

Martijn Stam (Ed.)

LNCS 8308

Cryptography and Coding

14th IMA International Conference, IMACC 2013
Oxford, UK, December 2013
Proceedings



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Martijn Stam (Ed.)

Cryptography and Coding

14th IMA International Conference, IMACC 2013
Oxford, UK, December 17-19, 2013
Proceedings



Springer

Volume Editor

Martijn Stam
University of Bristol
Dept. of Computer Science
Woodland Road, Bristol, BS8 1UB, UK
E-mail: martijn.stam@bristol.ac.uk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-45238-3 e-ISSN 978-3-642-45239-0
DOI 10.1007/978-3-642-45239-0
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013954847

CR Subject Classification (1998): E.3, D.4.6, K.6.5, G.1.3, J.1, G.2

LNCS Sublibrary: SL 4 – Security and Cryptology

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

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

Printed on acid-free paper

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

Preface

The 14th IMA Conference on Cryptography and Coding took place at St. Anne's College in Oxford during December 17–19, 2013. The official abbreviation of the biennially held IMA Conference on Cryptography and Coding is IMACC, but to many it is better known under its colloquial epithet “Cirencester,” referring to the many past editions that were held at the Agricultural College at Cirencester. However, this year's edition followed in the 13th edition's footsteps by being based in Oxford (UK) instead. This historical location with its rich academic tradition provided a suitably inspirational backdrop for a successful conference.

The academic program was spread out over three days and included a diverse range of activities and topics. The main part of the program, and all of these proceedings, were contributed talks. In July 2013 a wide variety of submissions were received. These were subsequently subjected to a rigorous review process and only the best submissions in the eyes of the Program Committee were selected for presentation at the conference. Overall, 20 contributed papers were accepted for inclusion in these proceedings. These contributions cover a wide range of topics, including cryptology, coding theory, and their interface. The authors come from the UK, continental Europe, the USA, and Asia and represent both academia and industry.

The Program Committee consisted of a mixture of coding theorists and cryptologists, both from the UK and from abroad, helping maintain an international quality level. I would like to take this opportunity to thank the Program Committee and the external reviewers for their dedicated work. We used IACR's WebSubRev system to aid with the selection of the program and preparation of the proceedings. We are indebted to the IACR, and Shai Halevi in particular, for providing us with this support.

On Wednesday there was an invited talk; many thanks to Patrick Solé for delivering a talk that was entertaining and enriching at the same time. Thanks to our generous sponsor GCHQ we were able to organize a tutorial on the first day of the conference. This tutorial was tailored toward introducing the wonderful world of coding-based cryptography to a wider audience. Many thanks to the invited speakers Vadim Lyubashevsky, Christiane Peters, Peter Schwabe, Jacob Schuldt, Nicolas Sendrier, and Henk van Tilborg for making the tutorial such a worthwhile experience.

On Wednesday a fascinating panel took place on the role of cryptologic education in the wider context of cybersecurity. While history has shown that cryptology can be a great enabler of increased security, it has also taught us that cryptology is only a very small cog in the development of secure IT systems and that it is very hard to get right in practice. The panel discussed what is expected of cryptologic education in building the cybersecurity leaders of tomorrow.

Finally, a word of thanks to those who kindly contributed toward the success of the conference. Firstly, Nigel Smart, Kenny Paterson, and Liqun Chen (all members of the IMACC Steering Committee) for their invaluable advice. Secondly, the people at Springer, in particular Alfred Hofmann and Anna Kramer, for their professionalism when preparing this LNCS volume. Thirdly, Richard Pinch from GCHQ whose input was instrumental in organizing the tutorial session on Tuesday. Finally, and certainly not least, many, many thanks to Lizzi Lake and her team from the IMA for taking care of what one would normally refer to as the “general chair” role.

December 2013

Martijn Stam

Dimitar Jetchev	EPFL, Switzerland
Jon-Lark Kim	Sogang University, Seoul, Korea
Gohar Kyureghyan	University of Magdeburg, Germany
David Naccache	Ecole Normale Supérieure, France
Siaw-Lynn Ng	Royal Holloway, University of London, UK
Matthew G. Parker	University of Bergen, Norway
Raphael C.-W. Phan	Multimedia University, Malaysia
María Naya-Plasencia	Inria Paris-Rocquencourt, France
Joachim Rosenthal	Universität Zurich, Switzerland
Ana Salagean	Loughborough University, UK
Hans Georg Schaathun	Aalesund University College, Norway
Dominique Schroeder	Saarland University, Germany
Ben Smith	Inria LIX, France
John Steinberger	Tsinghua University, China
Stefano Tessaro	UCSB, USA and MIT, USA
Gilles Van Assche	STMicroelectronics, Belgium
Frederik Vercauteren	K.U. Leuven, Belgium
Andreas Winter	Universitat Autònoma de Barcelona , Spain
Xing Chaoping	Nanyang Technological University, Singapore
Kyeongcheol Yang	Pohang University of Science and Technology (POSTECH), Korea

External Reviewers

Anja Becker	Nils Fleischhacker	Yannis Rouselakis
Ayç a Çeşmeliöğlü	Felix Fontein	Jacob Schuldt
Jean-Philippe Aumasson	Dimitar Jetchev	Valentin Suder
Donghoon Chang	Khoongming Khoo	Jaechul Sung
Baudoin Collard	Allison Lewko	
Robert Fitzpatrick	Michael Naehrig	

Table of Contents

Bits and Booleans

Semi-bent Functions from Oval Polynomials	1
<i>Sihem Mesnager</i>	
Efficient Generation of Elementary Sequences	16
<i>David Gardner, Ana Sălăgean, and Raphael C.-W. Phan</i>	

Homomorphic Encryption

On the Homomorphic Computation of Symmetric Cryptographic Primitives	28
<i>Silvia Mella and Ruggero Susella</i>	
Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme	45
<i>Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig</i>	
On the Relationship between Functional Encryption, Obfuscation, and Fully Homomorphic Encryption	65
<i>Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson</i>	

Codes and Applications

On Minimal and Quasi-minimal Linear Codes	85
<i>Gérard D. Cohen, Sihem Mesnager, and Alain Patey</i>	
A Code-Based Undeniable Signature Scheme	99
<i>Carlos Aguilar-Melchor, Slim Bettaieb, Philippe Gaborit, and Julien Schrek</i>	

Cryptanalysis

Filtered Nonlinear Cryptanalysis of Reduced-Round Serpent, and the Wrong-Key Randomization Hypothesis	120
<i>James McLaughlin and John A. Clark</i>	

Differential Cryptanalysis of Keccak Variants 141
Stefan Kölbl, Florian Mendel, Tomislav Nad, and Martin Schläffer

Recovering Private Keys Generated with Weak PRNGs 158
Pierre-Alain Fouque, Mehdi Tibouchi, and Jean-Christophe Zapalowicz

Protecting against Leakage

A Leakage-Resilient Pairing-Based Variant of the Schnorr Signature Scheme 173
David Galindo and Srinivas Vivek

High-Order Masking by Using Coding Theory and Its Application to AES 193
Guilhem Castagnos, Soline Renner, and Gilles Zémor

Hash Functions

Hashing Mode Using a Lightweight Blockcipher 213
Hidenori Kuwakado and Shoichi Hirose

Indifferentiability of Double Length Compression Functions 232
Bart Mennink

Security Amplification against Meet-in-the-Middle Attacks Using Whitening 252
Pierre-Alain Fouque and Pierre Karpman

Key Issues

Secure Key Management in the Cloud 270
Ivan Damgård, Thomas P. Jakobsen, Jesper Buus Nielsen, and Jakob I. Pagter

Estimating Key Sizes for High Dimensional Lattice-Based Systems 290
Joop van de Pol and Nigel P. Smart

Public Key Primitives

Sub-linear Blind Ring Signatures without Random Oracles 304
Essam M. Ghadafi

Constructions of Signcryption in the Multi-user Setting from Identity-Based Encryption	324
<i>Rintaro Nakano and Junji Shikata</i>	
Anonymous Constant-Size Ciphertext HIBE from Asymmetric Pairings	344
<i>Somindu C. Ramanna and Palash Sarkar</i>	
Author Index	365

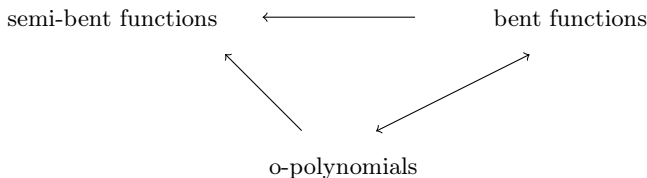
Semi-bent Functions from Oval Polynomials

Sihem Mesnager

LAGA (Laboratoire Analyse, Géométrie et Applications), UMR 7539, CNRS,
Department of Mathematics, University of Paris XIII and University of Paris VIII,
Sorbonne Paris Cité, France
smesnager@univ-paris8.fr

Abstract. Although there are strong links between finite geometry and coding theory (it has been proved since the 1960's that all these connections between the two areas are important from a theoretical point of view and for applications), the connections between finite geometry and cryptography remain little studied. In 2011, Carlet and Mesnager have showed that projective finite geometry can also be useful in constructing significant cryptographic primitives such as plateaued Boolean functions. Two important classes of plateaued Boolean functions are those of bent functions and of semi-bent functions, due to their algebraic and combinatorial properties. In this paper, we show that oval polynomials (which are closely related to the hyperovals of the projective plane) give rise to several new constructions of infinite classes of semi-bent Boolean functions in even dimension.

