing its references to Twitter.com with a server under our control. This was necessary because the Twitter.com account was shutdown by Twitter.com's administrators shortly after its detection. We used our own Base64-encoded messages with a bit.ly URL we set up that redirected to the payload stored on our server. The payload was a Base64-encoded, compressed archive containing two files: gbpm.exe and gbpm.dll. These files were the originally identified payload package of Naz. In our analysis, we reconstructed the original C&C flows described below and depicted in the left-hand side of Figure 1.

1. The bot makes a HTTP GET request to upd4t3's RSS (Really Simple Syndication) feed at Twitter.com.
2. Twitter.com returns the RSS feed, containing Base64-encoded text in the description nodes.

3. The bot decodes the Base64-encoded text, revealing one or more `bit.ly` URLs, and makes a `HTTP GET` request to each. The `bit.ly` website provides short aliases for long URLs.
4. Each `bit.ly` URL redirects to a malicious zip file hosted on an independent attack server.
5. The bot downloads the malicious zip file as a payload.
6. The bot decodes and unzips the payload, replicates itself and the payload's uncompressed files, and then executes the payload's contents.
7. The payload attempts to gather user information from the victim computer and send it to a server selected by the botmaster.

To have a deeper understanding of the internal control flow of Naz, we conducted further black-box testing using data provided by Network Monitor [9] and CWSandbox [10], from which we draw the control flow details on the right-hand side of Figure 1. We observed that the bot made a copy of itself and the two files mentioned above in a temporary directory, and that when executing `gbpm.exe`, the bot dynamically injected code into `gbpm.exe`'s process. Moreover, we observed that Naz handled unexpected inputs as follows:

1. When we provided a URL to a bogus RSS feed, the bot failed to connect and instead attempted to access an RSS feed from a `Jaiku.com` account. This account name was hardwired in the bot program and had been deactivated by `Jaiku.com`'s administrator. This second connection failure led the bot to issue a DNS query on the domain name `mansvelt.freehostia.com`, after which the bot stopped producing network traffic. The site `mansvelt.freehostia.com` is currently unregistered and has no IP address.
2. When we placed plaintext sentences and URLs in our in-house RSS feed, the bot read the RSS feeds but did not show evidence of decoding and using the text in connection attempts.
3. When we modified the payload file to Base64-encoded only, compressed only, and replaced it with an executable file, the bot did not attempt to unzip or execute the file's contents. When we renamed the two payload files `gbpm.exe` and `gbpm.dll`, the bot did not attempt to execute the renamed `gbpm.exe` file, implying that the bot was program name sensitive.

Finally, it is interesting to note that a dynamic analysis of `gbpm.dll` revealed that the payload attempted to connect to a bank in Brazil. Moreover, both the analysis in [25] and our independent experiments demonstrate that Naz's botnet C&C serves primarily as a Trojan downloader [32].

Strengths of Naz's Botnet C&C. Naz's botnet C&C has the following advantages when compared with other botnet C&C infrastructures and methods:

1. Abusing trusted popular websites as a C&C server. Social networks and Web 2.0 sites such as `Twitter.com`, `FaceBook.com`, `LinkedIn.com`, and `YouTube.com` are not only legitimate, with verifiable SSL or EV-SSL certificates, but also heavily used by millions of users. Due to this heavy usage, light occasional traffic to one or more accounts is unlikely to be noticed

compared to a user's actual traffic pattern. This avoids any unnecessary and sometimes suspicious DNS queries (e.g., for non-popular DNS names).

2. Exploiting popular port(s) for C&C communication. Port 80 is the de facto standard for web traffic, and most network traffic will flow through it. This helps bots blend in with benign traffic.
3. Abusing application features for automatic C&C. The botmaster uses application features, such as RSS feeds, to automatically update bots. Moreover, the commands are so light-weight that they cannot be easily discerned from normal social network traffic.

The above discussion demonstrates that botmasters have begun to exploit "*hiding in plain sight*" to conduct stealthy botnet C&C. By piggybacking on the reputation and legitimacy of social network websites, botnet C&C activities may remain hidden, while defeating the "many eyes" defense [30].

Weaknesses of Naz's Botnet C&C. We need to understand the weaknesses of current generation of social network-based botnet C&Cs because these weaknesses will likely be absent in future generations. This is not meant to help the attackers, rather it is meant to help the defenders look ahead. Our examination shows that Naz's botnet C&C has weaknesses, which are omitted due to space limitation.

4 Envisioning Future Social Network-Based Botnet

In order to defeat future social network-based botnets, we must think ahead of the attackers. For this purpose, we can show how the aforementioned weaknesses of the current generation of social networks-based botnet C&Cs can be avoided. Due to space limitation, details are omitted.

5 Server-Side Countermeasures

5.1 The Detection Mechanism

A key observation behind our detection mechanism is that, regardless of the channel, provider, or account, social network messages are in text. Thus, if botmasters want to use social networks for their C&C, they would encode their commands textually. Moreover, just like legitimate messages may include web links, so might C&C messages (e.g., links for downloading payload). These observations inspired us to distinguish between encoded and plain texts and to follow unencoded links to their destination. Our detection mechanism can be adopted by the webserver as shown in Figure 2, and the resulting system would operate as follows (with steps 2 and 3 relevant to our countermeasure):

1. Alice logs into her social network and updates her status display using a content form.
2. The social network's content updater sends the text content to our server-end system.

3. The detection mechanism, which will be implemented as a classifier in our prototype system, determines if the text is suspicious and returns a result.
4. The content form updates the database with the message and whether it was marked suspicious.
5. Bob check's Alice content display, either through a feed like RSS or by visiting the site.
6. The content display requests the content from the database.
7. The database returns non-suspicious content; if a threshold level of suspicious messages (determined by policy) has been reached, the database returns a "suspicious account" message.
8. The content display shows those retrieved messages to the user, or a "suspicious account" message if the suspicious message threshold has been reached.

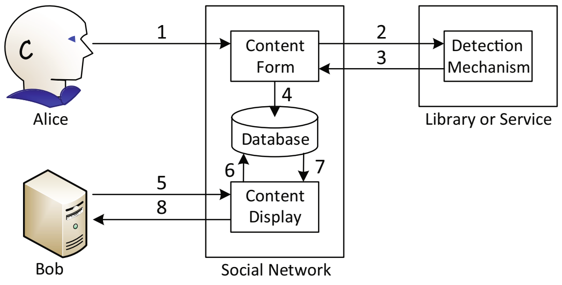


Fig. 2. Example scenario using our server-side detection mechanism

The server-side detection mechanism has the following advantages. First, it is *account agnostic* because it looks for text attributes that are shared with encoded text rather than individual behavioral patterns. Second, it is *language agnostic* because it looks at text for attributes that are shared with encoded text rather than individual words. As a result, the detection mechanism is effective for any language using Roman characters (English, Spanish, French, etc.). Third, it is *easy to deploy* because it can take advantage of light-weight machine learning algorithms and thus make decisions in real-time. Moreover, the code is easy to deploy as a library or software-as-a-service. Fourth, it can *follow unencoded links* to determine if the destination is a trusted source, say by using SSL authentication as a trust infrastructure. In the next two subsections we analyze the effectiveness and performance of our approach.

5.2 Prototype Implementation and Its Effectiveness and Limitations

Prototype implementation. To demonstrate the effectiveness of our system, we instantiated the detection mechanism as `Weka`'s [33] J48 decision tree algorithm (because it is quick and readily usable, but other tailored algorithms can be used instead in a plug-and-play fashion) to classify input messages so as to distinguish between Base64- or Hexadecimal-encoded text and natural language text. For links in the clear, by following links to their destination, we can mark the content as "suspicious" if it is an atypical file (e.g., an executable, library, encoded, or compressed file, or a combination of these). To build a pool of "non-suspicious" text, we screenscraped 200 `Twitter.com` accounts to build a list of 4000 messages. Our pool of bot commands were 400 short random commands of fifteen to thirty characters that were then encrypted using RC4 stream cipher and then encoded, giving a 10:1 set of normal to suspicious text. We then split the messages into a training set with 70% of both types and a test set with the remaining 30%. Recognizing that altering the natural Base64 or Hexadecimal alphabet with alternate characters such as spaces or punctuation could be used to obfuscate the text, we also ran our classifier against such alternate encoding schemes.

Effectiveness. For standard Base64 and Hexadecimal encoding schemes, our classifier was able to quickly distinguish between our "normal" and "suspicious" text samples in an account-agnostic way, no false positives and no false negatives, for both Base64 and Hexadecimal encoding (see Table 1). Moreover, our classifier maintained this accuracy even when the commands were obfuscated with other words—the distinctiveness of the encoded commands was readily apparent. The results were so perfect because the attributes we used—number of spaces in the text, longest word, and shortest word—cleanly divided the "normal" and "suspicious" text. To produce non-standard Base64 and Hexadecimal encoding schemes, we randomly swapped ten of the standard alphabet with alternate ones from a pool of space and punctuation characters. Our profiler was able to distinguish between them in an account-agnostic way, with a false positive rate of 0.0% and false negative rate of 1.25% for Alternate Base64 encodings, and a false positive rate of 3.25% and false negative rate of 12.5% for Alternate Hexadecimal encodings (see Table 1).

Table 1. Results with respect to various Base64 and Hexadecimal encodings

	Base64		**Hexadecimal**		**Alt. Base64**		**Alt. Hexadecimal**	
	Actual Positive	Actual Negative	Actual Positive	Actual Negative	Actual Positive	Actual Negative	Actual Positive	Actual Negative
Tested Positive	100%	0%	100%	0%	100%	1.25%	96.75%	12.5%
Tested Negative	0%	100%	0%	100%	0%	98.75%	3.25%	87.5%

The classifier's accuracy dropped significantly when the commands were obfuscated with other words, especially with Hexadecimal encoding schemes and with spaces as alternate characters. We note that with such an encoding mechanism, *a priori* knowledge of words or characters to excise from the message would be necessary to extract the non-command content from the meaningful botnet commands. This form of steganography is essentially indistinguishable from typical steganography, where a botmaster would hide the bot commands in such a way as to not attract attention to themselves, i.e., using natural language words as code for commands or URLs.

Limitations. Hiding commands in a social network-based C&C using steganography makes it difficult for programs or even humans to identify the presence of a command within a message. Since social network messages left by users are unstructured content, a crafty adversary can hide a bot command within a message in such a way that a human reading the message could not identify the message as a command. Combined with encryption, reverse analysis—even with a captured bot—may not yield the interpreted commands. Thus, running a steganographic reversal algorithm on a C&C message would not return data that was a clear bot command. [30]. However, our server-side solution coupled with a client-side counterpart that detects when a process is acting on input from a social network-based C&C would provide a complete solution to this emerging threat. In Section 7 we will discuss how these limitations may be overcome in a bigger solution framework.

5.3 Performance

Evaluation methodology and environment setup. In order to demonstrate the efficiency of our server-side detection mechanism, we measured the performance of our prototype. We implemented it into CompactSocial, a microblogging service that emulates the constraints of `Twitter.com`. CompactSocial provides a simple interface to both update a status message and view any account's messages using a web browser. Moreover, an auto-updated RSS feed contains the text of the last ten account updates. CompactSocial was written in Java 6, update 11 and ran as a web application deployed to Apache Tomcat 6.0.20. When used as a library, our server-end solution was deployed as a `.jar` file; when deployed as a service, the classifier ran as a stand-alone web application deployed to Apache Tomcat 6.0.20. The classifier and CompactSocial used shared cryptographic keys for authentication. For processing incoming and outgoing messages, Javas crypto library was used to compute any hash value or HMAC it needed, in both cases using the SHA1 algorithm.

The system environment is depicted in Figure 3. Both servers reside within a university campus network, and the CompactSocial clients are both within and outside the campus network. The CompactSocial and text-classifier servers are called `mercury` and `apollo`, respectively. There are two CompactSocial client machines: `minerva` acted as an external computer within the LAN with authorized access to `mercury` through a simple CompactSocial client, and `mars` was an adversary client machine within the campus network, employing Naz+ to read

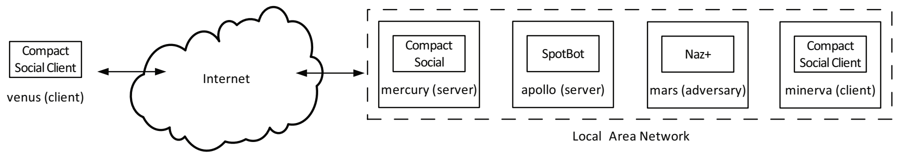


Fig. 3. Integrating the server-side solution into real life systems

updates made by minerva. A fifth machine, venus mimicked the minerva's functionality and tested the performance of the classifier on a non-dedicated internet connection. The three servers, hermes, jupiter, and euclid recognized each other by sharing some pair-wise keys. Table 2 reviews the concrete configurations of the machines and networks.

Table 2. System settings (all machines use Intel Core 2 Duo, 2.93 GHz processor)

Machine	**Internet Connection**	**Relevant Software**
mercury	Gigabit LAN	Apache Tomcat 6.0.20, Sun JVM, CompactSocial
apollo	Gigabit LAN	Apache Tomcat 6.0.20, Sun JVM, classifier service
minerva	Gigabit LAN	Firefox 3.5, CompactSocial client
mars	Gigabit LAN	.NET 3.5 Framework, Naz+
venus	100 Megabit Cable	Firefox 3.5, CompactSocial client

A CompactSocial client was developed in JavaScript to simulate a user updating their status in the CompactSocial web application. The CompactSocial client was developed as an addon and installed into Mozilla Firefox. Additionally, Naz+ was developed as a Windows service and written in C#, targeting the Microsoft .NET Framework, version 3.5. Naz+ periodically checked the RSS feed for CompactSocial test account, parsed the XML for the description node which contained the bot command as encrypted text (using a shared key with mars who updated the status message), and executed the command.

Performance benchmarking. To examine the delay incurred by the classifier when utilized by CompactSocial, time was marked before and after each transaction over 30000 requests at varying rates. We repeated this test ten times and took the average over the runs. Figure 4 shows the results over varying requests per second when the classifier was used as a library and a service.

When used as a library, our classifier performed roughly twice as fast than its service counterpart, since no network traffic was required. At even 500 requests per second, the classifier handled all requests without incident. In practice, a

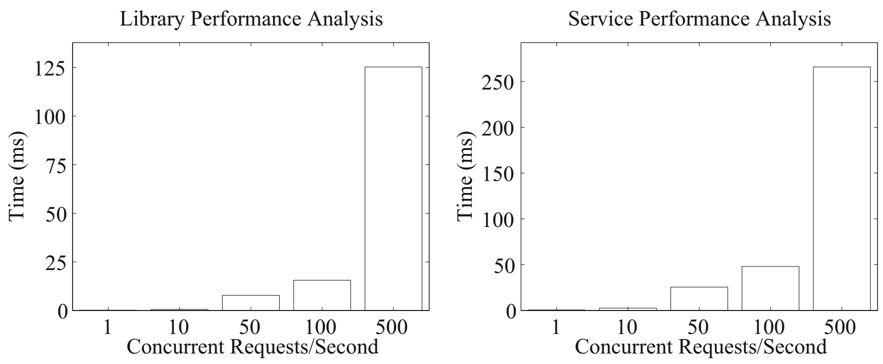


Fig. 4. Classifier library (left) and service (right) performance analysis

large-scale enterprise would use our server-side classifier as a service, whether as an in-house server to make requests or a pay-as-you-go service to a third party. In our preliminary testing, the classifier service handles 500 requests per second on a non-dedicated machine with cycles to spare. In the advent that a larger throughput was necessary, a load balancer can reduce the requests for a particular machine to a manageable level.

Meeting the needs of real-life systems. For a social network provider like `Twitter.com`, that has fifteen million users, and is increasing at roughly one million users a month, we wondered how our classifier would stack up. A large percentage (85.3%) of `Twitter.com` accounts post less than once a day, whether due to lost interest or not having much to post. We classify these accounts as "passive users". The vast majority (99.5%) of accounts post less than sixteen messages daily, which would be a class of "active" users, who post regularly about events. The remaining 0.5% post more than sixteen updates, although `Twitter.com` restricts accounts to a 1000 updates per day limit. These are particularly engaged users or are shared accounts by multiple people in an organization posting under a common account. We classify these users as "explosive" users [5].