The following diagram gives an indication of the main topics and interconnections arising in this paper.



Keywords: Boolean functions, Polynomial form, Walsh transform, Bent functions, Semi-bent functions, Oval polynomials, Hyperovals.

1 Introduction

Few connections between cryptography and projective finite geometry have been settled in the literature¹. Very recently, it has been shown how finite geometry

¹ For some applications of finite geometry in cryptography, the reader can see the recent paper of Klein and Storme [12].

can contribute to the theory of Boolean and vectorial functions and its application in symmetric cryptography. Boolean functions, that is, \mathbb{F}_2 -valued functions defined on the vector space \mathbb{F}_2^n (all binary words of a given length n) or the Galois field \mathbb{F}_{2^n} of order 2^n , are used in symmetric cryptosystems (more precisely, in the S-boxes of block ciphers and in the pseudo-random generators of stream ciphers). They play a critical role in their security. The class of plateaued functions over \mathbb{F}_{2^n} (or r -plateaued functions, $0 < r < n$) has been introduced in 2001 and studied actively in several papers as good candidates for designing cryptographic functions. Two important classes of plateaued Boolean functions are those of bent functions (0-plateaued functions) and of semi-bent functions (2-plateaued functions), due to their algebraic and combinatorial properties. Bent functions are maximally nonlinear Boolean functions with an even number of variables. They were introduced and firstly studied by Dillon [7] in 1974 and Rothaus [11] in 1976. For their own sake as interesting combinatorial objects, but also because of their relations to coding theory (Reed-Muller codes) and applications in cryptography (design of stream ciphers), they have attracted a lot of research, especially in the last 20 years. Semi-bent functions have been introduced by Chee, Lee and Kim [6] and previously investigated under the name of three-valued almost optimal Boolean functions [1]. Very recently, the notion of semi-bentness has been actively studied by many authors and the development of the theory of semi-bent functions has increased. The motivation of their study is firstly related for their use in cryptography. Indeed, unlike bent functions, semi-bent functions can also be balanced and resilient. They also possess various desirable characteristics such as a low Hadamard transform (which provides protection against fast correlation attacks [9] and linear cryptanalysis [8]), have low auto-correlation, satisfy the propagation criteria and have high algebraic degree. Secondly, besides their practical use in cryptography, they are also widely used in code division multiple access (CDMA) communication systems for sequence design. In even dimension, constructions of families of quadratic semi-bent functions have been provided by Charpin et al. [5] and families of semi-bent functions with optimal algebraic degree have been presented in [10] and in [4].

In this paper, we focus our attention on significant cryptographic functions built via *oval-polynomials* (o-polynomials, for short). The notion of o-polynomial comes from finite projective geometry and is closely related to the *hyperovals* of the projective space $PG_2(q)$ of dimension 2 (finite projective plane) over the Galois field \mathbb{F}_q . We establish connections between semi-bent functions and o-polynomials in characteristic 2 and show how these o-polynomials give rise to primary constructions and secondary-like constructions of infinite classes of semi-bent functions. The paper is organized as follows. We start with presenting in Section 2 the needed basic background on Boolean functions (Subsection 2.1) and introduce briefly some theoretical background related to finite geometry (Subsection 2.2). In Section 3, we begin with presenting a construction of semi-bent functions derived from [4] (Theorem 7) as sums of two bent functions (one of them lies in the well known class of partial spreads). Next, we present two

new infinite families of semi-bent functions (Theorem 8 and Theorem 13). Both families lie in the class of Maiorana-McFarland. Another important feature of the first family function is that its elements are sums of two bent functions. In [3], we have shown that the set of o-polynomials is invariant under some transformations that we have explicated. We therefore investigate the question whether these transformations lead to semi-bent functions that are extended affine-equivalent to semi-bent functions of the form stated in Theorem 8 or not. We next show that the constructions presented in Theorem 8 and Theorem 13 can be generalized (Theorem 14 and Theorem 16). That allows us to exhibit a third new family of semi-bent functions (Theorem 15). Finally, we show that the indirect sum introduced in [2] (initially introduced to construct new bent functions from bent functions) can be used to construct further semi-bent functions from those of Theorem 8 and the bent functions of the class introduced in [3] (Theorem 17). The particularity of the bent and semi-bent functions used in that construction is that they are all defined from o-polynomials.

2 Notation and Preliminaries

For any set E , $E^* = E \setminus \{0\}$ and $\#E$ will denote the cardinality of E . In the paper, \mathbb{F}_q denotes the finite field of order q .

2.1 Background on Theory of Boolean Functions and Basic Definitions.

Boolean Functions on \mathbb{F}_{2^n} and Trace Representation

A Boolean function f can be considered as an \mathbb{F}_2 -valued function on the Galois field \mathbb{F}_{2^n} of order 2^n . The *Hamming weight* of f is defined as $wt(f) := \#\{x \in \mathbb{F}_{2^n} \mid f(x) = 1\}$. For any positive integer k dividing n , the field trace from \mathbb{F}_{2^n} to \mathbb{F}_{2^k} , denoted by Tr_k^n , can be explicitly defined as $Tr_k^n(x) = \sum_{i=0}^{\frac{n}{k}-1} x^{2^{ik}}$. In particular, we denote the *absolute trace* of an element $x \in \mathbb{F}_{2^n}$ by $Tr_1^n(x) = \sum_{i=0}^{n-1} x^{2^i}$. The bivariate representation of Boolean functions is defined as follows: we identify \mathbb{F}_{2^n} with $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ (where $n = 2m$) and we consider then the input to f as an ordered pair (x, y) of elements of \mathbb{F}_{2^m} . There exists a unique bivariate polynomial over \mathbb{F}_{2^m} : $f(x, y) = \sum_{0 \leq i, j \leq 2^m-1} a_{i,j} x^i y^j$ such that f is the bivariate polynomial function over \mathbb{F}_{2^m} associated to it. The algebraic degree of f is equal to the maximum 2-weight $w_2(i) + w_2(j)$ for which $a_{i,j} \neq 0$, where the 2-weight $w_2(j)$ of an integer j is equal to the number of 1's in its binary expansion. f being Boolean, its bivariate representation can be written in the form $f(x, y) = Tr_1^m(P(x, y))$ where $P(x, y)$ is some polynomial over \mathbb{F}_{2^m} .

Walsh–Hadamard Transform, Bentness and Semi-bentness.

We denote by $\chi : x \in \mathbb{F}_2 \mapsto (-1)^x \in \{-1, 1\}$ the additive character of \mathbb{F}_2 . The *Walsh–Hadamard transform* of f is the discrete Fourier transform of $\chi_f = \chi \circ f$, whose value at $\omega \in \mathbb{F}_{2^n}$ is defined as

$$\widehat{\chi}_f(\omega) = \sum_{x \in \mathbb{F}_{2^n}} \chi(f(x) + \text{Tr}_1^n(\omega x)), \omega \in \mathbb{F}_{2^n}.$$

The notion of bentness plays an important role for Boolean functions in cryptography. It is related to the notion of *nonlinearity*. Bent functions are maximally non-linear Boolean functions. They can be defined in terms of the values of their Hadamard Walsh transform as follows.

Definition 1. *Let n be an even integer. An n -variable Boolean function f is bent if and only if its Walsh transform satisfies $\widehat{\chi}_f(a) = \pm 2^{\frac{n}{2}}$ for all $a \in \mathbb{F}_{2^n}$.*

Semi-bent functions on \mathbb{F}_{2^n} exist for n even or n odd. When n is even, such functions are defined as follows in terms of the values of its Hadamard Walsh transform.

Definition 2 (Semi-bent function). *A Boolean function $f : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2$ (n even) is said to be semi-bent if $\widehat{\chi}_f(\omega) \in \{0, \pm 2^{\frac{n+2}{2}}\}$, for all $\omega \in \mathbb{F}_{2^n}$.*

2.2 Some Background on Finite Geometry and Basic Definitions

Let \mathbb{F}_q be a finite field of q elements. Consider the equivalence classes of the elements of $\mathbb{F}_q^{n+1} \setminus \{0\}$ given by the following relation:

$$x \sim y \iff \exists \lambda \in \mathbb{F}_q^* : \lambda(x_0, \dots, x_n) = (y_0, \dots, y_n)$$

for $x, y \in \mathbb{F}_q^{n+1} \setminus \{0\}$. The aforementioned equivalence relation divides the elements of $\mathbb{F}_q^{n+1} \setminus \{0\}$ into $\frac{q^{n+1}-1}{q-1}$ equivalence classes, called *points*. The set of equivalence classes together with the incidence relation (which is reflexive and symmetric) is called n -dimensional projective space over \mathbb{F}_q , and is denoted by $PG_n(q)$. The one-dimensional subspaces of \mathbb{F}_q^{n+1} are then the points and the two-dimensional subspaces of \mathbb{F}_q^{n+1} are called the lines. In the paper, we are only dealing with finite fields \mathbb{F}_q with $q = 2^n$, where n is a positive integer. A hyperoval in $PG_2(2^n)$ can be defined as follows.

Definition 3 (Hyperoval). *A hyperoval in $PG_2(2^n)$ is a set of $2^n + 2$ points, no three points of them are collinear (that is, lie in a line²).*

A certain type of polynomials on \mathbb{F}_{2^n} give rise to hyperovals in $PG_2(2^n)$:

Definition 4. *an oval polynomial on \mathbb{F}_{2^n} is a polynomial G on \mathbb{F}_{2^n} such that the set of points $\{(1, t, G(t)), t \in \mathbb{F}_2^n\} \cup \{(0, 0, 1), (0, 1, 0)\}$ (denoted by $D(G)$) forms a hyperoval of $PG_2(2^n)$ (for short, an o-polynomial).*

There is a close connection between the hyperovals and the o-polynomials since a hyperoval of $PG_2(2^n)$ can be represented by $D(G)$ where G is an o-polynomial on \mathbb{F}_{2^n} . In fact, there exists a necessary and sufficient condition for a mapping over \mathbb{F}_{2^n} to give a hyperoval of $PG_2(2^n)$. This leads to a reformulation of the definition of an o-polynomial given as follows.

² We say a point $p = (x_0, \dots, x_n)$ is on a line $L[y_0, \dots, y_n]$ if and only if $x_0 y_0 + x_1 y_1 + \dots + x_n y_n = 0$.

Definition 5. A permutation polynomial G over \mathbb{F}_{2^n} is an o -polynomial if, for every $\gamma \in \mathbb{F}_{2^n}$, the function

$$z \in \mathbb{F}_{2^n} \mapsto \begin{cases} \frac{G(z+\gamma)+G(\gamma)}{z} & \text{if } z \neq 0 \\ 0 & \text{if } z = 0 \end{cases}$$

is a permutation of \mathbb{F}_{2^n} .

Remark 6. Let us indicate that, if G is an o -polynomial over \mathbb{F}_{2^n} then, $z \in \mathbb{F}_{2^n} \mapsto G(z) + \alpha z$ is 2-to-1 for every $\alpha \in \mathbb{F}_{2^n}^*$. We shall use this property in Section 3 and notably to establish Theorem 7, Theorem 8, Theorem 13 and Theorem 14.

We give in Table 1 the list, up to equivalence, of the known o -polynomials on \mathbb{F}_{2^m} .

Table 1. List of o -polynomials G on \mathbb{F}_{2^m} and their inverse G^{-1}

Expression of the o -polynomial G	Conditions	Expression of the o -polynomial G^{-1}
$G(z) = z^6$	m odd.	$G^{-1}(z) = z^{1/6}$
$G(z) = z^{3 \cdot 2^k + 4}$	m odd, $k = \frac{m+1}{2}$.	$G^{-1}(z) = z^{3 \cdot 2^{k-1} - 2}$
$G(z) = z^{2^k + 2^{2k}}$	$k = \frac{m+1}{4}$, $m \equiv 3 \pmod{4}$.	$G^{-1}(z) = z^{1 - 2^{3k-1} + 2^{2k} - 2^k}$
$G(z) = z^{2^{2k+1} + 2^{3k+1}}$	$k = \frac{m-1}{4}$, $m \equiv 1 \pmod{4}$.	$G^{-1}(z) = z^{1 - 2^{3k+1} + 2^{2k+1} - 2^k}$
$G(z) = z^{2^k} + z^{2^k+2} + z^{3 \cdot 2^k + 4}$	$k = \frac{m+1}{2}$, m odd.	$G^{-1}(z) = z \left(z^{2^k+1} + z^3 + z \right)^{2^{k-1}-1}$
$G(z) = z^{\frac{1}{6}} + z^{\frac{1}{2}} + z^{\frac{5}{6}}$	m odd	$(D_{\frac{1}{5}}(z))^6$
$\frac{G(z)}{z^{1/2}} = \frac{\delta^2(z^4+z) + \delta^2(1+\delta+\delta^2)(z^3+z^2)}{z^4 + \delta^2 z^2 + 1}$	$= Tr_1^m(1/\delta) = 1,$ $\delta \notin \mathbb{F}_4$ if $m \equiv 2 \pmod{4}$	Unknown
$G(z) = \frac{1}{Tr_m^n(v)} [Tr_m^n(v^r)(z+1) + Tr_m^n[(vz+v^{2^m})^r] \times (z + Tr_m^n(v)z^{1/2} + 1)^{1-r}] + z^{1/2}$	m even, $r = \pm \frac{2^m-1}{3}$, $v \in \mathbb{F}_{2^{2m}}$, $v^{2^m+1} = 1$ and $v \neq 1$.	Unknown

In Table 1, $D_{\frac{1}{5}}$ stands for the Dickson polynomial of index $\frac{1}{5}$ where $\frac{1}{5}$ is the inverse of 5 modulo $2^{2^m} - 1$.

3 Explicit Constructions of Classes of Semi-bent Functions from o -polynomials

In this section we show how one can construct several infinite classes of semi-bent functions from o -polynomials. Firstly, it has been showed in [4] that the

o-polynomials provide constructions of semi-bent functions in bivariate representation from the class of bent functions introduced in [3] and the class of partial spreads [7]. For completeness, we include the proof.

Theorem 7. *Let G be an o-polynomial on \mathbb{F}_{2^m} , g be Boolean function on \mathbb{F}_{2^m} such that $g(0) = 0$ and $wt(g) = 2^{m-1}$ (that is, g is balanced on \mathbb{F}_{2^m}). Let $\mu \in \mathbb{F}_{2^m}$. Define over $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ the Boolean function f by:*

$$f(x, y) = Tr_1^m(\mu y + xG(yx^{2^m-2})) + g(yx^{2^m-2}), \quad (x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$$

Then f is semi-bent.

Proof. The function f is the sum of two functions f_1 and f_2 defined as $f_1(x) = Tr_1^m(\mu y + xG(yx^{2^m-2}))$ and $f_2(x) = g(yx^{2^m-2})$. Recall that a m -spread of \mathbb{F}_{2^n} ($n = 2m$) is a collection of pairwise supplementary m -dimensional subspaces of \mathbb{F}_{2^n} whose union equals \mathbb{F}_{2^n} . Denote by \mathcal{S} the m -spread in $\mathbb{F}_{2^n} \approx \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, formed by $\{E_a, E_\infty\}$ where $E_a := \{(x, ax); x \in \mathbb{F}_{2^m}\}$ and $E_\infty := \{(0, y); y \in \mathbb{F}_{2^m}\}$. On one hand, according to [3], f_1 is a bent function such that its restriction to the m -spread \mathcal{S} of $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ is linear since G is an o-polynomial on \mathbb{F}_{2^m} . On the other hand, according to [7], f_2 is a bent function such that its restriction to the m -spread \mathcal{S} is constant since g is balanced vanishing at 0. The result follows from Theorem 1 in [4]. \square

Next, we show that the o-polynomials lead to further constructions of semi-bent functions in bivariate representation in the spirit of the well known construction of bent functions due to Maiorana and McFarland [7]. Recall that the Maiorana-McFarland class is the set of all the Boolean functions on $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, of the form $f(x, y) = Tr_1^m(\pi(x)y) + h(x)$ where π is any mapping on \mathbb{F}_{2^m} and h is any Boolean function on \mathbb{F}_{2^m} . That construction has been studied because one can construct bent functions this way by choosing for π a permutation on \mathbb{F}_{2^m} . The reader notices that using the Maiorana-McFarland method, any permutation leads to the construction of bent functions and any mapping 2-to-1 leads to the construction of semi-bent functions (in even dimension). An important point is that the notion of oval polynomial over \mathbb{F}_{2^m} appears to be suitable to build 2-to-1 mappings on \mathbb{F}_{2^m} . Such a property is intensely used in the paper to construct infinite classes of semi-bent functions.

Theorem 8. *Let α be a primitive element of \mathbb{F}_{2^m} and j a positive integer in the range $[0, 2^m - 2]$. Let G be an o-polynomial on \mathbb{F}_{2^m} and g a Boolean function on \mathbb{F}_{2^m} . Define over $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ a Boolean function f by:*

$$f(x, y) = Tr_1^m(xG(y) + \alpha^j xy) + g(y), \quad (x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}.$$

Then f is semi-bent.