If all accounts hit their ceilings on `Twitter.com`, we'd have the posting rates in Table 3. With fifteen million users, `Twitter.com` would handle a ceiling of 1410.7 messages per second (in actuality, most users do not hit these ceilings, so the actual threshold is far less). In this worst case, assuming the one million account per month growth rate and the same distribution of account usage, `Twitter.com` will accommodate an uptick per month of nearly 100 messages per second.

Table 3. `Twitter.com` usage analysis

15000000 users	**Percentage**	**Update Rate**	**Messages Per Day**	**Messages Per Second**
Passive users	85.3%	1/day	12795000	148.1
Active users	14.2%	16/day	34080000	394.5
Explosive users	0.5%	1000/day	75000000	868.1
Total users	100%	—	121875000	1410.7

Because our server-side approach is account agnostic, it does not need to build an account history for each user and as a result, would only need to check a few messages to determine if the account is being used in a suspicious way. Given the above scenario and a policy that checks periodically verifies one message daily and the first three messages for a new user's account, then our classifier would only need to check fifteen million messages per day, or 173.7 messages per second, with a monthly increase of 12.1 messages per second. If only active and explosive accounts were targeted (which would be more likely behavior for a botnet C&C), this would decrease to 25.6 messages per second with an increase of 2.1 messages per second. Thus, even with these simple non-discriminating policies and worst-case `Twitter.com` usage scenario, our classifier as a service can handle an enterprise-level throughput of requests, and different policy strategies may be employed to throttle down the throughput further.

6 Client-Side Countermeasures

6.1 The Detection Mechanism

Detection attributes. We propose detecting social network-based botnet C&Cs using three attributes: self-concealing, dubious network traffic, and unreliable provenance.

- Self-Concealing: A self-concealing process is one which attempts to avoid detection with the use of stealth mechanisms. We consider two specific instances of this type:

  Graphical User Interface. Many applications that read RSS feeds interact with a user via a graphical user interface. Bots and other malware will attempt to avoid detection, and as a result may run in the background as a service or hidden process without an explicit interface. A process without a graphical user interface can be identified as possibly self-concealing.

  Human Computer Interaction. Most benign software works by reacting to user input via a keyboard or mouse. Malware processes tend to run hidden and independent of user input and don't require explicit keyboard or mouse events provided by a user to perform a nefarious act. A process without human/computer interaction can be identified as possibly self-concealing.
- Dubious Network Traffic: A process with dubious network traffic is one which engages in network communication with another machine in a covert or devious way. We consider three specific instances of this type:

  Social Network Request. Exclusively visiting social networking sites is not suspicious; however, social network-based botnet C&C craftily abuse the popularity and good name of social networking sites; thus, exclusive requests to social networking or web 2.0 sites is considered a possible trigger event for dubious network traffic.

  Encoded Text Processing. Since social network-based bots read commands as encoded text, processes making connections to sources that provide encoded or encrypted text is anomalous. Accepting connections with encoded text and processing it by decoding or decrypting it can be considered as possibly dubious network traffic.

  Suspicious File Downloading. In general, applications do not download suspicious files such as executable, library, compressed, or encoded/encrypted files without permission (though they may download image or text files). Social network-based C&C bot processes, on the other hand, act as Trojan downloaders and almost exclusively save executables or DLLs to the filesystem as malicious payload. Downloading such suspicious files can be considered as dubious network traffic.
- Unreliable Provenance: A process with unreliable provenance is one which lacks a reliable origin. We consider three specific instances:

  Self-Reference Replication. This is a feature malware uses to survive disinfection on a host machine, occurring when a process copies itself into a newly created file or an existent file (by modifying it) on the file system [24]. An installed file with its installer not having a verified signature can be identified as possibly having an unreliable provenance.

Dynamic Code Injection. This is used by malware to insert malicious code into the memory space of an active process. Its end goal is to modify the process to perform nefarious deeds, possibly by piggybacking on that application's authorization settings. A process whose injector lacks a digital signature can be identified as possibly having an unreliable provenance since the injector's origin cannot be established.

Verifiable Digital Signature. Digital signatures may be considered a hallmark of trust between users and well established software. Most organizations that publish software provide a signature for their program and related files. Malware authors typically do not employ digital signatures; as a consequence, a process running without a verifiable digital signature can be identified as possibly having an unreliable provenance.

Detection model. We say a process P has the self-concealing attribute (P_{sc}) if it lacks a graphical user interface ($P_{gui} = \mathsf{false}$) and does not accept human computer interaction ($P_{hci} = \mathsf{false}$). More formally,

$$(\neg P_{gui}) \land (\neg P_{hci}) \rightarrow P_{sc}.$$

We say a process P has the dubious network traffic attribute (P_{dnt}) if it performs social network requests ($P_{snr} = \mathsf{true}$) and encoded text processing ($P_{etp} = \mathsf{true}$) or does suspicious file downloading ($P_{sfd} = \mathsf{true}$) (or both). More formally,

$$P_{snr} \land (P_{etp} \lor P_{sfd}) \rightarrow P_{dnt}.$$

We say a process P has the unreliable provenance attribute (P_{up}) if it performs self-reference replication ($P_{srr} = \mathsf{true}$) or does dynamic code injection ($P_{dci} = \mathsf{true}$), and also lacks a verified digital signature ($P_{vds} = \mathsf{false}$). More formally,

$$(P_{srr} \lor P_{dci}) \land (\neg P_{vds}) \rightarrow P_{up}.$$

Putting the above altogether, we classify a process P as being suspicious of being a social network-based bot C&C process (P_{snbb}) if it is either self-concealing ($P_{sc} = \mathsf{true}$) or has an unreliable provenance ($P_{up} = \mathsf{true}$) (or both), and engages in dubious network traffic ($P_{dnt} = \mathsf{true}$). More formally,

$$(P_{sc} \lor P_{up}) \land P_{dnt} \rightarrow P_{snbb}.$$

6.2 Effectiveness and Limitations

Evaluation methodology and environment setup. To examine the effectiveness of the detection model described above, we collected data with respect to our detection attributes for both benign and malicious processes in order to distinguish them. For this purpose, we considered eighteen benign applications,

Table 4. Client-side test set ("SN-Based Bot" stands for "Social Network-Based Bot")

Application	Type	Application	Type	Application	Type
AOL Explorer	Web Browser	Internet Explorer	Web Browser	RSS Bandit	RSS Aggregator
Avant	Web Browser	K-Meleon	Web Browser	RSS Owl	RSS Aggregator
Bobax	Traditional Bot	Maxthon	Web Browser	SeaMonkey	Web Browser
BlogBridge	RSS Aggregator	Mercury	RSS Aggregator	Snarfer	RSS Aggregator
FeedDemon	RSS Aggregator	Naz	SN-Based Bot	Tweetdeck	Twitter Client
FireFox	Web Browser	Naz+	SN-Based Bot	Twhirl	Twitter Client
Flock	Web Browser	Opera	Web Browser	Virut	Traditional Bot
Google Chrome	Web Browser	Ozdok	Traditional Bot	Waledac	Traditional Bot

four traditional bots, and the malicious Naz and prototype Naz+ bots (listed in Table 4). To provide a wide breadth, the benign applications are a broad selection of the most popular web browsers, RSS aggregators, Twitter clients, and RSS aggregators which read subscription feeds. Testing was performed using VMWare Workstation running Microsoft Windows SP3 using NAT for Internet access. Each application was executed separately for a period of four hours, followed by post-analysis. During testing of the eighteen benign applications, we interacted with each application by subscribing to and viewing different RSS feeds; attempting to subscribe to bogus RSS feeds, updating all RSS feeds every hour, reading individual feed articles. These tests were done to provide a wide range of expected and unexpected scenarios for each application to deal with while recording their behavior. In addition, we used a number of sensors to gather information about each process. Tracing of the three detection attributes described in Section 6.1 occurred from the moment a process starts executing.

Network traffic was collected using Windows Network Monitor [9]. Keyboard and mouse input was collected with a modified version of `GlobalHook`. Digital signatures were verified using SigCheck. Self-reference replication and dynamic code injection were accomplished with kernel hooks implementing known techniques [24]. The presence of a user interface was recorded by observing the creation of any window upon executing each application using `EasyHook`. User input, network traffic, graphical user interface interaction, self-reference replication and dynamic code injection were all recorded in real-time. For Virut and Waledac, static analysis and previous executions of these bots yielded their dubious network traffic results. Digital signatures were verified after the four hour testing of each process.

Effectiveness. The results are summarized in Table 5 and highlight some observations below. First, we observe that all benign applications lacked the self-concealing attribute as they all utilized a graphical user interface and accepted inputs from the user, such as reading an article, following a link, or updating an RSS feed. All bots but Virut demonstrated the self-concealing attribute since they did not have a graphical user interface or accept user input, although it appears that the command prompt window Virut displays may be accidental. Second, all applications but Naz and Naz+ possessed the dubious network traffic attribute; RSS applications did not download suspicious files, but all of the bots did. All bots but Virut read inordinate amounts of encoded text; legitimate RSS

Table 5. Client-Side detection results ("IE" stands for "Internet Explorer")

Application	Self-Concealing		Unreliable Provenance			Dubious Network Traffic			Result
	Graphical User Interface	Human Computer Interaction	Self-Reference Replication	Dynamic Code Injection	Verifiable Digital Signature	Social Network Request	Encoded Text Processing	Suspicious File Download	Social Network-Based Bot?
AOL Explorer	Y	Y	N	N	Y	N	N	N	N
Avant	Y	Y	N	N	Y	N	N	N	N
BlogBridge	Y	Y	N	N	Y	N	N	N	N
FeedReader	Y	Y	N	N	Y	N	N	N	N
Firefox	Y	Y	N	N	Y	N	N	N	N
Flock	Y	Y	N	N	Y	N	N	N	N
IE	Y	Y	N	N	Y	N	N	N	N
Chrome	Y	Y	N	N	Y	N	N	N	N
K-Meleon	Y	Y	N	N	Y	N	N	N	N
Maxthon	Y	Y	N	N	Y	N	N	N	N
Mercury	Y	Y	N	N	Y	N	N	N	N
Opera	Y	Y	N	N	Y	N	N	N	N
RSS Bandit	Y	Y	N	N	Y	N	N	N	N
RSS Owl	Y	Y	N	N	Y	N	N	N	N
SeaMonkey	Y	Y	N	N	Y	N	N	N	N
Snarfer	Y	Y	N	N	Y	N	N	N	N
Tweetdeck	Y	Y	N	N	N	Y	N	N	N
Twhirl	Y	Y	N	N	N	Y	N	N	N
Bobax	N	N	Y	Y	N	N	Y	Y	N
Ozdok	N	N	Y	Y	N	N	Y	Y	N
Virut	Y	N	Y	Y	N	N	N	Y	N
Waledac	N	N	Y	Y	N	N	Y	Y	N
Naz	N	N	Y	Y	N	Y	Y	Y	Y
Naz+	N	N	N	N	N	Y	Y	Y	Y

reader applications generally did not. Additionally, only Naz and Naz+ communicated nearly exclusively with social networking sites (benign processes read from them only a fraction of the time, and traditional bots made no such communication requests). Third, all benign applications did not exhibit the unreliable provenance attribute; none attempted to replicate itself or inject code into another process, and they all possessed a verifiable digital signature. On the other hand, all bots tested displayed this attribute, as they all copied themselves or injected code into other processes (except Naz+), and lacked a verifiable digital signature.

Of note is that no social network-based bot was misclassified as a benign process and no benign application or traditional bot was misclassified as a social network-based bot. We reiterate that our goal is to detect social network-based botnet C&C, which explains why the other bots Bobax, Ozdok, Virut and Waledac were not classified as social network-based bots. These bots may be detected by using complementary host-centric or network-centric approaches, or by applying the relaxed dubious network traffic attribute described above. This also justifies why the countermeasures presented in the paper need be integrated into a comprehensive defense framework.

Limitations. A limitation of our effectiveness analysis is the lack of real-time analysis of other social network-based botnet C&C, due to other botnets undiscovered in the wild. Moreover, our analysis is conducted in post-analysis. We plan to develop an implementation of this technique to provide real-time data gathering and evaluation. Another limitation is that any one sensor can be defeated if the malware author knows the concrete details of its implementation; knowledge of the high-level detection model is not enough. For example, if we only seek `.exe` and compressed files, a malicious file can purposely be renamed to `.jpg` or `.html` which would bypass our file download sensor. In this case, the bot's internal logic will have extra overhead, possibly checking every file in the network traffic with these file extension to identify the malicious one, thus making this approach infeasible. Malware in general are known to have attributes that trigger many of these sensors [32] and thus it is unlikely that a bot process will effectively work and bypass all our sensors.

6.3 Performance Analysis

In order to measure the client-side countermeasure's performance, we used PassMark's PerformanceTest 7.0 benchmarking to gather information on its CPU usage [27]. We benchmarked the baseline case with no data collection running, and then ran the data collector tracking zero to five processes after an hour of data collection had passed. Running the data collector added a 4.8% overhead to the overall system, and running it to track one to five processes had between a 13.3% and 28.9% overhead (See also Figure 5). With optimization, these numbers could be lowered further.

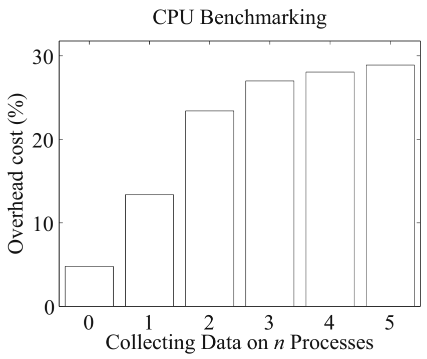


Fig. 5. CPU utilization for our client-side countermeasure process tracker

7 Integrating Server- and Client-Side Countermeasures

Integrated solution. As discussed above, the server-side and client-side countermeasures are integral parts of our client/server solution to detecting social network-based botnet C&C. When each type of countermeasures deployed alone, they can be defeated in certain ways. Integrating our server-side classifier into a social network webserver is straightforward whether it operates as a library or service, and integrating the client-side detection algorithms into existing malware detection schemes or operating its sensors as its own detection framework is equally as uncomplicated. Because both systems are stand-alone, there is no need to have the systems interoperate. Indeed, a user that has the client-side solution installed while using a social network that employs the server-side solution gains the benefits of both.

Limitations. Even when our classifier is utilized by a social network provider and a machine has our client solution installed, using both still has some limitations. Specifically, if the botmaster employs steganography into their social network-based C&C, the server-side solution in its current form will not detect that message being passed. Employing steganography in such a way will diffuse the content in the message, essentially expanding the text [4]. In this case, if using popular social networks like `Twitter.com` or `FaceBook.com` with character length limitations, spreading a command over multiple messages would likely be required. More specifically, our approach can benefit from host-centric and network-centric approaches as follows.

– **A bot that reads steganographic commands and can evade our client-side sensors.** One way for a bot to evade the client-side sensors is to exist at the kernel level. Since some of the client-side solution sensors exist at the user-level, the bot can effectively bypass enough of these sensors to mask its presence on the machine. Additionally, a bot that with intimate knowledge of the *implementation details* of the client-side sensors can maneuver around our countermeasures, such as writing code that falsifies sensor data. A host-centric approach to capture additional anomalous information at the kernel level would help mitigate this attack.

– **A bot that reads steganographic commands and masquerades as a benign process.** A bot that behaves as a benign process would have to lack the self-concealing or unreliable provenance attribute. By masquerading as a benign application, say by presenting itself as a graphical application that masks its true purpose, a bot could exist with such an interface. This bot would additionally have to trick the user into starting it and keeping it running, which might prove difficult. To avoid possessing an unreliable provenance, this bot would have to have a digital signature, which is difficult to forge. Additionally, it must not dynamically inject code into another source or replicate itself, which are hallmark signs of bots, since they wish to inculcate themselves into the host machine. A network-centric solution is necessary to analyze network layer data for similar events occurring from many machines in a network during a small timeframe.