Proof. We have to prove that f is semi-bent, that is, its Walsh transform takes only the values 0, 2^{m+1} and -2^{m+1} . Compute the Walsh transform of f . For

every $(a, b) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, we have:

$$\begin{aligned}
\widehat{\chi}_f(a, b) &= \sum_{x \in \mathbb{F}_{2^m}} \sum_{y \in \mathbb{F}_{2^m}} \chi \left(Tr_1^m(xG(y) + \alpha^j xy) + g(y) + Tr_1^m(ax) + Tr_1^m(by) \right) \\
&= \sum_{y \in \mathbb{F}_{2^m}} \chi \left(g(y) + Tr_1^m(by) \right) \sum_{x \in \mathbb{F}_{2^m}} \chi \left(Tr_1^m(xG(y) + \alpha^j xy) + Tr_1^m(ax) \right) \\
&= \sum_{y \in \mathbb{F}_{2^m}} \chi \left(g(y) + Tr_1^m(by) \right) \sum_{x \in \mathbb{F}_{2^m}} \chi \left(Tr_1^m((G(y) + \alpha^j y + a)x) \right) \\
&= 2^m \sum_{y \in \mathbb{F}_{2^m} \mid G(y) + \alpha^j y = a} \chi \left(g(y) + Tr_1^m(by) \right).
\end{aligned}$$

Now, note that (see [3]) since G is an o-polynomial on \mathbb{F}_{2^m} , then the mapping $y \in \mathbb{F}_{2^m} \mapsto G(y) + \alpha^j y$ is 2-to-1 for every $j \in [0, 2^m - 2]$ that is, $\#\{y \in \mathbb{F}_{2^m} \mid G(y) + \alpha^j y = a\} \in \{0, 2\}$. Therefore,

$$\widehat{\chi}_f(a, b) \in \{0, \pm 2^{m+1}\}$$

which completes the proof. \square

Example 9. Consider the regular hyperoval Γ of projective plane $PG_2(2^m)$ (see Figure 1) corresponding to the Frobenius mapping $t \mapsto t^2$ over \mathbb{F}_{2^m} . The points of the form $\langle (t^2, t, 1) \rangle$ for $t \in \mathbb{F}_{2^m}$ together with the point at infinity $\langle (1, 0, 0) \rangle$ are the points of the conic with equation $XZ = Y^2$; the point $(0, 1, 0)$ is the nucleus. In this example, we treat the simpler case where $m = 2$. The 6 points of the regular hyperoval Γ in $PG_2(4)$ are : $\langle (1, 0, 0) \rangle$, $\langle (0, 1, 0) \rangle$, $\langle (0, 0, 1) \rangle$, $\langle (1, 1, 1) \rangle$, $\langle (\alpha, \alpha^2, 1) \rangle$, $\langle (\alpha^2, \alpha, 1) \rangle$ where α is a root of the irreducible polynomial $P(X) = X^2 + X + 1$ over \mathbb{F}_4 . The diagonal line is the "line at infinity" with equation $Z = 0$. The 4×4 grid represents the 16 points of the affine plane $Z \neq 0$. Notice that not every line in the projective space is shown in the figure 1. The horizontal lines should extend to the point $\langle (1, 0, 0) \rangle$ and the vertical lines should meet in $\langle (0, 1, 0) \rangle$. In addition, there should be 4 additional lines through any of the remaining points at infinity. The affine points have coordinates $\langle (x, y, 1) \rangle$ for (x, y) in \mathbb{F}_4^2 . More precisely, the intersection point of the vertical line through $\langle (x, 0, 1) \rangle$ and the horizontal line through $\langle (0, y, 1) \rangle$ has the coordinates $\langle (x, y, 1) \rangle$. Now, the regular hyperoval in $PG_2(4)$ related to the Frobenius o-polynomial over \mathbb{F}_4 give rise to semi-bent functions g_0, g_1 and g_2 defined in bivariate on $\mathbb{F}_4 \times \mathbb{F}_4$ where $g_j(x, y) = Tr_1^2(xy^2 + \alpha^j xy) + h(y)$ for $j \in \{0, 1, 2\}$ where h be any Boolean function over \mathbb{F}_4 .

Let us now indicate that semi-bent functions obtained by Theorem 8 share a very particular feature.

Proposition 10. *Any semi-bent function of Theorem 8 is the sum of two bent functions in the class of Maiorana-McFarland.*

Proof. Indeed, an o-polynomial being a permutation, the Boolean function $(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m} \mapsto Tr_1^m(xG(y)) + g(y)$ is bent. On the other hand, the

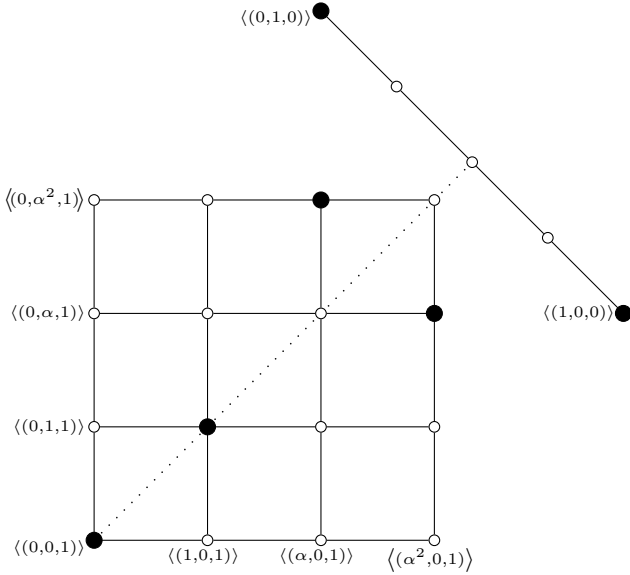


Fig. 1. The regular hyperoval Γ in $PG_2(4)$

Boolean function $(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m} \mapsto Tr_1^m(\alpha^j yx)$ is bent too and lies in the class of Maiorana-McFarland (the map $y \in \mathbb{F}_{2^m} \mapsto \alpha^k y$ is a permutation of \mathbb{F}_{2^m}). It suffices then to note that the addition of the two preceding functions leads to the expression of semi-bent functions stated in Theorem 8.

Therefore, we have obtained semi-bent functions that can be decomposed in the sum of two bent functions. Clearly, if we take at random two bent functions, even in the class of Maiorana-McFarland, their addition would not be probably semi-bent in most cases (the reader should notice that semi-bent functions of Theorem 7 can also be decomposed in the sum of two bent functions). A quite natural question arises then from knowing a similar result can be obtained with other permutations than oval polynomials. Note then that the key point in the proof of Theorem 8 is that $y \mapsto G(y) + \alpha^j y$ is 2-to-1. Now, to our knowledge, we do not know whether other permutations than oval polynomials have that property.

Another fascinating fact is that there is a relation between semi-bent functions of Theorem 7 and Theorem 8. That relation is described in the below Lemma.

Lemma 11. *Let $f_1(x, y) = Tr_1^m(\alpha^j y + xG(yx^{2^m-2})) + g(yx^{2^m-2})$ and $f_2(x, y) = Tr_1^m(xG(y) + \alpha^j xy) + g(y)$. Then $f_1(x, xy) = Tr_1^m(\alpha^j xy + xG(yx^{2^m-1})) + g(yx^{2^m-1}) = f_2(x, y)$, for every $(x, y) \in \mathbb{F}_{2^m}^* \times \mathbb{F}_{2^m}$.*

Let us now focus our attention to another fact. Recall that the set of all o -polynomials is invariant under the transformations (see [3]):

- T1: $G \mapsto G'$ where $G' : z \in \mathbb{F}_{2^m} \mapsto G'(z) := G(\lambda z + \mu)$ with $\lambda \in \mathbb{F}_{2^m}^*$ and $\mu \in \mathbb{F}_{2^m}$;
- T2: $G \mapsto G'$ where $G' : z \in \mathbb{F}_{2^m} \mapsto G'(z) := \lambda G(z) + \mu$ with $\lambda \in \mathbb{F}_{2^m}^*$ and $\mu \in \mathbb{F}_{2^m}$;
- T3: $G \mapsto G'$ where $G' : z \in \mathbb{F}_{2^m} \mapsto G'(z) := G(z^{2^l})^{2^{m-l}}$ where $l \in \mathbb{N}$.
- T4: $G \mapsto G'$ where $G' : z \in \mathbb{F}_{2^m} \mapsto G'(z) := zG(z^{2^m-2})$ (with $G(0) = 0$);
- T5: $G \mapsto G^{-1}$.

A quite natural question arises from knowing whether there is a relation between two semi-bent functions constructed from o -polynomials linked by one of the relations T1 to T5 defined above. Recall the notion of *extended affine equivalence* (in brief, *EA-equivalence*) between two Boolean functions.

Definition 12. *Two Boolean functions f and f' defined on \mathbb{F}_{2^n} are called extended affine equivalent (EA-equivalent) if $f' = f \circ \phi + \ell$ where the mapping ϕ is an affine automorphism on \mathbb{F}_{2^n} and ℓ is an affine Boolean function (affine functions are those whose algebraic degree is at most 1).*

To this end, we shall begin with considering the transformation T1 and semi-bent functions given by Theorem 8 :

$$\begin{aligned} f'(x, y) &= Tr_1^m((G'(y) + \alpha^j y)x) + g(y) \\ &= Tr_1^m((G(\lambda y + \mu) + \alpha^j y)x) + g(y) \\ &= Tr_1^m((\lambda G(\lambda y + \mu) + \alpha^j(\lambda y + \mu))(\lambda^{-1}x)) \\ &\quad + Tr_1^m(\mu \lambda^{-1} \alpha^j x) + g(\lambda^{-1}(\lambda y + \mu) + \mu \lambda^{-1}). \end{aligned}$$

Set $f(x, y) = Tr_1^m((\lambda G(y) + \alpha^j y)x) + g(\lambda^{-1}y + \mu \lambda^{-1})$. Then, f' is EA-equivalent to f .

Considering the second transformation T2 :

$$\begin{aligned} f'(x, y) &= Tr_1^m((G'(y) + \alpha^j y)x) + g(y) = Tr_1^m((\lambda G(y) + \mu + \alpha^j y)x) + g(y) \\ &= Tr_1^m((G(y) + \alpha^j \lambda^{-1} y)(\lambda x)) + Tr_1^m(\mu x) + g(y). \end{aligned}$$

Hence f' is EA-equivalent to f defined as $f(x, y) = Tr_1^m((G(y) + \alpha^j \lambda^{-1} y)x) + g(y)$.

For the third transformation T3 :

$$\begin{aligned} f'(x, y) &= Tr_1^m((G'(y) + \alpha^j y)x) + g(y) = Tr_1^m((G^{2^{m-l}}(y^{2^l}) + \alpha^j y)x) + g(y) \\ &= Tr_1^m((G(y^{2^l}) + \alpha^{2^l j} y^{2^l})x^{2^l}) + g(y). \end{aligned}$$

Hence, f' is EA-equivalent to f defined as $f(x, y) = Tr_1^m((G(y) + \alpha^{2^l j} y)x) + g(y^{2^{m-l}})$.

Therefore, the three transformations T1, T2 and T3 lead to EA-equivalent Boolean semi-bent functions covered by Theorem 8. Let us now investigate the

two last transformations T4 and T5. If G' and G are linked by transformation T4 then

$$\begin{aligned} f'(x, y) &= Tr_1^m((G'(y) + \alpha^j y)x) + g(y) = Tr_1^m((yG(y^{2^m-2}) + \alpha^j y)x) + g(y) \\ &= Tr_1^m((G(y^{2^m-2}) + \alpha^j)xy) + g(y). \end{aligned}$$

On the other hand, for the fifth transformation T5, one has

$$\begin{aligned} f'(x, y) &= Tr_1^m((G'(y) + \alpha^j y)x) + g(y) = Tr_1^m((G^{-1}(y) + \alpha^j y)x) + g(y) \\ &= Tr_1^m((G^{-1}(y) + \alpha^j G(G^{-1}(y)))x) + g(y) = f(x, G^{-1}(y)) \end{aligned}$$

with

$$f(x, y) = Tr_1^m((\alpha^j G(y) + y)x) + g(G(y)).$$

Thus, the two transformations T4 and T5 can lead to potential new semi-bent functions that are not EA-equivalent to semi-bent functions covered by Theorem 8.

We next provide another construction of semi-bent function in bivariate representation.

Theorem 13. *Let m be a positive integer. Assume $m = 2m_1 + 1$ odd. Let G be an o -polynomial on \mathbb{F}_{2^m} and g be a Boolean function on \mathbb{F}_{2^m} . Define a Boolean function f in bivariate representation as:*

$$\begin{aligned} f(x, y) &= Tr_1^m\left(xG^{2^{m_1+1}+1}(y) + xyG^{2^{m_1+1}}(y) + xG^3(y) + xyG^2(y)\right) \\ &\quad + Tr_1^m\left((xy^{2^{m_1+1}} + xy^2 + x)G(y) + xy^{2^{m_1+1}+1} + xy + xy^3\right) \\ &\quad + g(y), (x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}. \end{aligned}$$

Then f is semi-bent on $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$.

Proof. Let $(a, b) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$. A routine computation of the Walsh transform of f at (a, b) gives:

$$\begin{aligned} \widehat{\chi}_f(a, b) &= \sum_{(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}} \chi\left(Tr_1^m\left[xG^{2^{m_1+1}+1}(y) + xyG^{2^{m_1+1}}(y) + G^3(y) \right. \right. \\ &\quad \left. \left. + xyG^2(y) + (xy^{2^{m_1+1}} + xy^2 + x)G(y) + xy^{2^{m_1+1}+1} + xy + xy^3\right] \right. \\ &\quad \left. + g(y) + Tr_1^m(ax + by)\right) \\ &= \sum_{x \in \mathbb{F}_{2^m}} \sum_{y \in \mathbb{F}_{2^m}} \chi\left(Tr_1^m\left[(G^{2^{m_1+1}+1}(y) + yG^{2^{m_1+1}}(y) \right. \right. \\ &\quad \left. \left. + G^3(y) + yG^2(y) + (y^{2^{m_1+1}} + y^2 + 1)G(y) + y^{2^{m_1+1}+1} + y^3 + y + a)\right]\right) \chi(g(y) \\ &\quad + Tr_1^m(by)) \end{aligned}$$

$$\begin{aligned}
&= \sum_{y \in \mathbb{F}_{2^m}} \chi(g(y) + Tr_1^m(by)) \sum_{x \in \mathbb{F}_{2^m}} \chi\left(Tr_1^m\left[\left(G^{2^{m_1+1}+1}(y) + yG^{2^{m_1+1}}(y)\right.\right.\right. \\
&\quad \left.\left.\left.+ G^3(y) + yG^2(y) + (y^{2^{m_1+1}} + y^2 + 1)G(y) + y^{2^{m_1+1}+1} + y + y^3 + a\right)x\right]\right) \\
&= 2^m \sum_{y \in \mathbb{F}_{2^m} | \Psi(y)=a} \chi(g(y) + Tr_1^m(by))
\end{aligned}$$

where $\Psi : \mathbb{F}_{2^m} \mapsto \mathbb{F}_{2^m}$ defined as

$$\begin{aligned}
\Psi(y) &:= G^{2^{m_1+1}+1}(y) + yG^{2^{m_1+1}}(y) + G^3(y) + yG^2(y) \\
&\quad + (y^{2^{m_1+1}} + y^2 + 1)G(y) + y^{2^{m_1+1}+1} + y^3 + y.
\end{aligned}$$

Note then that $\Psi = \Phi \circ \phi$ where Φ and ϕ are two mappings on \mathbb{F}_{2^m} defined as: $\forall y \in \mathbb{F}_{2^m}, \Phi(y) := y^{2^{m_1+1}+1} + y^3 + y$ and $\phi(y) := G(y) + y$. According to Dobbertin [13], Φ is a permutation polynomial on \mathbb{F}_{2^m} . Moreover, since G is an o-polynomial on \mathbb{F}_{2^m} then, the equation $\phi(y) = a$ admits 0 or 2 solutions in \mathbb{F}_{2^m} , that is, the polynomial $G(y) + y$ defines a 2-to-1 mapping on \mathbb{F}_{2^m} . Consequently, $\#\{y \in \mathbb{F}_{2^m} | \Psi(y) = a\} \in \{0, 2\}$. Therefore, $\widehat{\chi}_f(a, b) \in \{0, \pm 2^{m+1}\}$. This completes the proof. \square

Theorem 8 and Theorem 13 can be generalized. Indeed, other semi-bent functions of the form $Tr_1^m(x(\pi_1(y) + \pi_2(y))) + g(y)$ can be obtained from o-polynomials.

Theorem 14. *Let π_1 and π_2 be two permutations of \mathbb{F}_{2^m} whose composition $\pi_1 \circ \pi_2^{-1}$ is an o-polynomial on \mathbb{F}_{2^m} . Let g be a Boolean function over \mathbb{F}_{2^m} . Let f be the Boolean function defined on $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ by*

$$(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}, \quad f(x, y) = Tr_1^m(x(\pi_1(y) + \pi_2(y))) + g(y).$$

Then f is semi-bent.