– **A bot that reads steganographic commands and runs scripts.** A bot that behaves as a social network-based bot but downloads text files instead of executables will not be classified as a social network-based bot, although it would be marked as a possibly suspicious bot. If the bot contains or is aware of a scripting engine such as a Python interpreter, the bot can run the script instead of an executable. A host-centric approach to contain general purpose malware or prevent or alert the user of script/program execution would help stop this attack. Additionally, a network-centric strategy to detect script file downloads would help prevent the scripts from being downloaded to the client machine.

8 Conclusion and Future Work

We systematically studied a social network-based botnet and its C&C and discussed their future evolutions. We investigated, prototyped, and analyzed both server-side and client-side countermeasures, which are integral parts of a solution to the emerging threat of social network-based botnets. We also discussed how our solution can benefit from host-centric and network-centric botnet detection solutions so as to formulate a comprehensive defense against botnets.

Our future work includes: (1) implementing the client-side countermeasures as real-time detection systems, (2) improving the server-side classifier to detect steganography, (3) handling multiple stepping stones in payload redirection, and (4) and porting the client-side countermeasures to other platforms.

Acknowledgments. This work is partially supported by grants from AFOSR, ONR, AFOSR MURI, and the State of Texas Emerging Technology Fund.

References

1. Athanasopoulos, E., Makridakis, A., Antonatos, S., Antoniades, D., Ioannidis, S., Anagnostakis, K., Markatos, E.: Antisocial networks: Turning a social network into a botnet. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 146–160. Springer, Heidelberg (2008)
2. Balatzar, J., Costoya, J., Flores, R.: The real face of koobface: The largest web 2.0 botnet explained. Technical report, Trend Micro (2009)
3. Binkley, J.R., Singh, S.: An algorithm for anomaly-based botnet detection. In: Proc. Reducing Unwanted Traffic on the Internet, SRUTI '06 (2006)
4. Chapman, M., Davida, G.I.: Plausible deniability using automated linguistic stegonagraphy. In: Conference on Infrastructure Security (October 2002)
5. Cheng, A., Evans, M.: Inside twitter: An in-depth look inside the twitter world, `http://www.sysomos.com/insidetwitter`
6. Collins, M., Reiter, M.: Hit-list worm detection and bot identification in large networks using protocol graphs. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 276–295. Springer, Heidelberg (2007)
7. Collins, M., Shimeall, T., Faber, S., Janies, J., Weaver, R., De Shon, M., Kadane, J.: Using uncleanliness to predict future botnet addresses. In: Proc. IMC '07 (2007)
8. Cooke, E., Jahanian, F., McPherson, D.: The zombie roundup: understanding, detecting, and disrupting botnets. In: Proc. SRUTI '05 (2005)
9. Microsoft Corporation. Network monitor 3.3, `http://go.microsoft.com/fwlink/?LinkID=103158&clcid=0x409`
10. CWSandbox.org. Cwsandbox—behavior-based malware analysis, `http://www.cwsandbox.org`
11. Dagon, D., Gu, G., Lee, C., Lee, W.: A taxonomy of botnet structures. In: Choi, L., Paek, Y., Cho, S. (eds.) ACSAC 2007. LNCS, vol. 4697, Springer, Heidelberg (2007)
12. DigiNinja. Kreiosc2: Poc using twitter as its command and control channel, `http://www.digininja.org`
13. Easton, T., Johnson, K.: Social zombies. In: DEFCON '09 (2009)

14. Goebel, J., Holz, T.: Rishi: identify bot contaminated hosts by irc nickname evaluation. In: Proc. HotBots '07 (2007)
15. Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B., Dagon, D.: Peer-to-peer botnets: overview and case study. In: Proc. HotBots '07 (2007)
16. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Security '08 (2008)
17. Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: Detecting malware infection through ids-driven dialog correlation. In: USENIX Security '07 (2007)
18. Gu, G., Zhang, J., Lee, W.: BotSniffer: Detecting botnet command and control channels in network traffic. In: Proc. NDSS '08 (2008)
19. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.: Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In: LEET '08 (2008)
20. Hu, X., Knysz, M., Shin, K.G.: Rb-seeker: Auto-detection of redirection botnets. In: Proc. NDSS '09 (2009)
21. Finjan Software Inc. Web security trends report q4 2007. Technical report, Finjan Software Inc. (2007), `http://www.finjan.com/Content.aspx?id=827`
22. John, J., Moshchuk, A., Gribble, S., Krishnamurthy, A.: Studying spamming botnets using botlab. In: Proc. NSDI '09 (2009)
23. Karasaridis, A., Rexroad, B., Hoeflin, D.: Wide-scale botnet detection and characterization. In: Proc. HotBots '07 (2007)
24. Morales, J.A., Clarke, P.J., Deng, Y., Kibria, B.G.: Identification of file infecting viruses through detection of self-reference replication. Journal in Computer Virology (2008)
25. Nazario, J.: Twitter based botnet command and control (2009), `http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel`
26. Nazario, J., Holz, T.: As the net churns: Fast-flux botnet observations. In: Proc. MALWARE '08 (2008)
27. PassMark.com. Passmark performancetest 7.0, `http://www.passmark.com/products/pt.htm`
28. Poland, S.: How to create a twitter bot (2007), `http://blog.stevepoland.com/how-to-create-a-twitter-bot/`
29. Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: A multifaceted approach to understanding the botnet phenomenon. In: Proc. IMC '06 (2006)
30. Singh, K., Srivastava, A., Giffin, J., Lee, W.: Evaluating email's feasibility for botnet command and control. In: Proc. DSN
31. Stinson, E., Mitchell, J.C.: Characterizing bots' remote control behavior. In: Hämmerli, B.M., Sommer, R. (eds.) DIMVA 2007. LNCS, vol. 4579, pp. 89–108. Springer, Heidelberg (2007)
32. Szor, P.: The Art of Computer Virus Research and Defense. Symantec Press (2005)
33. Weka 3 data mining software, `http://www.cs.waikato.ac.nz/ml/weka/`
34. Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G., Osipkov, I.: Spamming botnets: signatures and characteristics. In: Proc. SIGCOMM '08, pp. 171–182 (2008)
35. Zhao, Y., Xie, Y., Yu, F., Ke, Q., Yu, Y., Chen, Y., Gillum, E.: Botgraph: large scale spamming botnet detection. In: Proc. NSDI '09 (2009)
36. Zhu, Z., Yegneswaran, V., Chen, Y.: Using failure information analysis to detect enterprise zombies. In: Proc. Securecomm '09 (2009)
37. Zhuang, L., Dunagan, J., Simon, D., Wang, H., Osipkov, I., Hulten, G., Tygar, J.: Characterizing botnets from email spam records. In: Proc. LEET '08 (2008)

COP: A Step toward Children Online Privacy

Wei Xu, Sencun Zhu, and Heng Xu

Pennsylvania State University
{wxx104,szhu}@cse.psu.edu, hxu@ist.psu.edu

Abstract. We propose COP, a client-side system for protecting children's online privacy and empowering parental control over children's information disclosure with little manual effort. COP is compliant with the Children's Online Privacy Protection Act (COPPA) in the United States and it implements acquisition of parental consent before any private information submitted online by children, e.g., registration to a Web service. Instead of restricting access to certain Web services or blocking sensitive data from websites, COP employs perturbation techniques over personal data with the goal of concealing the sensitive information while providing certain usability of the data to the websites. We address several challenges in the implementation of COP, e.g., perturbation of different types of data, parsing user input and retaining transparency to children without obstructing their normal Web surfing activities. We apply COP in registrations to 23 most popular websites. The results indicate COP's effectiveness as a privacy protection tool. We also discuss some potential security attacks against COP's design and provide our countermeasures.

1 Introduction

1.1 Background

The Internet has evolved into a platform for communicating, exchanging information, carrying out commerce, streaming media and social networking among many users. Children, being part of the Internet users, have been given unprecedented opportunities to communicate online with one another. Exposing themselves to the virtual world has caused great concerns, especially considering the growing cases of children's online abuses [1], online predators [2], online children pornography [3] and other such matters. Moreover, many operators of websites are also interested in collecting children's personal information such as their names, ages, email addresses and phone numbers for commercial purposes [4,5]. Release of such data jeopardizes children's privacy. According to a study [6] over U.S. census data and its follow-up work [7], the combination of gender, 5-digit ZIP code and full date of birth can uniquely or nearly uniquely identify 63% of the US population. Hence, it is not hard to imagine how much private information will be compromised once it falls into the hands of malicious parties.

Compared to adults, children are more vulnerable to threats like re-identification because they are not mature enough to realize the harm of privacy divulgence. A study shows that almost half of teens (47%) do not even worry about others using their personal information [8]. Besides, children are not sophisticated enough to protect themselves against such information leakage. For example, by exploiting the naivety of children, some websites lure children to online prizes in exchange for their personal information [9]. Without rational judgment of the websites, children intend to submit their private information to access certain Web services. Due to these inherent vulnerabilities, children's online privacy protection has become an imminent and challenging task.

To take a step toward children's online privacy, we focus on personal information gathering processes, a representative case of which is online registration. We will design client-side privacy protection mechanisms for registration processes because it is not realistic to assume that website operators will not violate children's privacy in any way. We cannot hope for full parental supervision whenever their children are online since it places an extra burden on the parents who most likely do not have such time. This calls for an automated technical solution to facilitate parental control without bothering them or with little effort.

Our design of COP is compliant with Children's Online Privacy Protection Act (COPPA) [10], which governs the online collecting of personal information from children under the age of 13 and further distribution of such information in the United States. COPPA states that any website directed to children under 13 must post a link to their privacy policy at any place where it collects personal information from children. COPPA also requires the website to obtain verifiable parental consent for the collecting action and any further use and disclosure of children's personal information. The definition of children's personal information in COPPA includes individually identifiable information such as full name, physical address, email address (or other online contact information), telephone number, age, gender, social security number as well as other auxiliary information such as hobbies, preference and information collected through cookies.

1.2 Related Work

Most of the current technical solutions intend to protect users' online privacy in general. We will first introduce several such solutions and then focus on techniques for protecting children's privacy.

Cookies, a unique identifier that can be used for retrieving records from the databases, authenticating users and tracking users' activities, were seen as a major threat to users' online privacy [11]. COPPA recognizes cookies as privacy-invasive and disallows operators from collecting cookies that can be linked to a child. As a countermeasure, most Web browsers adopt cookie control to give users the option to disallow cookies from a website. These cookie blocking features are effective but they only address a very small portion of COPPA's requirements because these solutions can not prevent websites from explicitly collecting personal information from children under the age of 13.

Anonymizer [12] is a solution that protects user's privacy by providing a way for anonymous Web surfing. It redirects all Web traffic through intermediary proxy servers to hide the user's IP address. The Anonymizer serves as a good privacy solution to fight against phishing and pharming attacks. However, it is not sufficient for protecting children's online privacy because it cannot prevent websites from collecting personal information during online registration process. In addition, anonymous browsing may encourage children to access objectionable materials once they are aware that they are not being identified as children.

Another popular approach to privacy assurance is through self-regulatory efforts which involve the setting of standards either by the website itself or an industry group and the voluntary adherence to the set standards or policies [13]. Under a self-regulatory approach to regulating children's online privacy, groups like TRUSTe [14] have been active as the third party entities policing children's privacy and promoting trustworthiness to websites through seals of approval. By becoming a member of these private watchdog groups, a website is permitted to post the seal of approval. These seal programs provide a means to guarantee that members abide by a set of clearly identified self-regulatory standards [15]. However, research has shown that, most privacy policies posted online are written in jargon and ambiguous language and thus readability is low [16,17,18]. For those parents who are not technically inclined or are unaware of COPPA, they usually fail to make informed decisions for their children's information disclosure. In addition, it has been found that few users recognize privacy seals [17]. Thus, we conclude that the self-regulatory approach to children's privacy through privacy policies or privacy seals cannot be adopted as a stand-alone solution but as an additional protection layer complimentary to technical enforcement of COPPA.

A few tools were dedicated to protecting children's online privacy. Parental Online Consent for Kids Electronic Transactions (POCKET) [19] is one of such tools designed to give parents control over children's personal information disclosure. POCKET requires both Web clients and merchant websites to maintain a privacy preference file (PPF) stating their privacy policies. POCKET enables a trusted third party (TTP) server to perform mutual authentication between clients and merchant websites. Children's privacy is preserved by comparing the PPF from a merchant website with a user's own PPF and disclosing only the mutual parts. Setup of a PPF on a client side reflects the parental consent on children's personal information disclosure. First problem with POCKET is its avoidance of HTML forms, which generates inconsistency with users' normal activities. For example, users normally submit their registration information by filling out HTML forms, but when POCKET is enabled, users do not have any control over what information will be submitted on a site by site basis. Another concern is that merchant websites may not always follow their privacy policies and it should be the users' responsibility to protect their own privacy. Moreover, TTP might be the single point of failure although it only works in the system registration phase.

Several other software packages have also been proposed to empower parental control over children's online behavior. One of them is Windows Vista's parental control [20], which is designed to help parents to manage what their children can do on computers. Another is Privo [21], which will suspend children's online registration and ask for parents' opinions if the websites require privacy information. Other tools such as icouldbe [22], Net Nanny [23] and Parental Control Bar [24] are also developed to protect children's online safety by filtering Web contents and blocking functions. The problem with these tools is that they usually filter out outbound user inputs or change them to asterisks to prevent children from divulging any privacy. This restraint on information disclosure during registration may hinder children from gaining access to normal services. Maintaining the balance between protecting children and retaining their accesses to appropriate Web contents has become the concern of industry practitioners and government agencies [25]. It is also one of COP's design goals.

1.3 Contributions

The main contributions of this paper include:

- We present COP, a light-weight client side solution to protect children's online privacy.
- We show that COP fulfills COPPA's requirements by implementing verifiable parental consent before collecting of children's personal information.
- We demonstrate that COP achieves the balance between children's privacy protection and usability of collected data set by leveraging concepts from privacy preserving data mining (PPDM).
- We evaluate the effectiveness of COP in preserving children's private information during online registration.

The remainder of the paper is organized as follows. Section 2 describes COP's design and implementation. Section 3 evaluates the effectiveness of COP serving as a children's privacy protection solution. Section 4 lists potential attacks from malicious websites and our countermeasures. Section 5 discusses the limitation of COP as well as our future work on COP. Section 6 concludes.

2 Design and Implementation of COP

COP is designed to prevent private information leakage in online registrations and is implemented as a client-side Web browser extension. We rule out the usage of trusted third party in COP to avert single point failure and to give parents maximal control over COP.

To begin with, we define two types of users in the design of COP, namely, parent user and child user. These two users are given different access rights to COP and involved in different phases of COP's operation. In the installation phase, only parent users who have the administrator privilege should install COP in their computers and set up their parent passwords to protect COP from being modified or disabled. After installation is done, COP works as a browser extension and keeps itself transparent to child users. The only information child users can see is the warnings prompted by COP prohibiting registration to certain websites. Since only parent users have access to COP's settings after they verify themselves as parents to COP by inputting their passwords, it is the parents' responsibility to keep the password from being disclosed to others such as their children.

Figure 1 illustrates the system level design of COP. Once COP has been installed, configured and activated properly by a parent user, it starts to monitor the outgoing traffic. When children intend to sign up in a website, the browser sends an HTTP request to the Web server which responds with a Web page containing a registration form. Children will fill out the form with their information such as age, firstname, lastname and submit the form to the website (Here we assume that children always provide true information and we argue that if they provide false information either intentionally or by mistake, their privacy will not be jeopardized since no real personal information is disclosed). This submission will be intercepted by COP as shown in Figure 1. Then COP will adopt data perturbation on the registration information sent to this website. COP will also notify parents either immediately or at a pre-selected time depending on the preference of parents. Upon receiving the registration information, the website, which is assumed to conform to COPPA, would know the client is under 13, and it should send back its privacy policy. This policy will be intercepted by COP and showed to parents later. Once the registration is done, the name of the website as well as the perturbed information submitted to the website will be logged by COP.