Proof. One can repeat the first calculation at the beginning of the proof of Theorem 8 and get that

$$\widehat{\chi}_f(a, b) = 2^m \sum_{y \in \mathbb{F}_{2^m} | \pi_1(y) + \pi_2(y) = a} \chi(g(y) + Tr_1^m(by)).$$

Now let us note that

$$\pi_1(y) + \pi_2(y) = a \iff \pi_1 \circ \pi_2^{-1}(\pi_2(y)) + \pi_2(y) = a.$$

We have supposed that $\pi_1 \circ \pi_2^{-1}$ is an oval polynomial. Therefore, the equation $\pi_1 \circ \pi_2^{-1}(z) + z = a$ has zero or two solutions for every $a \in \mathbb{F}_{2^m}$. The map π_2 being a permutation, the equation $\pi_1 \circ \pi_2^{-1}(\pi_2(y)) + \pi_2(y) = a$ has the same number of solutions as $\pi_1 \circ \pi_2^{-1}(z) + z = a$ for every $a \in \mathbb{F}_{2^m}$. That implies that $\widehat{\chi}_f(a, b) \in \{0, \pm 2^{m+1}\}$ for every $(a, b) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ proving that f is semi-bent. \square

If we set $G = \pi_1 \circ \pi_2^{-1}$, then the Boolean function f defined in Theorem 14 can be rewritten as $f(x, y) = f'(x, \pi_2(y))$ where $f'(x, y) = Tr_1^m(xG(y) + xy) + g'(y)$ where $g' = g \circ \pi_2^{-1}$. Now f' is a semi-bent function according to Theorem 8. Therefore, semi-bent functions of Theorem 14 are obtained from those of Theorem 8 by applying a permutation, not necessarily linear, to the second coordinate. Clearly, that feature strongly relies on the fact that G is an oval polynomial. Indeed, semi-bentness property is generally lost if we apply a nonlinear transformation to the coordinates of a semi-bent function.

As an application of the previous theorem, we provide below another construction of semi-bent functions.

Theorem 15. *Let m be an odd positive integer. Define the Boolean function f on $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ as*

$$(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}, \quad f(x, y) = Tr_1^m((y^6 + y^5 + y^3 + y)x) + g(y)$$

where g is any Boolean function over \mathbb{F}_{2^m} . Then f is semi-bent.

Proof. Set $\pi_1(y) = y^6$ and $\pi_2(y) = y^5 + y^3 + y = D_5(y)$ where D_5 stands for the fifth Dickson polynomial. Recall that 5 and $2^{2m} - 1$ are coprime when m is odd. That implies that D_5 is a permutation polynomial of \mathbb{F}_{2^m} . On the other hand, $2^m - 1$ is coprime with 6 when m is odd yielding that $y \mapsto y^6$ is a permutation of \mathbb{F}_{2^m} . Next note that $\pi_1 \circ \pi_2^{-1}(y) = (D_{1/5}(y))^6$ which is an o-polynomial on \mathbb{F}_{2^m} (see Table 1). Therefore, according to Theorem 14, f is semi-bent. \square

Theorem 13 can also be generalized as follows (the proof being a straightforward adaptation of the proof of Theorem 8, we omit it)

Theorem 16. *Let π be a permutation of \mathbb{F}_{2^m} . Let α be a primitive element of \mathbb{F}_{2^m} and j a nonnegative integer. Let G be an o-polynomial and g a Boolean function over \mathbb{F}_{2^m} . Define*

$$\forall (x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}, \quad f(x, y) = Tr_1^m(\pi(G(y) + \alpha^j y)x) + g(y).$$

Then f is semi-bent.

Another infinite class of semi-bent functions can be obtained by combining functions (via the secondary construction of bent functions given in [2]) from the semi-bent functions provided by Theorem 16 and the bent functions from the o-polynomials ([3]).

Theorem 17. *Let G_1, G_2 be two o-polynomial on \mathbb{F}_{2^s} and $\widetilde{G}_1, \widetilde{G}_2$ two o-polynomials on \mathbb{F}_{2^r} . Let h_1 and h_2 be two Boolean functions on \mathbb{F}_{2^s} . Let μ_1 and μ_2 be two elements of \mathbb{F}_{2^r} and α a primitive element of \mathbb{F}_{2^s} . Set $\mathbb{F}_{2^s}^* = \{\alpha^j, 0 \leq j \leq 2^s - 2\}$. We define four Boolean functions f_1, f_2 (both are defined in bivariate representation over $\mathbb{F}_{2^s} \times \mathbb{F}_{2^s}$), g_1 and g_2 (both are defined in bivariate representation over $\mathbb{F}_{2^r} \times \mathbb{F}_{2^r}$) as follows.*

$$f_k(x, y) := Tr_1^s(xG_k(y) + \alpha^j xy) + h_k(y), \quad k \in \{1, 2\};$$

$$g_k(z, t) := Tr_1^r(\mu_k t + z\widetilde{G}_k(tz^{2^r-2})), \quad k \in \{1, 2\}.$$

Set

$$\forall (x, y, z, t) \in \mathbb{F}_{2^s} \times \mathbb{F}_{2^s} \times \mathbb{F}_{2^r} \times \mathbb{F}_{2^r},$$

$$f(x, y, z, t) := f_1(x, y) + g_1(z, t) + (f_1(x, y) + f_2(x, y))(g_1(z, t) + g_2(z, t)).$$

Then f is semi-bent.

Proof. Compute the Walsh transform of f . For every $(a, b, c, d) \in \mathbb{F}_{2^s} \times \mathbb{F}_{2^s} \times \mathbb{F}_{2^r} \times \mathbb{F}_{2^r}$, we have:

$$\begin{aligned} \widehat{\chi}_f(a, b, c, d) &= \sum_{(x, y, z, t) \in \mathbb{F}_{2^s}^2 \times \mathbb{F}_{2^r}^2} \chi\left(f_1(x, y) + g_1(z, t) \right. \\ &\quad \left. + (f_1(x, y) + f_2(x, y))(g_1(z, t) + g_2(z, t)) \right) \\ &\quad + Tr_1^s(ax) + Tr_1^s(by) + Tr_1^r(cz) + Tr_1^r(dt). \end{aligned}$$

Now, one can split the sum depending on the value of $g_1(z, t) + g_2(z, t)$:

$$\begin{aligned} \widehat{\chi}_f(a, b, c, d) &= \sum_{(x, y) \in \mathbb{F}_{2^s}^2} \sum_{(z, t) \in \mathbb{F}_{2^r}^2 | g_1(z, t) + g_2(z, t) = 1} \chi\left(f_2(x, y) + g_1(z, t) \right. \\ &\quad \left. + Tr_1^s(ax + by) + Tr_1^r(cz + dt) \right) \\ &\quad + \sum_{(x, y) \in \mathbb{F}_{2^s}^2} \sum_{(z, t) \in \mathbb{F}_{2^r}^2 | g_1(z, t) + g_2(z, t) = 0} \chi\left(f_1(x, y) + g_1(z, t) + \right. \\ &\quad \left. Tr_1^s(ax + by) + Tr_1^r(cz + dt) \right). \end{aligned}$$

Given a set E , denote by δ_E the indicator of E ($\delta_E(u)$ equals 1 if $u \in E$ and 0 otherwise). Note that we have $\delta_{\{(z, t) \in \mathbb{F}_{2^r}^2 | g_1(z, t) + g_2(z, t) = 1\}} = \frac{1 - \chi(g_1(z, t) + g_2(z, t))}{2}$.

Similarly, one has $\delta_{\{(z, t) \in \mathbb{F}_{2^r}^2 | g_1(z, t) + g_2(z, t) = 0\}} = \frac{1 + \chi(g_1(z, t) + g_2(z, t))}{2}$.

Hence,

$$\begin{aligned} \widehat{\chi}_f(a, b, c, d) &= \sum_{(x, y) \in \mathbb{F}_{2^s}^2} \chi\left(f_2(x, y) + Tr_1^s(ax) + Tr_1^s(by) \right) \\ &\quad \sum_{(z, t) \in \mathbb{F}_{2^r}^2} \chi\left(g_1(z, t) + Tr_1^r(cz) + Tr_1^r(dt) \right) \left(\frac{1 - \chi(g_1(z, t) + g_2(z, t))}{2} \right) \\ &\quad + \sum_{(x, y) \in \mathbb{F}_{2^s}^2} \chi\left(f_1(x, y) + Tr_1^s(ax) + Tr_1^s(by) \right) \\ &\quad \sum_{(z, t) \in \mathbb{F}_{2^r}^2} \chi\left(g_1(z, t) + Tr_1^r(cz) + Tr_1^r(dt) \right) \left(\frac{1 + \chi(g_1(z, t) + g_2(z, t))}{2} \right). \end{aligned}$$

After expanding, this gives

$$\begin{aligned} \widehat{\chi}_f(a, b, c, d) &= \widehat{\chi}_{f_2}(a, b) \sum_{(z,t) \in \mathbb{F}_{2^r}^2} \left(\frac{1}{2} \chi \left(g_1(z, t) + Tr_1^r(cz) + Tr_1^r(dt) \right) \right. \\ &\quad \left. - \frac{1}{2} \chi \left(g_1(z, t) + Tr_1^r(cz) + Tr_1^r(dt) + g_1(z, t) + g_2(z, t) \right) \right) \\ &\quad + \widehat{\chi}_{f_1}(a, b) \sum_{(z,t) \in \mathbb{F}_{2^r}^2} \left(\frac{1}{2} \chi \left(g_1(z, t) + Tr_1^r(cz) + Tr_1^r(dt) \right) \right. \\ &\quad \left. + \frac{1}{2} \chi \left(g_1(z, t) + Tr_1^r(cz) + Tr_1^r(dt) + g_1(z, t) + g_2(z, t) \right) \right). \end{aligned}$$

Therefore,

$$\widehat{\chi}_f(a, b, c, d) = \widehat{\chi}_{f_2}(a, b) \delta(c, d) + \widehat{\chi}_{f_1}(a, b) \gamma(c, d)$$

by setting $\delta(c, d) := \frac{\widehat{\chi}_{g_1}(c, d) - \widehat{\chi}_{g_2}(c, d)}{2}$ and $\gamma(c, d) := \frac{\widehat{\chi}_{g_1}(c, d) + \widehat{\chi}_{g_2}(c, d)}{2}$.