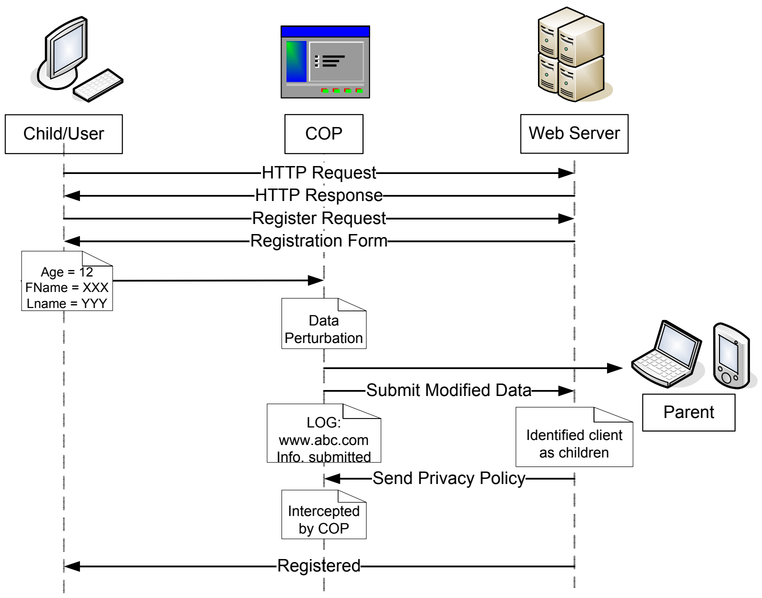


Fig. 1. System design overview

2.1 Privacy Preference

In this section we discuss the implementation of privacy preference in COP. Instead of applying simple rules such as allowing or prohibiting personal data being collected by a website, COP establishes one privacy preference entry for each website. Each preference entry indicates what categories of information can be collected and what cannot be collected by that website. These categories of personal information are defined by COPPA, e.g., 'Name', 'Age', 'Date of Birth', 'Gender', 'Phone Number', 'Address'. A preference entry is a reflection of parental consent on collecting of their children's information by the website in the entry. Some entries are pre-defined; others are automatically generated by COP when that website is visited for the first time. For example, when a user connects to "www.example.com", COP first searches the preference entry list for an entry of www.example.com. If a match is found, COP will apply that entry to the personal information required by www.example.com in registration. If no match is found, a pre-defined default privacy preference will be used for www.example.com. Fully customizable feature of the preference list provides parents with a fine-grained control over disclosure of their children's personal information.

2.2 Privacy Preserving Data Perturbation

One feature distinguishing COP from other similar schemes is that COP treats personal information as a set of privacy metrics instead of one single piece of information. Since COP is a client-side solution, personal information is stored and processed in a distributed fashion. Without the knowledge of the population of a data set, existing privacy measurements such as k-anonymity can not be applied here. To embody the privacy protection from adopting COP in this work as well as to build a foundation for quantitatively analyzing privacy (discussed as future work in Section 5), we leverage the concept of privacy preserving data perturbation to generate data that appear to be genuine instead of random results and substitute these data for user inputs which contain protected personal information.

One advantage of data perturbation over blocking user inputs is the avoidance of failure in registration. Web servers always perform extensive user inputs validation check by either server-side examination or inline script functions. For example, when a registration form expects users to input an email address, it normally will not accept strings like "John" or "test.com". When a credit card number is required, a VISA card indicated by user must at least have a starting digit of "4" and a 16 digits length in order to pass the validation check. In some cases, only a valid credit card number can survive the examination (e.g., Luhn algorithm), not even one false digit is allowed and these validations will keep bothering users until a genuine data is input. To this end, data perturbation provides a solution to pass these checks without providing personal information.

Another reason to use data perturbation is to retain the statistical properties of submitted data when individually collected data records are put together as a data set by the Web server. As long as COPPA is not violated, we should allow websites to analyze collected data for their own purposes and consider users' privacy preserved at the same time.

In practice, challenges arise from introducing data perturbation into COP. First we need to process many different types of data, such as numerical, string and enumeration. Clearly, there is no single algorithm suitable for perturbing all these kinds of data. Second, as mentioned before, there exist constraints on the perturbed data for passing validation checks to be accepted by Web servers. Third, conflicts might appear between perturbed data, for example, the first three digits of a phone number might give different geographical information than a ZIP code[1]. In cases where such conflicts impede user registrations, maintaining consistency between various perturbed data is a necessity. Last but not least, some parts of personal information such as email address cannot be automatically generated because a fake email account will not support further communication between a user and a website.

[1] This discrepancy might also happens within true data because some people use phone number from other regions.

To address these challenges, we borrow some approaches from PPDM, namely additive perturbation [26], multiplicative perturbation [27,28] and probability distortion [29]. In additive perturbation, noise is added to data in order to mask the attribute values of records. The noise added is sufficiently large so that the individual record values cannot be recovered from the perturbed data. Note that although there are known drawbacks of additive perturbation such as additive noise may be easily filtered out through correlation of the data points within a large data set, it will not be an issue for data processed in COP because a website only has one data entry from each child. There are two basic approaches to perform multiplicative perturbation. We only consider the first one. This method is based on generating random numbers that have a truncated Gaussian distribution with mean equal to one and a small variance. It multiplies each element of the original data by this noise. Unlike the previous two approaches, probability distortion perturbs the value of each data element (point distortion) and replaces it with another sample from the same (estimated) distribution. The merit of this approach is the difficulty to compromise perturbed data using repeated queries.

Table 1 shows four types of data that might be requested from a child during registration. For each data type, its related data items, potential perturbation methods, range or formats and special notes are listed. For non-format numerical type of data such as

Table 1. Data Perturbation Approaches for different Data Types

Data Type	Data Item	Possible Perturbation Methods	Format	Notes
Numerical Value (non-format)	Age	1: Follow certain predefined distribution; 2:ϵ, normal distribution with $\mu = 1$ and $\sigma = 0.5$. $\alpha = (Age \times \epsilon) mod 13$, if$\geq$ 6, $R(Age) = \alpha$, else $R(Age) = \alpha + 6$	$6 \sim 12$	1: Assume we know the distribution of ages from 6 to 12; 2:Multiplicative perturbation
Numerical Value	Phone Number	Reserve area code generate other 7 digits	123-XXX-XXXX	Certain Rules
	SSN Number	Randomly Perturb	XXX-XX-XXXX	Certain Rules
	Date of Birth	1: Year must be in accordance with age; 2:Month and day follow age's perturbation	XX-XX-(>1996)	
	ZIP Code	Reserve the first 3 digits, perturb the last two digits	021XX	Consist with Address
	Credit Card	Follow the CCN rules, randomly perturb or use predefined dataset	16 or 15 digits	-
String	Name	Choose from certain data set like cartoon names	Mickey Mouse	-
	Username	Do not perturb unless real name used	-	-
	Address	Keep state name change door number street name and city name	1234, test street, fake city, PA	City name can be preserved if defined by parents
	E-mail	Change to parents' email address	-	-
Enumeration	Gender	change with probability a%	-	-

age, there are two perturbation options. The first one applies probability distortion, and the retention of existing distribution of ages is achieved by following that distribution when generating perturbed age data. The other option exploits the idea of multiplicative perturbation to conceal individual user's age in a normal distribution. By default, COP adopts the second perturbation approach for the reason that no such age distribution is known worth following. For formatted numerical type of data, we first confirm the potential information each format gives away, and then determine the extent to which COP's protection will cover. Take the ZIP code and phone number as examples, the first three digits of these two numbers indicate users' geographical information and COP's policy allow the disclosure of these information for the balance between user anonymity and data usability. Perturbation policy of email address differs from others for the consideration of possible usage of email account to retrieve password in future. Unlike other data, validity of perturbed email addresses can not be assured by COP. For enumeration type of data, it can be processed as numerical data with a certain range.

Another issue in data perturbation is the validation of perturbed data. For example, when a ZIP code is perturbed from "02108" to "02107", the perturbed value "02107" is not a validate ZIP code thus it can not pass the validation check. To avert this issue, COP prepare a list of all valid ZIP code, and the perturbation process can choose from the list according to the rules in Table 1.

2.3 Parsing User Input

COP is designed to minimize its interference with users' normal Web activities. To this end, parsing user input in COP needs to distinguish online registration Web pages with other Web pages in the first place. Generally, a registration Web page always contains a form and a submit button, and the button is associated with an event handler (e.g., OnClick) to send out the form. Although many online shopping Web pages may also have the similar structures, we assume that a child user under the age of 13 is unlikely to shop online. Thus COP can discriminate registration Web pages from others by these characteristics.

If the current Web page is considered as a non-registration page, COP follows each user input and compares the content of the input with pre-stored personal information. Upon a match is encountered, COP will retrieve id/name attributes of the input field and the tag before the field. If this retrieved information indicates that the input field asks for personal information. COP will treat this input field as a potential leakage of privacy and perturb the user input. Otherwise, COP will just leave the input field unchanged.

In the case of registration Web pages, COP not only follows and compares each user input with pre-stored personal information, but also considers the data type of the input content. If the input is a string and matches one of the personal information stored as a string such as name, address and email, the input is considered as personal information and is perturbed. If the input is a numerical value with format and matches one the following personal information: ZIP, Credit Number, Phone Number, SSN and Date of Birth, it is also considered as personal information and is perturbed. However, if the input is a numerical value with no format and matches the pre-stored age information, for example, user inputs "12" and her age is also 12. In this case, COP will resort to tag and id/name to recognize the meaning of this input. If the tag and id/name indicate that this input field does ask for user's age, then COP will perturb this input to protect privacy. Otherwise, COP will not change this input because many elements in an HTML file are treated as input fields with a small numerical content. A simple example is a dropdown-list. The selection on the list will be stored as a numerical value representing the list index. If COP perturbs each input field that has the same numerical value as the user's age, a high false positive will be introduced and the user's normal Web activities will be interrupted.

However, there are cases where the input needs to be perturbed even the content of the input does not match any personal information. Considering a child lying about his age to access restricted content, the tag or id/name of the input field will indicate this field asks for user's age, but the content does not match stored age information. In this case, COP will perturb the input to a number less than 13 to prevent the child from lying.

2.4 Transparency to Children

Since COP is dedicated to protecting children under the age of 13, one of COP's design objectives is keeping their child-like innocence. We notice that children might confuse perturbation with deceiving because they are not mature enough to understand the privacy protection purpose of perturbation. To prevent this, COP keeps the data perturbation operations unobserved to child users. After parsing a user's input, COP generates the perturbed data and marks the input field without changing its content immediately. Only when the submission action is triggered, COP will change all the marked contents with perturbed ones and submit the form. Obviously, reentry would be a problem for users if their usernames have been changed by COP without notifying them. Since username is not treated as personal information by COPPA, under normal circumstances, COP will not interfere with username. However, there are cases where children use their real names as username for login, the real names would be randomly perturbed by COP and this would cause failure in logging into users' accounts. COP addresses this problem by implementing a logging facility. When children visit a website that has been recorded in the log, COP first looks up the related perturbation data for this website. Once found, COP will use these data when required instead of creating new perturbed data. This means if a username has been perturbed during registration, COP will provide it when users try to log in. With the assistance of logging function, COP can preserve transparency to child users.

2.5 Verifiable Parental Consent

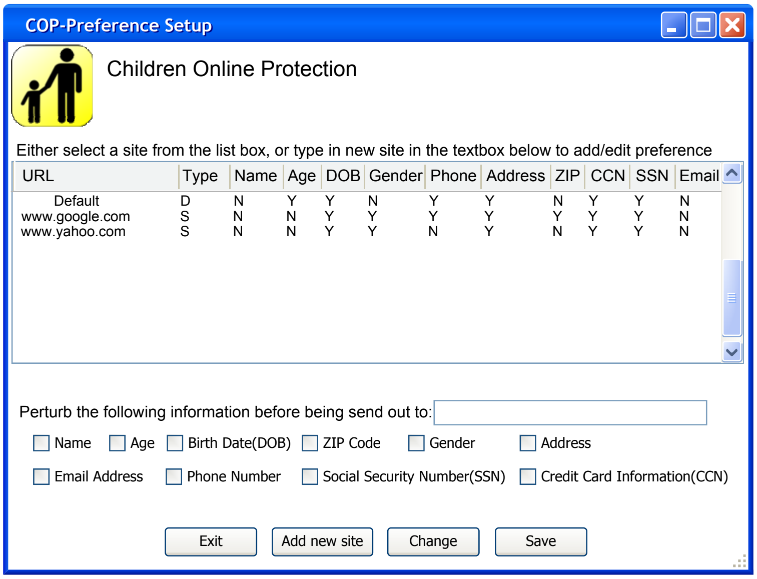
COPPA requires that "prior to collection, use, and/or disclosure of personal information about a child, an operator must obtain from a parent of the child verifiable parental consent..." [10]. Obtaining such content every time when information is required from a child may get cumbersome both for the child and the parents. Therefore, we need a way of delegating this responsibility to the tool for approving website policies. Note that since COP will be enforcing the privacy preferences set by the parents, irrespective of the website's policy, COP is always ready to approve the website's policy and to release information that has been pre-approved by parents. The only technical issue is to provide parental consent to the Web server in an automated way, which is not yet implemented in the current version of COP. Nevertheless, in reality, automated approval is not always necessary. Since websites with age censorship tend to acquire this consent directly from parents (see Section 3.1). Common practice of websites is sending email to parents and letting them click a link to give their approval.

2.6 Browser Extension Implementation

We implement COP as a Firefox extension (Firefox version 2.0.0.20). As an add-on to Firefox's functionality, COP is integrated into the browser's main frame once installed. As discussed at the beginning of Section 2, only parent users with administrator privilege should install COP. The installation involves three important steps, which are setting password, specifying privacy preference and filling in children's information. Password is used to identify parent users before they are allowed to modify COP's configuration.

Figure 2 shows the panel where parents can specify privacy preferences in COP. Parents can add any websites they know of into the list and set up the appropriate policy for each of them. Specifically, parents can indicate which categories of their children's information should be withheld from the website by selecting the check-boxes below the list. A checked one means COP will perturb this information before sending it out, which is marked as 'Y' in list (otherwise 'N'). For unknown websites, COP will apply the default policy.

Figure 3 illustrates the user-interface for parents to add their children's information. This information is used as the reference for parsing user input as discussed in



Either select a site from the list box, or type in new site in the textbox below to add/edit preference

URL	Type	Name	Age	DOB	Gender	Phone	Address	ZIP	CCN	SSN	Email
Default	D	N	Y	Y	N	Y	Y	N	Y	Y	N
www.google.com	S	N	N	Y	Y	Y	Y	Y	Y	Y	N
www.yahoo.com	S	N	N	Y	Y	N	Y	N	Y	Y	N

Perturb the following information before being send out to:

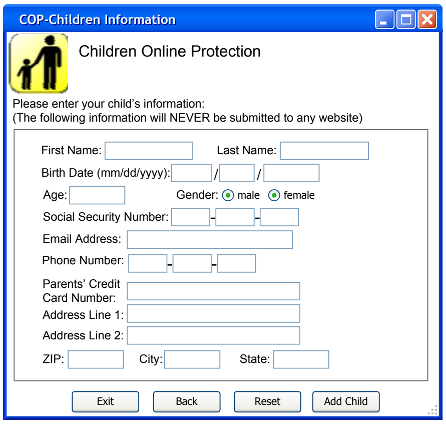
☐ Name ☐ Age ☐ Birth Date(DOB) ☐ ZIP Code ☐ Gender ☐ Address

☐ Email Address ☐ Phone Number ☐ Social Security Number(SSN) ☐ Credit Card Information(CCN)

Fig. 2. Installation phase: specifying privacy preference

Section 2.3. Parents need to fill out the form in this panel for each of their children. If they have more than one child, they need to use "Add Child" button to generate a new form for another child. All the information gathered in this step is considered to be private and COP will prevent children from releasing any of this information to websites, depending on the privacy preferences.

After installation, COP will be activated with an indication appearing in Firefox's status bar. Figure 4 shows the menu list of COP. "Activate/ Deactivate" option is password protected. This ensures COP can only be disabled by the parents but not the children. "Parent Identification" option is used by parents to change the identification information entered during the installation phase. "Preferences Setup" option enables parents to change preferences as showed in Figure 2. "Log" option gives parents access to logged activities, which include the websites visited by children and the perturbed information submitted to those websites.



COP-Children Information

Children Online Protection

Please enter your child's information:
(The following information will NEVER be submitted to any website)

First Name: ____ Last Name: ____
Birth Date (mm/dd/yyyy): __ / __ / ____
Age: ____ Gender: ◉ male ◉ female
Social Security Number: ___ - __ - ____
Email Address: ____
Phone Number: ___ - ___ - ____
Parents' Credit Card Number: ____
Address Line 1: ____
Address Line 2: ____
ZIP: ____ City: ____ State: ____

Fig. 3. Installation phase: filling in children's information

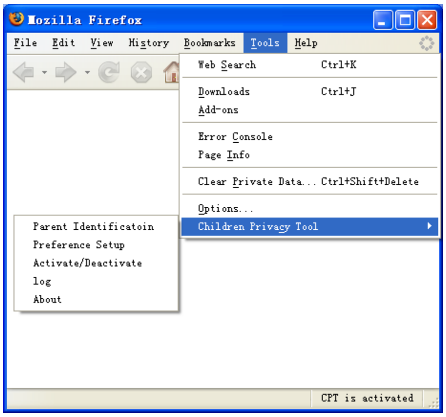


Fig. 4. COP works as a Firefox extension

3 Evaluation

In this section, COP is evaluated for its effectiveness of protecting children's online privacy.

We test COP with registrations on 23 websites, as listed in table 2. Twelve of these websites are selected from the top site list for kids and teens (suggested by Alexa [30]), e.g., Skyrock, GameSpot, Hyves, Nick. Other sites are the representatives of most popular Web services, e.g., Yahoo, Google, MySpace and Facebook.

We examine the results from the following aspects. First, we look into the amount of personal information required for registration. Numbers in the second column of Table 2 indicate which pieces of information are required by a website among a total of 11 categories. The average number of categories of information in this column is 4.48. The most wanted information is email, (82.6% of visited websites ask for email) followed by full name (65.2%), date of birth (60.8%) and gender (56.5%). Considering the threat of cross identification studied in [7,6], the required registration information from 5 websites is enough for a malicious party to identify a person, which highlights the importance of privacy protection scheme for online users especially children.

Table 2. Popular Websites Visted in COP's Evaluation

Website	Information Required for Registration [a]	Age Verification
www.yahoo.com	2,4,5,6,9	Yes
www.live.com	2,3,4,5,6,9	Yes
www.google.com	4,5	No
www.kidscom.com	4,5,6,10	No
www.gzkidzone.com	3,4,5,10,11	No
dashboard.aim.com	1,2,3,4,5,6	No
www.facebook.com	1,2,3,4,5	Yes
www.livejournal.com	1,3	Yes
www.youtube.com	1,2,3,6,9	Yes
www.myspace.com	1,2,3,4,5	Yes
www.blogger.com	3	No
www.hi5.com	1,3,4,5	Yes
www.wordpress.com	3,4,5	No
www.skyrock.com	1,2,3,4,5,6	Yes
www.gamespot.com	1,2,3,4,5,6,7,8,9	Yes
www.hyves.nl	1,2,3,4,5,6,9	No
www.gamefaqs.com	1,2,3,4,5,6,7,8,9	Yes
www.neopets.com	1,2,4,6,7,9	No
www.nick.com	1,2,3	No
www.everythinggirl.com	3	No
www.stardoll.com	1,2,3	No
www.lego.com	1,3	No
www.timeanddate.com	3,4,5	No

[a] 1:birth date; 2:gender; 3:email; 4:first name; 5:last name 6:country; 7:state; 8:address; 9:ZIP; 10: age; 11:parent email.

Second, we evaluate the effectiveness of COP protecting personal information. By using default privacy preference setting, all the 11 categories of personal information are protected by COP. Data perturbation shields real information from disclosure. However, in some rare cases such as KidsCom [31], where users provide personal information by clicking pictures figuring preselected answers, COP could not intercept this information. We consider this case as an example of covert information access and will discuss the details in Section 4.1.

We take the registration process on www.yahoo.com as an example to demonstrate how COP works. After COP has been installed in Firefox, "COP is activated" shows up in the status bar. When the browser opens the registration page of Yahoo, personal information such as first name, last name, gender, birthday, country, ZIP, alternative email are required in a form. After we finish filling out all the fields in the registration form and click "Create My Account", COP intercepts this action, performs data perturbation based on the local privacy preference setting, places all the perturbed data into the corresponding fields and sends out the form. Since Yahoo has age censorship, upon receiving this registration request from a user under 18, Yahoo requires an adult's Yahoo ID to proceed. We provide it to finish the registration. All the information Yahoo gets from this registration is a subset of what we, as parent users, allow to release. The most detailed information in this case is an area code in ZIP. This very limited private information disclosure can effectively protect our identities online. From the perspective of child users, this registration process is no different than the one without COP, only their privacy is preserved.

During our test, we notice that age censorship is quite common among current websites. However, these censorships can be bypassed by children lying about their ages to be over 18. Since in configuration COP already has the real age of the child, it can change the filled age to a number less than 13 to make sure those censorships will be invoked.

4 Attacks from Malicious Websites

Some websites may not favor COP for its data perturbation design. If COP is widely deployed, those "untrustworthy" websites might try to exploit any design vulnerabilities in COP. In this section, we discuss some of the attacks that malicious websites would take against COP and our countermeasures.

4.1 Covert Information Access

Although COP spends a lot of efforts on preserving the usability of perturbed data for collectors, it is not surprising to see that some websites which are aware of the usage of COP would try to bypass COP. For example, in the KidsCom case, the website acquires personal information from children by letting them choose from a group of preselected pictures. In this way, no recognizable user input happens, and COP will not be able to intercept any private information. Solution to this problem requires image recognition techniques, which is beyond the scope of the current design of COP. On the other hand, we believe that this information collection fashion is not a general practice among websites, because it is inconvenient for both the users and the websites.

4.2 Embedded Code

One potential attack comes from special embedded JavaScript code. In normal cases, a button control is embedded in a form; once clicked, the Web browser composes an http request by putting together all user input data and sending it in the request. In this case, COP is able to intercept this standard request. However, with JavaScript, a webpage may link a button to an embedded JavaScript code, for example, <input key="send" onclick="SendData()" value="sendsecretly" type="button">, where SendData() is an inline JavaScript function. This function can easily read all the data the user has provided so far, encode it in a specified secret way understandable to the Web server alone, and finally send the data to the server. Thus, the browser (and COP) will not see the original data fields as the values look random. This in turn could make perturbation of the data an impossible task. To address this attack, we consider automatically adding a hook JavaScript method before the JavaScript data submission method. For example, we may implement COP inside the Web browser layout engine (e.g., Gecko for Firefox). When it detects the above HTML source code, it can modify the code to <input... onclick= “ICheckFirstHook(); SendData()”> instead. Here ICheckFirstHook() is a JavaScript function, added by COP to the HTML source to check or perturb the user input data before it is passed to the original JavaScript function defined by the Web server.

5 Discussion

COP’s mission is to reduce divulgence of children’s personal information to websites. Despite COP’s effective protection, it is not realistic to solely rely on COP, a scheme focus on registration process, to eliminate the possibility of any invasion of privacy. A simple example would be a child posting his or her name, hobbies and maybe photos on an online blog after registration. In another case, a child may release sensitive information such as addresses to others during online chatting. These activities obviously violate the privacy preserving requirements, but their diversity makes it challenging for a single scheme to prevent all of them. Moreover, children might install other browsers to bypass COP or they might be able to disable or uninstall the Firefox extension. To prevent COP from being circumvented, a comprehensive implementation such as a proxy is envisioned. The proxy will intercept all Web traffic. It is installed by system administrator and can only be uninstalled or disabled by users with the same privilege as the administrator. As long as children are not given such privilege, the proxy scheme can not be circumvented.

COP suggests default privacy preference avoiding exposure of “ZIP”, “Gender” and “Date of Birth” information to the same website to defend against cross-identification attacks. However, other forms of privacy intrusion might happen due to availability of other auxiliary information. Solutions to this intrusion will include understanding of the mechanisms of such attacks and a more conservative default privacy preference that prevents disclosure of almost all the personal information.

One of our future works, as discussed in Section 2.2, is to propose an approach for quantitatively analyzing data privacy under the distributed model, in which sensitive data are collected from individuals where privacy control (E.g., COP) are placed. The challenge in this problem is the definition and measurement of privacy from an individual’s perspective without the knowledge of data possessed by others. The approach adopted in this work, which considers each piece of personal information as a separate privacy metric might suggest a possible direction for solving this problem. In our future effort, we also plan to conduct a comprehensive survey with a sufficient large sample size. The sample should consist of parents as well as their children to deliver more representative feedbacks on the effectiveness of COP.

6 Conclusion

COP offers a novel solution to address the ever-growing concern on children's online privacy divulgence. By concentrating on online registration process, the most common way of user private information collection, COP preserves children's privacy from releasing to websites while trying to preserving their normal activities. COP can also fulfill the requirements of COPPA and maximize parents control over their children's online activities. Moreover, parents' real-time involvement is little if any except the easy configuration in the installation phase. COP manages to achieve these goals by implementing functions like privacy preserving data perturbation, user input parsing and transparency to children. We demonstrate COP's effectiveness in serving its privacy protection purpose by evaluating its performance in popular websites. We also believe COP is one of the directions for protecting children from online threats. With further improvements, COP can help parents to protect their children in a more effective way.

Acknowledgement. We thank the reviewers for their valuable comments and suggestions. This work was partially supported by NSF CAREER 0643906.

References

1. Katz, L.: When 'digital bullying' goes too far (June 2005), http://news.cnet.com/when-digital-bullying-goes-too-far/2100-1025_3-5756297.html (Retrived January 2010)
2. Online predators: Help minimize the risk (January 2007) (retrived January 2010)
3. BBC: Net blamed for rise in child porn (2004), http://news.bbc.co.uk/1/hi/technology/3387377.stm (retrived January 2010)
4. FTC: Xanga.com to pay 1 million for violating children's online privacy protection rule (2006), http://www.ftc.gov/opa/2006/09/xanga.shtml (retrived January 2010)
5. FTC: mbee.com settles ftc charges social networking site for kids violated the children's online privacy protection act; settlement includes 130,000 civil penalty (2008), http://www.ftc.gov/opa/2008/01/imbee.shtml (retrived January 2010)
6. Sweeney, L.: Uniqueness of simple demographics in the u.s. population. Technical report, LIDAPWP4, Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh (2000)
7. Golle, P.: Revisiting the uniqueness of simple demographics in the us population. In: 2006 Workshop on Privacy in the Electronic Society, pp. 77–80. ACM Press, New York (2006)
8. Cox communications teen internet safety survey wave ii. Technical report, Teen Research Unlimited (March 2007)
9. Youn, S.: Teenagers' perceptions of online privacy and coping behaviors: A risk-benefit appraisal approach. Journal of Broadcasting & Electronic Media 49, 86–110 (2005)
10. Children's online privacy protection act (1998) (Retrived January 2010)
11. Center, E.P.I.: Pretty poor privacy: An assessment of p3p and internet privacy (June 2000), http://epic.org/reports/prettypoorprivacy.html (retrived January 2010)
12. How anonymizers work (2007), http://www.livinginternet.com/i/is_anon_work.htm (retrived January 2010)

13. Zwick, D., Dholakia, N.: Models of privacy in the digital age: Implications for marketing and e-commerce. Technical report, Research Institute for Telecommunications and Information Marketing (RITIM), University of Rhode Island (1999)
14. Truste, http://www.truste.org/ (retrived January 2010)
15. Culnan, M.J., Bies, R.J.: Consumer privacy: Balancing economic and justice considerations. Journal of Social Issues 59(2), 104–115 (2003)
16. Milne, G.R., Culnan, M.J.: Strategies for reducing online privacy risks: Why consumers read(or don't read) online privacy notices. Journal of Interactive Marketing 18(3), 15–29 (2004)
17. Hsiao, M., Belanger, F., Hiller, J., Aggarwal, P., Channakeshava, K., Bian, K., Park, J.M.: Parents and the internet: Privacy awareness, practices and control. In: Proceedings of Americas' Conference on Information Systems (2007)
18. Culnan, M.J., Milne, G.R.: The culnan-milne survey on consumers & online privacy notices: Summary of responses. In: Proceedings of Get Noticed: Effective Financial Privacy Notices, Washington, DC, A Federal Trade Commission Workshop (2001)
19. Crossler, B., Belanger, F., Hiller, J., Aggarwal, P., Channakeshava, K., Bian, K., Park, J.M., Hsiao, M.: The development of a tool to protect children's privacy online. In: Annual Workshop on Information Security and Assurance, Montral, Canada (2007)
20. Parental controls in windows vista, http://www.microsoft.com/windows/windows-vista/features/parental-controls.aspx (retrived January 2010)
21. Privo, http://www.privo.com/ (retrived January 2010)
22. icouldbe, http://www.icouldbe.org/ (retrived January 2010)
23. netnanny, http://www.netnanny.com/alt_rotate (retrived January 2010)
24. Parental control bar, http://www.parentalcontrolbar.org/ (retrived January 2010)
25. Thierer, A.: Social networking and age verification: Many hard questions; no easy solutions. Progress & Freedom Foundation Progress on Point Paper No. 14.5 14(5) (2007)
26. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. In: ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining, Washington, DC, November 2003, pp. 99–106. IEEE Computer Society, Los Alamitos (2003)
27. Kim, J.J., Kim, J.J., Winkler, W.E., Winkler, W.E.: Multiplicative noise for masking continuous data. Technical report, Statistical Research Division, US Bureau of the Census, Washington, D.C. (2003)
28. Krishnamurty Muralidhar, D.B., Kirs, P.J.: Accessibility, security and accuracy in statistical database: The case for the multiplicative fixed data perturbation approach. JSTOR-Management Science 41(9), 1549–1564 (1995)
29. Liew, C.K., Choi, U.J., Liew, C.J.: A data distortion by probability distribution. ACM Trans. Database Syst. 10(3), 395–411 (1985)
30. Alexa-top sites by category, http://www.alexa.com/topsites/category/top/kids_and_teens (retrived January 2010)
31. Safe kids chat rooms, http://www.my.kidscom.com/ (retrived January 2010)

A Hybrid Method to Detect Deflation Fraud in Cost-Per-Action Online Advertising

Xuhua Ding

Singapore Management University
xhding@smu.edu.sg

Abstract. Web advertisers prefer the cost-per-action (CPA) advertisement model whereby an advertiser pays a web publisher according to the actual amount of transactions, rather than the volume of advertisement clicks. The main obstacle for a wide deployment of this model is the deflation fraud. Namely, a dishonest advertiser under-reports the transaction count in order to discharge less. In this paper, we present a mechanism to detect such a fraud using a hybrid of cryptography and probability tools. With the assistance from a small number of users, the publisher can detect deflation fraud with a success probability growing exponentially with the fraud amount, and can estimate the amount of frauds. Our scheme is amiable to both the advertiser and the users because the existing transaction model remains unchanged. It is also efficient and scalable as the incurred communication, computation and storage costs are independent of the number of transactions.

1 Introduction

Cost-per-action (CPA) is gathering its popularity among online advertisers due to its cost-effectiveness. Different from the cost-per-click model where an advertiser pays the web publisher for every user click, the payment in the CPA model is based on the amount of predefined user action, e.g. downloading, sale or sign-up. Nonetheless, such a model is not favored by the web publishers, because a dishonest advertiser may undercount the actions and consequently pays less commission fee. This type of cheating is called *deflation fraud*. Such frauds can also be found in other applications. For example, in publish-subscribe networks [7,14], an event publisher shares profits with the brokers, and a dishonest broker can undercount the number of subscriptions. In online content distribution businesses, a content distributor may cheat a content provider in a similar fashion.

A dual problem of deflation fraud is *inflation fraud*, whereby an entity cheats by maliciously over-counting or over-reporting transactions for its financial advantages. Although there exist many schemes proposed to address the inflation fraud as in [8,17,18,15,9,11,20], the deflation fraud has not caught sufficient attention except in [10]. As explained in [10], the philosophy for all inflation fraud detection schemes is knowledge proof. Intuitively, an inflation adversary is challenged to prove sufficient knowledge pertaining to the inflated count. Unfortunately, the same tactic fails for deflation fraud detection, as no scheme can challenge an adversary to present a proof on *absence of knowledge*. Therefore, the general approach for deflation detection is that the verifier (e.g. the web publisher in our context) gathers as much information as possible regarding the claim made by the prover (e.g. the advertiser). The solutions used in [10] are based on an online trusted third party (TTP) which mediates the transactions. A similar approach taken by Google's AdWords is that the verifier watches over the transactions directly. Obviously, the more information the verifier gathers, the stronger the detection scheme is.