According to [3], since \widetilde{G}_1 (resp. \widetilde{G}_2) is an o-polynomial, the function g_1 (resp. g_2) is bent, that is, $\widehat{\chi}_{g_1}(c, d) \in \{\pm 2^r\}$ (resp. $\widehat{\chi}_{g_2}(c, d) \in \{\pm 2^r\}$) for every $(c, d) \in \mathbb{F}_{2^r}^2$. Therefore, $\delta(c, d)\gamma(c, d) = 0$. Thus only the two following possibilities can occur:

$$\delta(c, d) = 0 \text{ and } \gamma(c, d) = \pm 2^r$$

or

$$\delta(c, d) = \pm 2^r \text{ and } \gamma(c, d) = 0.$$

Now, according to Theorem 8, f_1 and f_2 being semi-bent. Hence $\widehat{\chi}_{f_1}(a, b) \in \{0, \pm 2^{s+1}\}$ and $\widehat{\chi}_{f_2}(a, b) \in \{0, \pm 2^{s+1}\}$. Therefore $\widehat{\chi}_f(a, b, c, d) \in \{0, \pm 2^{s+r+1}\}$ proving that f is semi-bent. \square

4 Conclusion

In this paper we show how finite geometry can contribute to construct cryptographic Boolean functions. In [3], it has been shown how to construct bent functions from oval polynomials. That result has revealed a link between oval polynomials, that are important objects in finite geometry, and Niho bent functions. Other connections between bent vectorial functions and oval polynomials have been highlighted in [14]. In this paper, we push further the link initiated in [3] and obtain several new infinite families of semi-bent functions from oval polynomials.

Acknowledgement. The author would like to thank anonymous referees and Pascale Charpin for valuable suggestions that considerably improve the paper.

References

1. Canteaut, A., Carlet, C., Charpin, P., Fontaine, C.: On cryptographic properties of the cosets of $R(1,m)$. *IEEE Transactions on Information Theory* 47, 1494–1513 (2001)
2. Carlet, C.: On the secondary constructions of resilient and bent functions. In: *Proceedings of the Workshop on Coding, Cryptography and Combinatorics 2003*, pp. 3–28. Birkhäuser Verlag (2004)
3. Carlet, C., Mesnager, S.: On Dillon’s class H of bent functions, niho bent functions and O-polynomials. *Journal of Combinatorial Theory, Series A* 118(8), 2392–2410 (2011)
4. Carlet, C., Mesnager, S.: On Semi-bent Boolean Functions. *IEEE Transactions on Information Theory-IT* 58(5), 3287–3292 (2012)
5. Charpin, P., Pasalic, E., Tavernier, C.: On bent and semi-bent quadratic Boolean functions. *IEEE Transactions on Information Theory* 51(12), 4286–4298 (2005)
6. Chee, S., Lee, S., Kim, K.: Semi-bent Functions. In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) *ASIACRYPT 1994*. LNCS, vol. 917, pp. 107–118. Springer, Heidelberg (1995)
7. Dillon, J.: Elementary Hadamard difference sets. In PhD dissertation, University of Maryland
8. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseht, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
9. Meier, W., Staffelbach, O.: Fast correlation attacks on stream ciphers. In: Günther, C.G. (ed.) *EUROCRYPT 1988*. LNCS, vol. 330, pp. 301–314. Springer, Heidelberg (1988)
10. Mesnager, S.: Semi-bent functions from Dillon and Niho exponents, Kloosterman sums and Dickson polynomials. *IEEE Transactions on Information Theory-IT* 57(11), 7443–7458 (2011)
11. Rothaus, O.S.: On “bent” functions. *J. Combin. Theory Ser. A* 20, 300–305 (1976)
12. Klein, A., Storme, L.: Applications of finite geometry in coding theory and cryptography. In: *Information Security, Coding Theory and Related Combinatorics*, pp. 38–58 (2011)
13. Dobbertin, H.: Almost Perfect Nonlinear power functions on $GF(2n)$: The Welch case. *IEEE Transactions on Information Theory* 45(4), 1271–1275 (1999)
14. Mesnager, S.: Bent vectorial functions and linear codes from o-polynomials (preprint, 2013)

Efficient Generation of Elementary Sequences

David Gardner¹, Ana Sălăgean¹, and Raphael C.-W. Phan²

¹ Department of Computer Science, Loughborough University, UK
{D.Gardner2,A.M.Salagean}@lboro.ac.uk

² Faculty of Engineering, Multimedia University, Malaysia
raphael@mmu.edu.my

Abstract. Given an irreducible non-primitive polynomial g of degree n over $\mathbb{F}_2[x]$ we aim to compute in parallel all the elementary sequences with minimal polynomial g (i.e. one sequence from each class of equivalence under cyclic shifts). Moreover, they need to each be in a suitable phase such that interleaving them will produce an m-sequence with linear complexity $\deg(g)$; this m-sequence is therefore produced at the rate of $q = (2^n - 1)/\text{ord}(g)$ bits per clock cycle. A naive method would use q LFSRs so our aim is to use considerably fewer. We explore two approaches: running a small number of Galois LFSRs with suitable seeds and using certain registers, possibly with a small amount of buffering; alternatively using only one (Galois or Fibonacci) LFSR and computing certain linear combinations of its registers. We ran experiments for all irreducible polynomials of degree n up to 14 and for each n we found that efficient methods exist for at least one m-sequence. A combination of the two approaches above is also described.

Keywords: Fibonacci LFSR, Galois LFSR, m-sequences, interleaving, elementary sequences.

1 Introduction

The linear feedback shift register (LFSR) has become a standard way to generate linearly recurrent sequences and in particular m-sequences, i.e. sequences which have the maximum period given the length of the LFSR. m-sequences are commonly used as good approximations of random binary sequences, so called pseudo-random or pseudo-noise sequences. Sequences with such properties are invaluable in symmetric key cryptographic systems implemented both in hardware and software. Hence rapid generation of m-sequences without excessive memory requirements is desirable. There are many other areas of applications for m-sequences.

Some authors have explored various ways to artificially increase the rate of output of such cryptographic systems; Robshaw[6] described a method of interleaving a number of phases of the desired m-sequence, all these phases being generated synchronously by separate LFSRs (see also [3]). Blackburn[1] then extended this idea by using a smaller number of Galois LFSR in order to produce the required synchronous sequences exploiting the fact that a Galois LFSR with

a primitive feedback polynomial can produce a greater spread of phases of a single m-sequence than an equivalent Fibonacci LFSR. The main difficulty in [1] lies in identifying a primitive feedback polynomial that will produce sequences in the required phases.

Surböch and Weinrichter[8] explored interleaving “elementary sequences” instead of m-sequences. In the present paper we are further developing this approach. An elementary sequence is one whose minimal polynomial g is irreducible but not necessarily primitive. For a fixed non-primitive g , there are several such sequences, not all of which are cyclic shifts of one another. More precisely, with equivalence given by cyclic shifts, there are exactly $q = (2^n - 1)/\text{ord}(g)$ classes, each having $\text{ord}(g)$ elements.

Given an m-sequence s of length $2^n - 1$ it is known that for any proper factor q of $2^n - 1$, the (improper) decimations of s by q are elementary sequences, all having the same irreducible minimal polynomial g .

Taking the reverse approach, we can interleave q elementary sequences in order to obtain an m-sequence. Not every arbitrary collection of q elementary sequences with the same irreducible minimal polynomial will produce an m-sequence by interleaving. The sequences need to satisfy further conditions: to be inequivalent under cyclic shifts, and to be in a certain phase, see Theorem 4. Our aim is to obtain in an efficient way exactly such a collection of sequences.

Our general approach is to use a certain number (less than q) of Galois or Fibonacci LFSRs, and then extract from different registers the sequences we need, by possibly using additional buffering or XORs. We look then at two particular cases of this approach. In the first case, we use a small number of Galois LFSRs and possibly some small amount of buffering, but no additional XORs. We exploit the fact that, unlike a Fibonacci LFSR, each register of a Galois LFSR can produce elementary sequences from different equivalence classes. In the second particular case we consider, we only run a single (Galois or Fibonacci) LFSR and obtain all the required sequences by doing further XOR operations between different registers. We exploit the fact that the n sequences obtained from the n registers form a basis in the vector space of all elementary sequences with fixed minimal polynomial g . We ran experiments for all irreducible polynomials of degree n up to 14, where $2^n - 1$ is not prime.

While our construction of elementary sequences was targeted at fast generation of m-sequences, there are other possible applications of this construction. In coding theory, elementary sequences are the codewords of minimal cyclic codes (also known as irreducible cyclic codes), hence our construction will produce the non-equivalent codewords of such a code. In cryptography, one could use the elementary sequences as inputs to a non-linear function in order to construct a filtering generator for a stream cipher.