Obviously, the aforementioned approaches are inefficient and unscalable as they are intrusive to the advertiser's business operation and involve an online TTP. We observe that missing a few transactions is a tolerable counting error to Ps as long as the fraction of fraud is sufficiently small. Therefore, we design an efficient and flexible scheme by slightly relaxing the security. The main results of this paper is a cryptography and probability based deflation-fraud detection scheme with the following attractive features.

– It detects any z amount of deflation frauds with a success probability at least growing exponentially with z. The web publisher can tune a security parameter to strike a balance between a high security assurance and a low cost.
– It allows the web publisher to estimate the expected transaction amount, which provides a sound basis to detect any frauds in a large magnitude.
– It is not intrusive to the advertiser in the sense that the transactions only involve the advertiser and its users only. It is also user-friendly as end-users do not need to maintain any secret information.
– It is efficient and scalable. The communication, computation and storage costs incurred by our scheme are independent of the amount of transactions.

The rest of the paper is organized as follows. We discuss related work in Section 2 and the building blocks in Section 3. Then, we formulate the problem in Section 4. Section 5 proposes the deflation fraud detection scheme. We analyze its security and performance in Section 6, then conclude the paper in Section 7.

2 Related Work

The most relevant work is by Johnson and Staddon [10]. They considered deflation fraud in content distribution. Three schemes are proposed in [10]. In the first scheme, the verifier impersonates regular users using different pseudonyms. Then, it checks whether its pseudonyms appears in the content distributor's report. The second scheme involves a TTP to pre-issue a set of keys to users in certain distribution. The content distributor must use a key known to all users, which helps the auditor to estimate the user set size. The third scheme is similar to the second one with the difference being a reduced user storage. The constructions of [10] has obvious architecture drawbacks and are not applicable to online advertising.

The problem investigated in this paper is akin to the count integrity in publish-subscribe networks [7,14]. A publish-subscribe system [7] involves a publisher which is the data source, a set of subscribers who receive their preferred data items, and a broker network consisting of a set of broker nodes which disseminate data from the publisher to the subscribers. The brokers report to the publisher about the amount of data delivered to the subscribers. The approach to count integrity proposed in [14] requires the publisher to participate in *all* data delivery transactions. This approach obviously abandons the design principle of publish-subscribe systems whose primary goal is to decouple publishers and subscribers in order to be more scalable and to save the publisher from the heavy workload of data delivery.

Web metering mainly deals with *overcount fraud* (a.k.a. *inflation attacks*). Naor and Pinkas [17] proposed a secret sharing based scheme to verify the number of users served by a web server. Their scheme is not suitable for generic online transactions since it requires the audit agency to initialize every user before running transactions. A special form of inflation attack is the well-known *click fraud*, whereby the adversary cheats on the amount of website visits instead of transactions. Gandhi et. al proposed in [9] countermeasures based on construction of advertisement code. Similar works also include [11,20,18,15,9].

In [12], Markus Kuhn proposed a novel approach to probabilistically counting a large collection of digital signatures, which may be used in applications like web page metering or ranking mechanisms. Though very efficient, this scheme suffers from inaccuracy, since it only provides an estimation on magnitude. However, many business applications demand a more precise count. Moreover, it does not address deflation fraud.

Another line of research related to our study is electronic voting schemes, such as [6,5,16]. Among many security requirements such as receipt-freeness, a fundamental requirement is that the ballots should be tallied correctly. Although secure e-voting schemes can theoretically defeat inflation/deflation attacks, they are not suitable for online advertising or content distribution, mainly because e-voting has a special and expensive infrastructure and has a heavy toll on the computation/communication costs.

3 Building Block and Notations

The cryptographic building block used in our scheme is the signature of knowledge [4,13], a non-interactive form of the zero-knowledge proof. The most primitive signature of knowledge is Schnorr signatures [19], whereby the signer proves that she knows the discrete logarithm of y to the base g in a cyclic group $G = \langle g \rangle$. An extension of Schnorr signatures can be used to prove the equality of two discrete logarithms. Suppose that g and g' are two generators of group G, and $y = g^x, y' = g'^x$. Knowing x, the signer produces a signature of knowledge proving that $DLOG(y, g) = DLOG(y', g')$. In essence, the signer generates a tuple which can be treated as two Schnorr signatures sharing the same challenge. We denote it by $\text{SKELOG}[x : y = g^x \wedge y' = g'^x]$.

In [4], Camenisch and Stadler defined the signature of knowledge of a *double discrete logarithm* of y to the base g and a. Let $y = g^{a^x}$. Knowing x, the signer computes $(c, s_1, \cdots, s_l)$ as the signature of knowledge denoted as SKLOGLOG$(x : y = g^{a^x})$, where l is a security related parameter. Their scheme is essentially a non-interactive version of l rounds of zero-knowledge proof with l binary challenges. Since the challenge used in each round is one bit, the signer can successfully cheat the verifier in one round with a probability 1/2. Therefore, an adversary can forge SKLOGLOG$[x : y = g^{a^x}]$ with a probability $1/2^l$. When l is sufficiently large, the probability is negligible.

In this paper, we combine SKLOGLOG$[x : y = g^{a^x}]$ and SKELOG$[x : y = g^x \wedge y' = h^x]$, so that with the knowledge of x, the signer can produce a signature of knowledge proving that the double discrete logarithm of $y = g^{a^x}$ equals to the discrete logarithm of $y' = h^x$. We denote it as SKELOGLOG$[x : y = g^{a^x} \wedge y' = h^x]$. Note that the computation of a^x in computing y and the computation of h^x are in the same group. The details are shown in Section 5.

4 Problem Formulation

4.1 System Overview

A typical online advertisement system consists of three types of entities: a web publisher Ps offering the advertising service through its web or search engines; a (unknown-sized) set of users $\mathcal{U}$; and an advertiser Ad who offers services or products to users. Ps displays to users the advertisement for Ad. By clicking the advertisement, the user is redirected to Ad's web site. The user is said to perform a *transaction* if he signs up the service or downloads the product. The user may or may not perform a transaction, solely depending on his own willingness. Periodically, Ad reports to Ps the number of transactions contributed by Ps's advertisement. In deflation fraud, Ad under-reports to Ps in order to pay less commission fee.

Our goal is to allow Ps to detect Ad's cheating in an efficient fashion. In a nutshell, the basic approach is to collect information from Ad and a tiny subset of $\mathcal{U}$. By analyzing the received data, Ps can discover (or suspect) the fraud in a probability growing with the fraud amount. The proposed scheme consists of the protocols/algorithms listed below.

- Initialization: Both Ps and Ad are initialized with the proper states. Ps authorizes Ad to run n transactions in maximum within one billing cycle.
- Advertising: In this protocol, a user $U \in \mathcal{U}$ interacts with Ps, where an advertisement of Ad is shown to U.
- Transaction: U may interact with Ad for an transaction, e.g. sign-up. It receives a receipt signed by Ad.
- Feedback: With a probability ρ, U runs this protocol with Ps to return his receipt, where ρ is a system-wide parameter selected by Ps. (The choice of ρ is discussed in Section 6.)
- Report: Ad reports to Ps about the amount of transactions performed. Ps detects fraud based on both the receipts collected from users and the report from Ad, and estimates the fraud amount (if any).

4.2 Assumptions, Adversary Model and Security Notions

We assume that every user $U \in \mathcal{U}$ is independent, and their protocol executions are regarded as independent events. We assume that all communication channels are confidential and authentic, e.g. via SSL/TLS connections. Henceforth, we do not consider attacks on the communication channels.

The adversary in our scheme is Ad. If there are in fact N transactions, Ad's objective is to report to Ps a fraudulent transaction count $N' < N$ without being detected. We assume that Ad is rational in the sense that it would not risk the exposure of its long term secret and it sets a risk threshold for itself. We do not consider collusion attacks between Ad and corrupted users. This is because Ad can always run the transaction with its colluders without executing the prescribed protocol[1].

The security strength of our scheme is defined based on the upper bound of the detection miss probability with regard to the amount of frauds.

Definition 1 (Deflation Resistance). *A deflation fraud detection scheme involving a publisher* Ps *and an advertiser* Ad *is said to be* (z, ν)*-secure for* $\nu \in [0, 1]$ *and* $z \in \mathbb{N}$*, if and only if the probability that* Ad *successfully undercounts* $x \geq z$ *transactions without being detected is bounded by* ν*, i.e.* $\mathsf{Pr}[\textit{detection fail } | x \geq \delta] \leq \nu$*.*

A perfect detection scheme should be $(1, 0)$-secure. Namely, Ps successfully detects any amount of deflation fraud. This can only be achieved by supplying Ps with the complete information about Ad's transactions. The aforementioned naive approaches, for instance, by introducing an online TTP, fall in this category. Our goal is to construct an efficient and scalable scheme by relaxing the security strength slightly, as long as z and ν are small enough to meet the application needs.

5 The Scheme

A high level view of the proposed scheme is as follows. Ps delivers a sequence of hash tokens to Ad. When Ad performs a transaction, a receipt derived using a fresh token is returned to the user. A small set of users report their receipts to Ps. The latter verifies whether Ad honestly runs the prescribed protocol and detects any anomaly using the received data. The details are presented below. In the sequel, we use the notations listed in Table 1.

5.1 Initialization

Given a system wide security parameter κ, all participants agree on the following cryptographic setting. Let p, q, q' be three large primes satisfying $q | p - 1$ and

[1] Note that cryptography techniques alone can not detect collusion attacks. Promising solutions could be based on trusted hardware or TTP. Nonetheless, we remark that it is infeasible for Ad to collude with a large portion of Internet users.

Table 1. Table of Notations

Notations	Description
(x, y)	Ad's private and public key pair for signing receipts to users;
(u, v)	Ad's another private and public key pair for token usage proof;
t_i	the i-th tokens issued to Ad;
C	Ad's transaction counter;
I_{click}	Ps's click counter;
C_f	the maximum transaction count received by Ps.
I_f	the click count corresponding to C_f.
N	the actual amount of transactions;
M	the total number of advertisement clicks;
ρ	the probability for a user to run Feedback after Transaction.

$q = 2q' + 1$, and the discrete logarithm problem is intractable in both $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$. Let $g \in_R \mathbb{Z}_p^*$ such that $\mathbb{G} = \langle g \rangle$ is a cyclic subgroup of Z_p^* of order q. Let $h \in_R \mathbb{Z}_q^*$ such that $\mathbb{G}_h = \langle h \rangle$ is a cyclic subgroup of Z_q^* of order q'. Hereafter, we omit the modulus p and q for group operations in $\mathbb{G}$ and $\mathbb{G}_h$ respectively, if they are indicated from the bases in use. Let $H : \mathbb{Z}_q \to \mathbb{Z}_q$ be a collision resistant hash function; and let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ be a collision resistant hash function, where k is a parameter determined by κ, e.g. $k = 160$. Let l be a parameter for SKELOGLOG determined by κ as well, e.g. $l = 80$.

Ps authorizes Ad to generate a chain of n hash tokens for n transactions. The benefits of using a hash chains are: 1) to save the communication and storage cost (i.e. a seed can generate the entire chain.); 2) to model the hash function as a random oracle for the provable security, because a hash token will be used in Transaction to derive a random number for signing a receipt. Note that the communication cost for the protocol is constant. The protocol of initialization is shown in Figure 1.

5.2 Advertising

Ps displays Ad's advertisements, e.g. a flash or a banner, on its web pages. It also maintains a counter I_{click} to keep track of the number of advertisement clicks. Initially, $I_{click} = 0$. Advertising begins when U clicks the advertisement. Then, Ps sends the (I_{click}, t_0) to U. Then, Ps sets $I_{click} \leftarrow I_{click} + 1$. To avoid confusion, we use M to denote the final I_{click}, i.e the total number of advertisement clicks.

5.3 Transaction

Ad maintains a counter denoted by C to count the number of transactions. Initially, $C = 0$. After running Advertising, U may decide to run a transaction with Ad. Similarly, a user can also decide not to run Transaction.

In the protocol, U first sends its I_{click} to Ad requesting for a transaction. Ad then signs I_{click} using its receipt signature key x together with a random number derived from the token t_C. As a result, Ad responds to U with two parts: a receipt

Initialization Protocol (by Ps and Ad)

1. Ps executes the following:
   (a) Generate a random seed $T \in_R \mathbb{Z}_q$; choose $n \in \mathbb{N}$ as the maximum amount of transactions Ad can perform;
   (b) Set $t_0 \leftarrow H^n(T)$, where $H^n(T) \overset{\triangle}{=} \underbrace{H(\dots H}_{\text{n times}}(T)\cdots)$;
   (c) Send $\{T, t_0, n\}$ to Ad;
2. Ad executes the following:
   (a) Select $x \in_R \mathbb{Z}_q$ and set $y \leftarrow g^x \bmod p$.
   (b) Select $u \in_R \mathbb{Z}_{q'}$ and set $v \leftarrow h^u \bmod q$
   (c) Output $PK := (y, v, p, q)$ as Ad's public key and $SK := (x, u)$ as its private keys. Ad's receipt signature key pair are (y, x).
   (d) On receiving $\{T, t_0, n\}$, check whether $t_0 = H^n(T)$. Abort if not equal. Otherwise, accept them.

Fig. 1. The Initialization protocol

which is a Schnorr signature on I_{click}, and a SKELOGLOG proof proving that the randomness in the Schnorr signature is derived from a hash token in the authorized hash chain. U completes the transaction if both the signature and the proof are verified true. The protocol details are described below in Figure 2.

5.4 Feedback

After running Transaction, U may choose to run Feedback to send a feedback to Ps. Let ρ denote the expected probability for a user to run Feedback after Transaction[2]. In the protocol, U simply returns its I_{click} and its receipt (C, t_C, γ, β) to Ps. Let Λ denote all receipts received. Let C_f denote the largest C in Λ, and I_f denote the I_{click} paired with C_f. When a new receipt is inserted into Λ, Ps runs the function Fraud(Λ) which returns 1 if there exist two distinct receipts with the same hash token. The details are described in Figure 3. Initially, $\Lambda = \emptyset$ and $C_f = 0$.

Note that the first step in Fraud(Λ) does not require additional modular exponentiations since $g^\beta y^\gamma$ has been computed when verifying the Schnorr signature (γ, β). To save the time cost for finding cheating, Ps can make use of a Bloom Filter to test the membership of $g^\beta y^\gamma$.

5.5 Report

Report is run by Ad and Ps at the end of each billing cycle. In the protocol, Ad reports to Ps with $\hat{C}$ as the number of transactions. In a deflation fraud, $\hat{C}$

[2] Ps sets ρ by using financial tools, e.g. receipt redemption or lucky draw, to attract users to run the protocol.

Transaction Protocol (by Ad and U)

Ad's input : $\{T, x, u\}$, the present transaction counter C; U's input: $\{I_{click}, t_0\}$.

1. Ad executes the following steps, when receiving I_{click} from U.
   (a) Compute $t_C \leftarrow H^{n-C}(T)$, and $\alpha \leftarrow \mathcal{H}(t_C \| \mathsf{Ad})$.
   (b) Compute $r \leftarrow \alpha^u \bmod q$.
   (c) Compute a Schnorr signature (γ, β) as $\gamma = \mathcal{H}(t_C \| y \| g \| I_{click} \| g^r)$ and $\beta = r - x \cdot \gamma \bmod q$.
   (d) Randomly selecte $r_1, \cdots, r_l \in_R \mathbb{Z}_{q'}$
   (e) Compute $w_i \leftarrow h^{r_i}$ and $w'_i \leftarrow g^{(\alpha^{r_i})}$, for $i = 1, \cdots, l$.
   (f) Set $\psi \leftarrow \mathcal{H}(v \| h \| g^r \| g \| t_C \| w_1 \| \cdots \| w_l \| w'_1 \| \cdots \| w'_l)$.
   (g) For $i = 1, \cdots, l$, set

$$\lambda_i = \begin{cases} r_i & \text{if } \psi[i] = 0, \\ r_i - u \bmod q' & \text{otherwise.} \end{cases}$$

   (h) Send $\{C, t_C, (\gamma, \beta), (\psi, \lambda_1, \cdots, \lambda_l)\}$ to U; then $C \leftarrow C + 1$.
2. $\mathcal{U}$ performs the following on receiving $\{C, t_C, (\gamma, \beta), (\psi, \lambda_1, \cdots, \lambda_l)\}$,
   (a) Verify whether t_C is in the hash chain rooted at t_0. Namely, if $t_0 \neq H^C(t_C)$, return 0 and abort;
   (b) Compute $\alpha \leftarrow \mathcal{H}(t_C \| \mathsf{Ad})$.
   (c) Evaluate the transaction verification function denoted by $V(t_C, I_{click}, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l)$ as follows.
      i. Compute $W \leftarrow g^\beta y^\gamma$.
      ii. If $\gamma \neq \mathcal{H}(t_C \| y \| g \| I_{click} \| W)$, return 0 and abort;
      iii. For $i = 1, \cdots, l$, compute

$$w_i = \begin{cases} h^{\lambda_i} & \text{if } \psi[i] = 0, \\ v^{\lambda_i} & \text{otherwise.} \end{cases} \quad \text{and } w'_i = \begin{cases} g^{(\alpha^{\lambda_i})} & \text{if } \psi[i] = 0, \\ W^{(\alpha^{\lambda_i})} & \text{otherwise.} \end{cases}$$

      iv. If $\psi \neq \mathcal{H}(v \| h \| W \| g \| t_C \| w_1 \| \cdots \| w_l \| w'_1 \| \cdots \| w'_l)$, return 0 and abort.
   (d) Accept (C, t_C, γ, β) as a receipt for this transaction and return 1.

Fig. 2. The Transaction Protocol

is less than the actual number of transactions (denoted by N). Since $\mathsf{Feedback}$ is independent of the Report, the execution of Report implies that Ps does not discover cheating from $\mathsf{Feedback}$. Therefore, Ps assesses the credibility of $\hat{C}$ using the data it receives.

The data includes M, C_f, I_f, where M is the total number of advertisement clicks and is the result from $\mathsf{Advertising}$; C_f and I_f are from $\mathsf{Feedback}$. With these data and ρ, Ps runs the following steps.

1. If $C_f > \hat{C}$, then Ps claims that Ad cheats and the amount of deflation fraud is at least $C_f - \hat{C}$.

Feedback Protocol (by Ps and U)

Ps's input: Λ, C_f; U's input: $\{C, t_C, I_{click}, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l\}$.

1. U sends to Ps: $\{I_{click}, C, t_C, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l\}$.
2. Ps executes the following steps.
   (a) Verify the receipt using the same algorithm in Figure 2. Abort if $V(t_C, I_{click}, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l) = 0$.
   (b) Set $\Lambda \leftarrow \Lambda \cup \{(C, I_{click}, \gamma, \beta)\}$.
   (c) Execute Fraud(Λ) as follows:
      i. If $\exists(C, I'_{click}, \gamma', \beta') \in \Lambda$, s.t 1) $I'_{click} \neq I_{click}$ and 2) $g^{\beta}y^{\gamma} = g^{\beta'}y^{\gamma'}$, compute $x \leftarrow (\beta - \beta')/(\gamma' - \gamma)$. Claim Ad's deflation fraud and return 1.
      ii. If $C > C_f$, set $C_f \leftarrow C$, and $I_f \leftarrow I_{click}$. Return 0.

Fig. 3. The Feedback Protocol

2. Ps may runs an anomaly detection using $\hat{C}, \rho, C_f, I_f, M$. In short, Ps first computes $\mathbf{E}[N]$. An alarm will be raised if the difference between $\mathbf{E}[N]$ and $\hat{C}$ is larger than a positive threshold selected by Ps. Furthermore, Ps evaluates the probability that Ad cheats. If both are significantly large, Ps can seek for the intervene of a trusted party for auditing. Note that anomaly detection produces false positives and false negatives. More details are explained in Section 6.2.
3. Ps and Ad settle the payment. If $\hat{C} = n$, they reset the entire protocol. Namely, Ps issues a new batch of hash tokens to Ad by running a new round of Initialization. Otherwise, Ps and Ad continue to use the present batch of tokens until n transactions are performed.

6 Analysis

Recall that we do not consider collusion attacks. Therefore in order to produce a valid receipt, Ad has to either honestly execute all Transaction with fresh tokens or to cheat by using duplicated tokens. For easiness of discussion, we refer to the first type of attack by *withholding attack* and the second type of attack by *duplication attack*. Let $P_{w,z}$ denote the maximum probability of detection failure when Ad runs withholding attacks for z transactions only; and $P_{d,z}$ denote the maximum probability of detection failure when Ad runs duplication attacks for z transactions only.

CAVEAT. The withholding attack actually does not benefit Ad in the long run, as it will not get new authorization tokens until the current batch of n tokens are used up. If Ad undercounts for the present, it has to inflate the count back in the future. Moreover, as shown later in Lemma 3, the duplication attack is more advantageous to Ad. Although our scheme has such deterrence, we still include the withholding attack into our analysis for the completeness of the discussion.

We now first proceed to analyze the success probability that Ps catches duplication attacks. We then analyze how Ps further detects fraud by finding anomaly. Finally, we analyze other security properties and performance.

6.1 Token Duplication Detection

To prove the security strength of the scheme, we first show that one hash token only results in one unique random number in Ad's Schnorr signature (γ, β).

Lemma 1. *Let* $\sigma = (t, M, \gamma, \beta, \psi, \lambda_1, \cdots, \lambda_l)$ *and* $\sigma' = (t', M', \gamma', \beta', \psi', \lambda'_1, \cdots, \lambda'_l,)$ *be the user receipts for two transactions. If* $t = t'$ *and* $V(\sigma) = V(\sigma') = 1$, *then* $\mathsf{Pr}(g^{\beta}y^{\gamma} \neq g^{\bar{\beta}}y^{\bar{\gamma}})$ *is negligible.*

Proof. The proof is trivial. We show that $\mathsf{Pr}(g^{\beta}y^{\gamma} \neq g^{\beta'}y^{\gamma'}) < 2\epsilon$, where ϵ denotes the error probability of the signature of knowledge scheme.

Let $W = g^{\beta}y^{\gamma}$ and $W' = g^{\beta'}y^{\gamma'}$. Since $V(\sigma) = V(\sigma') = 1$, we have

$$\mathsf{Pr}[\text{DLOGLOG}(W,g,t) = \text{DLOG}(v,u)] = \mathsf{Pr}[\text{DLOGLOG}(W',g,t') = \text{DLOG}(v,u)] = 1 - \epsilon$$

according to the soundness definition of signature of knowledge. Since $t = t'$, we have $\mathsf{Pr}(W \neq W') < 1 - (1-\epsilon)^2 < 2\epsilon$. Therefore, if the signature of knowledge scheme is sufficiently sound, the probability that one hash token results in different randomness is negligible. □

Thus, if two users have verified their receipts with the same hash token t_C in $\mathsf{Transaction}$, the probability that Ad has used the same randomness g^r in generating two different Schnorr signatures is overwhelming. This serves as the basis for Ps to catch Ad's fraud. We summarize it in the following lemma.

Lemma 2. *If* $\rho < 1/3$, *then* Ad*'s duplication attack with* z *tokens can escape detection with the maximum probability being* $(1-\rho^2)^z$. *Namely,* $P_{d,z} = (1-\rho^2)^z$.

Proof. (sketch) For each of the z duplication attacks, Ad computes a Schnorr signature with a duplicated hash token which has been used in another signature. According to Lemma 1, Ad will be caught if any two users return the same hash token.

To cheat z times, Ad has two exclusive tactics. One is to use z distinct tokens with each being used exactly twice, i.e. reused exactly once. The other approach is that there exists at least one token which is used more than twice in total. The first tactic is more optimal for Ad than the second one. It can be proved by using an induction on z to compare the two probabilities of Ad's successful evasion. A rigorous proof is in Appendix A. The intuition is as follows. For the first tactic, only when both users receiving the same hash token return their receipts to Ps, can Ad be caught. In contrast, for the second one, it allows *polynomially* more combinations of feedbacks. Therefore, the first tactic maximizes the likelihood for Ad to evade detection.

To be conservative, we evaluate the scheme's resistance to Ad's best tactic, i.e. no hash token is used more than twice. Therefore, there are z pairs of Schnorr signatures which share a common hash token. Note that every user runs $\mathsf{Feedback}$ independently. Hence, the probability that two users with the same hash token return is ρ^2. As a result, Ad can escape after cheating z times with a probability $(1-\rho^2)^z$. □

Next, we show that the duplication attack is more advantageous to Ad, as it allows Ad to escape fraud detection with a higher probability than the withholding the same amount of transactions.

Lemma 3. *For a deflation fraud with z transactions,* Ad *has a higher probability to escape detection by the duplication attack only than by the withholding attack only for z transactions. Namely, $P_{w,z} < P_{d,z}$.*

Proof. In proving Lemma 2, we have shown that $P_{d,z} = (1-\rho^2)^z$. The withholding attack can only be detected in Report, when there exists $(C, I_{click}, \gamma, \beta)$ in Λ s.t. $C > \hat{C}$. Thus, for withholding z transactions, Ad evades detection as long as none of those z users ignored by Ad runs $\mathsf{Feedback}$ with Ps. The probability of that event to occur is $\mathsf{Pr}_{w,z} = (1-\rho)^z$, which is less than $P_{d,z}$. □

From the lemmas above, we show the security strength of the proposed scheme in the following theorem.

Theorem 1. *The proposed scheme is $(z, (1-\rho^2)^z)$-deflation-resistant for $\rho < 1/3$.*

Proof. Suppose that Ad intends to deflate the transaction count by z in total. Without loss of generality, suppose that Ad withholds z_1 transactions and duplications z_2 tokens, s.t. $z = z_1 + z_2$ and $z_1, z_2 \geq 0$. Ad evades detection when Ps fails to detect both attacks. Thus $\mathsf{Pr}[\text{detection fail}|z] = P_{w,z_1} \cdot P_{d,z_2}$. From Lemma 3, $\mathsf{Pr}[\text{detection fail}|z] < P_{d,z_1} \cdot P_{d,z_2} = P_{d,z} = (1-\rho^2)^z$.

Alternative, we can prove this by showing $\mathsf{Pr}[\text{detection fail}|z] = (1-\rho)^{z_1}(1-\rho^2)^{z_2} = (1-\rho)^z(1+\rho)^{z_2}$ which reaches its upper bound when $z_2 = z$. Thus, the propose scheme is $(z, (1-\rho^2)^z)$-deflation-resistant. □

Remark 1. Our scheme deters deflation fraud by revealing Ad's private key, in a similar fashion as in offline detection of double-spending e-cash [3]. Suppose that there exit two distinct user receipts (β_1, γ_1) and (β_2, γ_2) using the same random number r, i.e. $\beta_1 = r - x \cdot \gamma_1 \bmod q$ and $\beta_2 = r - x \cdot \gamma_2 \bmod q$. Thus, it is straightforward to derive x by computing $(\beta_1 - \beta_2)(\gamma_2 - \gamma_1)^{-1} \bmod q$. We argue that the private key extraction is a deterrence to Ad's duplication attacks.

Remark 2. It is an interesting challenge to design an efficient and practical $(1, 0)$-deflation-resistant deflation fraud detection scheme. The difficulty stems from the efficiency requirement. We observe that it seems infeasible to detect all frauds by using cryptographic techniques alone, unless *all* transactions data are known to Ps.

Remark 3. When $\rho \geq 1/3$, the probability to detect fraud is even higher, because Ps collects more information with a higher ρ. Nonetheless we do not have a close form formula to describe the success probability. Therefore, the constraint on ρ in the above theorem is not an advantage to the adversary. We also argue that a practical ρ is typically small due to Ps's expense constraints.

6.2 Anomaly Detection

Even though Ad's attack possibly evades the token-duplication detection in both Feedback and Report, Ps can also detect the fraud by using probability analysis. Different from the previous detection which relies on the non-repudiable evidences, the probability analysis only indicates the likelihood of Ad's cheating.

Detection using C_f. Recall that from Feedback execution, Ps concludes that there are at least C_f transactions. Based on C_f, Ps estimates the expected N using the following theorem.

Theorem 2. *Let N be the random variable denoting the total number of transactions. Let X be the random variable denoting the maximum transaction count received in Feedback, whose space is $[0, N]$. Then $\mathbf{E}[N] > \mathbf{E}(X) + (1-\rho)/\rho$.*

Proof. Since every user is independent in running Feedback, thus $\mathsf{Pr}[X = c|N = n] = \rho(1-\rho)^{n-c}$. Thus, we have $\mathbf{E}[X|N = n] = \sum_{i=0}^{n} i\rho(1-\rho)^{n-i} = n - (1-\rho)/\rho - (1-\rho)^{n+1}/\rho$. Therefore, we have

$$\begin{aligned}
\mathbf{E}[X] &= \mathbf{E}[\mathbf{E}[X|N = n]] = \sum_n \mathbf{E}[X|N = n]\mathsf{Pr}[N = n] \\
&= \sum_n (n - (1-\rho)/\rho - (1-\rho)^{n+1}/\rho)\mathsf{Pr}[N = n] \\
&= \mathbf{E}[N] - (1-\rho)/\rho \sum_n \mathsf{Pr}[N = n] - \frac{1}{\rho}\sum_n (1-\rho)^{n+1}\mathsf{Pr}[N = n] \\
&< \mathbf{E}[N] - (1-\rho)/\rho
\end{aligned}$$

Therefore, $\mathbf{E}[N] > \mathbf{E}(X) + (1-\rho)/\rho$ which completes the proof. □

In fact, if N is known to be in the range $(N_0, +\infty)$ where $(1-\rho)^{N_0+1} \approx 0$, Ps can even conclude that $\mathbf{E}[N] \approx \mathbf{E}[X] + (1-\rho)/\rho$. Next, we analyze how to estimate $\mathbf{E}[X]$ from the known information.

Let ϵ_A be Ad's risk threshold. Let Z be the total amount of duplication attacks Ad feels safe to perform in order to evade the token-duplication detection. Namely $(1-\rho^2)^Z < \epsilon_A$. To maximize its deflation fraud, Ad attempts to minimize C_f as much as possible. Since each user runs Feedback independently, Ad cannot predict which user would return a receipt. We remark that for every transaction, Ad's cheating is prior to the user's Feedback. Thus, Ad's Z cheating is equivalent to reduce the transaction count by Z, since Z tokens are duplicated. Thus, the actual $\mathbf{E}[X]$ is only $\mathbf{E}[N] - Z - (1-\rho)/\rho$.

As Ps knows ρ, it can estimate $Z < z_0$ where $(1-\rho^2)^{z_0} \approx \epsilon_A$. Then, it estimates the expected N as $C_f + z_0 + (1-\rho)/\rho$ according to the above theorem. If the difference between the calculated $\mathbf{E}[N]$ and $\hat{C}$ is much larger than a predetermined threshold, Ps suspects that Ad has deflation frauds.

Detection using I_f. From the protocol execution, Ps observes that there exist $M - I_f$ users whose click counts are larger than I_f and none of them offers feedback. Let $d = M - I_f$ and let Ω denote this set of d users. According to Ad's report, there exist $k = \hat{C} - C_f$ transactions generated by Ω. Ps checks whether Ad cheats by executing the following steps.

1. Ps computes $\xi = C_f/I_f$ as the expected probability for one user making a transaction following an advertisement click.
2. Let P_i denote the probability of the event that there are i users in Ω who have run Transaction with Ad. Ps computes $P_i = \binom{d}{i}\xi^i(1-\xi)^{d-i}$.
3. Let a random variable A denote the number of feedbacks from Ω, and a random variable B denote the number of transactions from Ω with no feedback being sent to Ps. Let χ denote $\mathsf{Pr}[B > k | A = 0]$. Essentially, χ is the conditional probability that Ad has under-reported in the present circumstance. Ps evaluates χ as

$$\chi = \frac{\sum_{i=k+1}^{d} P_i(1-\rho)^i}{\sum_{i=0}^{d} P_i(1-\rho)^i} = \frac{\sum_{i=k+1}^{d} \binom{d}{i}\xi^i(1-\xi)^{d-i}(1-\rho)^i}{(1-\xi\rho)^d}$$

If χ is larger than a positive threshold, e.g. 0.5, Ps suspects that Ad cheats.

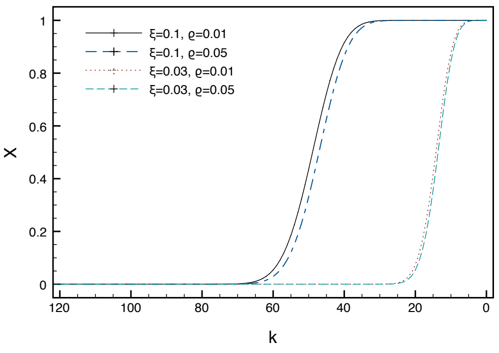
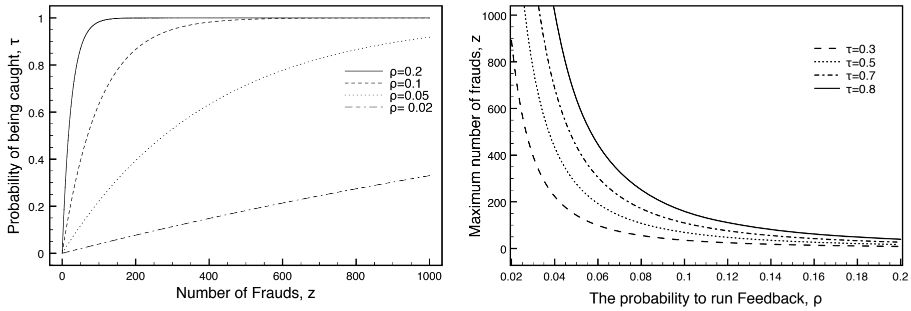


Fig. 4. χ grows when k decreases, for $d = 500$, $\xi = 0.1, 0.03$, and $\rho = 0.01, 0.05$

In summary, Ps can detect deflation cheating according to C_f and I_f. With a chosen ρ, the expected $k = \hat{C} - C_f$ is at least $(1-\rho)/\rho$. If k is significantly less than its expected value, Ps calculates χ. Figure 4 plots the relation between k and χ for $d = 500$ and different ρ, ξ. It shows that if Ad under-reports $\hat{C}$, a smaller k will result in a larger χ. In the worst case, $\rho = 0.01$ and $\xi = 0.03$ where the expected k is 99, Ad can hide about 80 transactions if the threshold for χ is 0.5. In the best case, $\rho = 0.05$ and $\xi = 0.1$ where the expected k is only 19, any deflation will be detected since χ is almost 1 when $k < 19$.

Note that ξ is application specific. Ps can only tune ρ for the desired security strength. Next, we show how to tune ρ by taking all factors into consideration.

Tuning Security and Cost. On the one hand, Ps prefers a larger ρ so that more users return their receipts, and as a result, the chance of successful fraud detection is higher; on the other hand, a larger ρ implies a higher financial cost for more rewards. We show below how Ps strikes a balance between security and cost by tuning ρ.[3] Let τ denote the probability to successfully catch the dishonest Ad by protocol execution, i.e. $\tau = 1 - (1 - \rho^2)^z$.



(a) Exponential growth of τ with z using different ρ.

(b) Fraud deterrence with different ρ

Fig. 5. Balancing security and cost by tuning the redemption probability

Figure 5(a) depicts the exponential correlation between τ and the number of frauds z, when different fixed ρ are chosen by Ps. As evident in the figure, a low ρ results in a reasonably high probability in detecting a small amount of deflation frauds. Note that in a large scale web advertising, the volume of transactions is usually in thousands or even more. With around one tenth of all users run Feedback, it is expected for our scheme to catch any fraud which accounts for more than one percent of the total volume.

Suppose that Ad sets up a risk threshold τ and will not make frauds causing a risk higher than τ. Figure 5(b) shows the maximum frauds Ad would have in different risk profiles. The figure depicts the correlation between z and ρ with respect to different constant risk profile τ. For instance, if 10% of the users redeem their receipts, Ad can only cheat around 80 transactions to keep its risk below 50%.

In summary, the anomaly detection has a weaker demand for ρ, than duplication detection does. Therefore, Ps can set ρ according to its own risk profile. Generally, for applications with thousands of transactions, Ps only needs to set ρ between 0.05 and 0.1, which provides sufficient security assurance.

[3] The more rewards are offered by Ps, the higher ρ for user redemption. The exact relation between ρ and financial cost is beyond the scope of this paper.

6.3 Other Security Properties

Unforgeability and Non-repudiation. A malicious publisher Ps may collude with corrupted users to forge Ad's signatures so as to frame Ad. We show that the Schnorr signature from Ad is still existentially unforgeable against chosen message attacks. In other words, our scheme does not compromise the security of the standard Schnorr signature scheme.

Theorem 3. *With the discrete logarithm assumption in $\mathbb{G}$ and $\mathbb{G}_h$, Ad's Schnorr signature in $\mathsf{Transaction}$ is secure against existential forgery attacks under the random oracle model.*

Proof. (sketch) Let $\mathcal{A}$ be the algorithm forging Ad's Schnorr signature. $\mathcal{A}$ is allowed to access a signature oracle $\mathcal{O}_s$ with a query m, and is allowed to query a random oracle $\mathcal{O}$. $\mathcal{A}$'s goal is to forge a signature (γ^*, β^*) on a message m^* which is not sent to $\mathcal{O}_s$. We show that if $\mathcal{A}$ succeeds in forging a signature, we can construct an algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve the discrete logarithm problem in $\mathbb{G}$.

$\mathcal{B}$ is given (g, y, p, q, q') where p, q, q' are large primes satisfying $q = 2q' + 1$ and $q|p-1$, and g's order in $\mathbb{Z}_p^*$ is q. $\mathcal{B}$'s goal is to find x such that $y = g^x \bmod p$. It simulates $\mathcal{O}$ and $\mathcal{O}_s$ and interacts with $\mathcal{A}$. $\mathcal{B}$ sets Ad's public key as y. Then it randomly chooses $h \in \mathbb{Z}_q$, picks $u \in \mathbb{Z}_{q'}$ and computes $v = h^u \bmod q$. $\mathcal{B}$ initializes $\mathcal{A}$ with (y, v, g, h, p, q). The hash function $\mathcal{H}()$ is modeled as a random oracle.

- When $\mathcal{A}$ queries $\mathcal{O}_s$ with t, M, $\mathcal{B}$'s simulation is done as follows:
  1. Select $\gamma, \beta \in_R \mathbb{Z}_q$; Select $\alpha \in_R \{0,1\}^k$;
  2. Set $\mathcal{O}$ such that $\gamma = \mathcal{H}(t\|y\|g\|M\|g^\beta y^\gamma)$ and $\alpha = \mathcal{H}(t\|\mathsf{Ad})$. Namely, $\mathcal{O}$ stores $(t\|y\|g\|M\|g^\beta y^\gamma, \gamma)$ and $(t\|\mathsf{Ad}, \alpha)$ into its local table.
  3. Select $\psi, \lambda_1, \cdots, \lambda_l \in_R \mathbb{Z}_{q'}$;
  4. For $i = 1, \cdots, l$, set $w_i = h^{\lambda_i}$ and $w_i' = g^{\alpha^{\lambda_i}}$ if $\psi[i] = 0$; otherwise set $w_i = v^{\lambda_i}$ and $w_i' = (g^\beta y^\gamma)^{\alpha^{\lambda_i}}$;
  5. Set $\mathcal{O}$ such that $\psi = \mathcal{H}(v\|h\|g^\beta y^\gamma\|g\|t\|w_1\|\cdots\|w_l\|w_1'\|\cdots\|w_l')$, and store $(v\|h\|g^\beta y^\gamma\|g\|t\|w_1\|\cdots\|w_l\|w_1'\|\cdots\|w_l', \psi)$ in the local table and then return $(\alpha, \gamma, \beta, \lambda_1, \cdots, \lambda_l)$ to $\mathcal{A}$.

  Note that $\mathcal{A}$ is not able to distinguish whether the tuples returned by $\mathcal{B}$ are simulated results or from a real protocol execution, because all are from the same uniform distribution. Note that $g^\beta y^\gamma$ in our construction is random as well, because α is an output from the random oracle.
- When $\mathcal{A}$ queries $\mathcal{O}$ with a query m, $\mathcal{B}$ searches its table. If there is an entry (m, m_h), it returns m_h to $\mathcal{A}$. If no such entry is found, $\mathcal{B}$ randomly picks $m_h \in_R \{0,1\}^k$, returns m_h and stores (m, m_h) into the table.

Finally, $\mathcal{A}$ halts and outputs a Schnorr signature $(\gamma^*, \beta^* = r^* - xh^*)$. By the Forking Lemma, $\mathcal{A}$ can produce another valid signature $(\gamma_1^*, \beta_1^* = r^* - xh_1^*)$ with a non-negligible probability by rewinding the random oracle. Therefore, $\mathcal{B}$ can solve the discrete logarithm problem by computing $x = (\beta_1^* - \beta^*)/(h^* - h_1^*)$. □

Theorem 3 shows that neither Ps nor users can forge Ad's Schnorr signatures. This has twofold implications. On the one hand, it provides security assurance for Ad since its signatures are not be forged; on the other hand it implies that Ad cannot repudiate its fraud, if it is caught with two Schnorr signatures using the same hash token.

Privacy. If Ad uses hash tokens along the hash chain, a user knows the count of transactions. In case the exposure of transaction count is undesirable for Ad, the following minor revision on the protocol can be applied.

In Initialization, both Ad and Ps can agree on a pseudo-random permutation function F_s keyed by a shared secret s. For $C \in [1, n]$, $F_s(C)$ maps C into another random number in $[1, n]$. In Transaction, Ad picks a hash token based on $F_s(C)$. Without knowledge of s, $\mathcal{U}$ cannot infer the total amount of hash tokens used by Ad. In Feedback, Ps recovers C using the hash token and $F_s^{-1}(C)$. Note that the revelation of unused hash tokens poses no threat to either Ad or Ps.

6.4 Performance

The amount of information stored by Ad are only its secret keys (x, u), a transaction counter C and a hash chain seed T. Therefore, the storage cost is constant with respect to the amount of transactions. In terms of computation cost, Ad makes $3l + 1$ modular exponentiations. Nonetheless, if Ad needs to reduce its real-time response time, it can pre-compute all the modular exponentiations, then only performs modular additions and hash computations to issue receipts. In terms of communication cost, Ad has a constant communication cost with Ps during Initialization and Report, and sends $(l + 3) \cdot |q|$ bits in Transaction, which is nearly two kilobytes in a practical setting.

The main cost of Ps is its storage overhead for Λ, whose expected size is asymptotically linear with ρN. The main cost for users is their computations for signature verification in Transaction, which involve $3l + 1$ modular exponentiations. Therefore, our scheme may not be suitable for devices with constraints in computation resource, such as mobile phones.

7 Conclusion

In conclusion, we propose a deflation fraud detection scheme for the CPA advertising model, taking a hybrid approach based on cryptography and probability techniques. For any z amount of deflation cheating in the advertiser Ad's transactions, the web publisher Ps can detect it with a success probability at least $1 - (1 - \rho^2)^z$. As a deterrence, Ps can extract Ad's secret signature key. Although in the long run Ad does not benefit from the deflation fraud by withholding transactions, Ps can still detect it if the amount is over a prescribed threshold. Furthermore, Ps can estimate the expected transaction amount N as $C_f + (1 - \rho)/\rho + z_0$ where z_0 is estimated according to Ad's risk profile.

The proposed scheme preserves the simplicity of the existing advertising model, without introducing any third party. It is user-friendly in the sense that users are not required to possess any secrets. Ad's communication, computation, storage cost are all constant. In addition, Ps can tune its security parameter ρ to balance the security and cost.

Acknowledgement

We thank Robert Deng, Jian Wen and Junzhuo Lai for the valuable discussions. We thank anonymous reviewers for their constructive comments. This work is supported by the Office of Research, Singapore Management University.

References

1. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and veriably en-crypted signatures from bilinear maps. In: Proceedings of Advances in Cryptology - EUROCRYPTO (2003)
2. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Proceedings of Advances in Cryptology- ASIACRYPTO (2007)
3. Brands, S.: Untraceable off-line cash in wallet with observers. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 302–318. Springer, Heidelberg (1994)
4. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
5. Chaum, D.: Secret-ballot receipts: True voter-verifiable elections. In: IEEE Security and Privacy (2004)
6. Chaum, D., Ryan, P.Y., Schneider, S.: A practical voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
7. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.: The many faces of publish/subscribe. ACM Computing Surveys 35 (June 2003)
8. Franklin, M., Malkhi, D.: Auditable metering with lightweight security. Journal of Computer Security 6(4) (1998)
9. Gandhi, M., Jakobsson, M., Ratkiewicz, J.: Badvertisements: Stealthy click-fraud with unwitting accessories. Anti-Phishing and Online Fraud, Part I Journal of Digital Forensic Practice 1 (November 2006)
10. Johnson, R., Staddon, J.: Deflation-secure web metering. International Journal of Information and Computer Security 1 (2007)
11. Juels, A., Stamm, S., Jakobsson, M.: Combatting click fraud via premium clicks. In: Proceedings of USENIX Security (2007)
12. Kuhn, M.: Probabilistic counting of large digital signature collections. In: USENIX Security Symposium (2000)
13. Lysyanskaya, A., Ramzan, Z.: Group blind digital signatures: A scalable solution to electronic cash. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 184–197. Springer, Heidelberg (1998)
14. Majumdar, S., Kulkarni, D., Ravishankar, C.: Addressing broker violations of count integrity in publish-subscribe systems. In: Proceedings of ACM Infocom (2007)
15. Masucci, B., Stinson, D.: Efficient metering schemes with pricing. IEEE Transactions on Information Theory 47(7) (2001)

16. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
17. Naor, M., Pinkas, B.: Secure and efficient metering. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 576–590. Springer, Heidelberg (1998)
18. Ogata, W., Kurosawa, K.: Provably secure metering scheme. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, p. 388. Springer, Heidelberg (2000)
19. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
20. Zhang, L., Guan, Y.: Detecting click fraud in pay-per-click stream of online advertising network. In: Proceedings of ICDCS (2008)

Appendix A

Theorem 4. *Suppose that each user has a probability ρ to redeem a receipt. Let P_k denote* Ad*'s escape probability of cheating k times by reusing k distinct hash tokens. Let P'_k denote* Ad*'s maximum escape probability of cheating k times by reusing k' distinct hash tokens, $0 < k' < k$. Therefore, if $\rho < 1/3$, then for all $k > 1$, $P_k > P'_k$.*

Proof. We prove the theorem by using induction on k.
(i) $\mathbf{k = 2}$. It is straightforward to see that $P_2 = (1 - \rho^2)^2$. In addition, we have $k' = 1$. In other words, Ad uses a hash token for three times, two out of which are considered as cheating. Therefore, Ad's fraud can be detected when any two of the three corresponding users redeem their receipts. Thus, $P'_2 = (1-\rho)^3 + 3\rho(1-\rho)^2$. So, $P_2 - P'_2 = \rho^2 - 3\rho^3 = \rho^2(1 - 3\rho) > 0$.
(ii) $\mathbf{k > 2}$. (Induction hypothesis) Suppose that $P_k > P'_k$ for some k. We prove that $P_{k+1} > P'_{k+1}$. Clearly, we have $P_{k+1} = (1 - \rho^2)^{k+1}$. Let U denote any user among the $k + 1$ cheating victims and t denote the hash token in the signature U receives. Suppose that there are c cheats whereby the token t is used. Therefore, Ad can escape detection when these c cheats are not detected and the remaining $k + 1 - c$ cheats are not detected either. Hence, If $c = 1$, $P'_{k+1} = P'_k P_1 < P_k P_1 = P_{k+1}$, due to the induction hypothesis. Otherwise, $P'_{k+1} \leq P'_{k+1-c} P'_c < P_{k+1-c} P_c = P_{k+1}$. Thereby, $P'_{k+1} < P_{k+1}$ which completes the proof. □

Author Index