2 Preliminaries

Throughout the paper all the fields are finite fields. We define linear recurring sequences as usual and note elementary sequences and m-sequences as special cases:

Definition 1. An infinite sequence $s = s_0, s_1, \dots$ with elements in a field K is called a homogeneous linear recurring sequence if there exists a homogeneous linear recurrence relation of the form $s_{i+n} = c_{n-1}s_{i+n-1} + \dots + c_1s_{i+1} + c_0s_i$ for all $i = 0, 1, \dots$, where $c_0, c_1, \dots, c_{n-1} \in K$ are constants. We associate to it a characteristic polynomial $f(x) = x^n + c_{n-1}x^{n-1} + \dots + c_1x + c_0$. If n is minimal for the given sequence we call n the linear complexity of s and f the minimal polynomial of s .

An elementary sequence is a sequence that has an irreducible minimal polynomial. If the minimal polynomial is moreover primitive, the the sequence is called m-sequence.

Recall that an m-sequence with linear complexity n has period $|K|^n - 1$, so binary m-sequences have period $2^n - 1$. More generally, a sequence with irreducible minimal polynomial f has period equal to $\text{ord}(f)$, where $\text{ord}(f)$ denotes the order of f , i.e. the minimum integer $k > 0$ such that $f|x^k - 1$. We have that $\text{ord}(f)$ is a factor of $|K|^{\deg(f)} - 1$ with $\text{ord}(f) = |K|^{\deg(f)} - 1$ achieved iff f is primitive.

Decimation and its inverse, interleaving, will play an important role in our constructions:

Definition 2. Given a sequence $s = s_0, s_1, \dots$, its q -decimation starting at position j is the sequence u such that $u_i = s_{j+iq}$. If j is not specified, it is by default $j = 0$.

Note that if s has period N then any q -decimation of s has (not necessarily minimal) period $N/\text{gcd}(N, q)$.

Definition 3. The interleaving of q sequences $u^{(0)}, u^{(1)}, \dots, u^{(q-1)}$ is the sequence s defined as $s_{j+iq} = u_i^{(j)}$.

Note that if all the sequences $u^{(0)}, u^{(1)}, \dots, u^{(q-1)}$ have (not necessarily minimal) period d , then their interleaving will have (not necessarily minimal) period dq .

For the rest of this paper we will restrict to binary sequences. Some of these constructions could be applicable to other fields, but this will be a subject for further research. We will denote by \mathbb{F}_{2^n} the finite field with 2^n elements.

Any linear recurrent sequence can be represented using the trace transformation. We are interested in elementary sequences:

Theorem 1. [4, Theorem 6.24] Let s_0, s_1, \dots be a linear recurring sequence in \mathbb{F}_2 whose characteristic polynomial g is irreducible over \mathbb{F}_2 and has degree n . Let β be a root of g in the extension field \mathbb{F}_{2^n} . Then there exists a uniquely determined $a \in \mathbb{F}_{2^n}$ such that $s_i = \text{Tr}(a\beta^i) = \sum_{k=0}^{n-1} a^{2^k} (\beta^{2^k})^i$, $i = 0, 1, \dots$

We introduce the following short hand notation to refer to elementary sequences in terms of their trace representation:

Definition 4. We define $\text{Seq}_\beta(a)$ as the sequence s whose i -th element is represented by $s_i = \text{Tr}(a\beta^i)$.

We note that Seq is linear: for any $a_1, a_2 \in \mathbb{F}_{2^n}$ and $c_1, c_2 \in \mathbb{F}_2$ we have

$$c_1 \text{Seq}_\beta(a_1) + c_2 \text{Seq}_\beta(a_2) = \text{Seq}_\beta(c_1 a_1 + c_2 a_2)$$

For elements β, x in a finite field we will denote by $\log_\beta(x)$ the discrete logarithm of x in the base β , i.e. the smallest positive integer i with the property $\beta^i = x$, if such an integer exists. When β is not a primitive element of the fields we must remember that \log_β will not always be defined.

Given $s = s_0, s_1, \dots$ we denote by $(s \ll k)$ the sequence obtained by shifting s by k positions to the left, i.e. the sequence s_k, s_{k+1}, \dots . If s is periodic with period N we denote by $s \gg k$ the sequence obtained by cyclicly shifting s by k positions to the right, i.e. $s_{N-k}, s_{N-k+1}, \dots, s_{N-1}, s_0, s_1$. Note that $(s \gg k) = (s \ll (N - k))$.

If two sequences of period N are such that one can be obtained from the other by a (cyclic) shift, then we say the two sequences are equivalent (under cyclic shifts). The different cyclic shifts of a sequence are sometimes called “phases”, especially in engineering contexts.

Shifting relates to the $\text{Seq}()$ notation as follows:

Lemma 1. *Let $b, b_1, b_2 \in \mathbb{F}_{2^n}^*$, and $h \in \mathbb{Z}$.*

$$\begin{aligned} (\text{Seq}_\beta(b) \gg h) &= \text{Seq}_\beta(b\beta^{-h}) \\ (\text{Seq}_\beta(b) \ll h) &= \text{Seq}_\beta(b\beta^h) \\ (\text{Seq}_\beta(b_1) \gg h) &= \text{Seq}_\beta(b_2) \Leftrightarrow h = \log_\beta(b_1 b_2^{-1}). \end{aligned}$$

Proof. The proof is similar to [7, Lemma 1], as it does not depend on whether the minimal polynomial of β is primitive or not.

3 Fibonacci and Galois LFSRs

We recall the definition of both the Fibonacci and Galois Linear Feedback Shift Registers (LFSRs) and establish the notations that we will use later. As there are some variations in the literature, we will define everything explicitly.

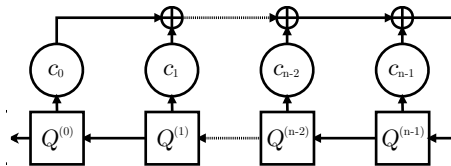


Fig. 1. Fibonacci LFSR

A Fibonacci LFSR is depicted in Fig. 1. We will denote the n registers $Q^{(0)}, Q^{(1)}, \dots, Q^{(n-1)}$. At time $t = 0, 1, \dots$ the content of register $Q^{(j)}$ will be

denoted $q_t^{(j)}$, so the contents of the register $Q^{(j)}$ over time forms the sequence $q^{(j)} = q_0^{(j)}, q_1^{(j)}, \dots$. The contents of all the registers at time t , i.e. the n -tuple $(q_t^{(0)}, \dots, q_t^{(n-1)})$, is known as the state of the LFSR at time t . The initial state of the LFSR is the *state* at time 0. The content of the registers will be updated at each clock interval i according to the following

$$q_{i+1}^{(j)} = \begin{cases} c_{n-1}q_i^{(n-1)} + \dots + c_1q_i^{(1)} + c_0q_i^{(0)} & \text{if } j = n - 1 \\ q_i^{(j+1)} & \text{otherwise.} \end{cases} \quad (1)$$

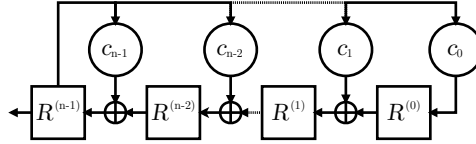


Fig. 2. Galois LFSR

The Galois LFSR, shown in Figure 2, will produce from each register $R^{(j)}$, $j = 0, 1, \dots, n - 1$ a sequence denoted $r^{(j)} = r_0^{(j)}, r_1^{(j)}, \dots$. These values are updated at each clock interval i according to the following

$$r_{i+1}^{(j)} = \begin{cases} c_0r_i^{(n-1)} & \text{if } j = 0 \\ r_i^{(j-1)} + c_jr_i^{(n-1)} & \text{otherwise.} \end{cases} \quad (2)$$

There are two related polynomials associated to the (Fibonacci or Galois) LFSR, namely the *feedback polynomial* $c_0x^n + c_1x^{n-1} + \dots + c_{n-1}x + 1$ and the *characteristic polynomial* $f(x) = x^n + c_{n-1}x^{n-1} + \dots + c_1x + c_0$. The feedback polynomial is the reciprocal of the characteristic polynomial, so if one of these polynomials is irreducible/primitive, so is the other.

Both the Fibonacci and the Galois LFSR produce as their output $q^{(0)}$, respectively $r^{(n-1)}$ a sequence with characteristic polynomial f . The initial states are in a one-to-one correspondence with the sequences with characteristic polynomial f .

Given the initial state of a Fibonacci LFSR, a Galois LFSR that produces exactly the same output sequence will need to have initial state given by

$$r_0^{(j)} = q_0^{(n-1-j)} + c_{n-1}q_0^{(n-1-j-1)} + \dots + c_{j+1}q_0^{(0)}$$

for $j = 0, 1, \dots, n - 1$.

Conversely, given the initial state of a Galois LFSR, a Fibonacci LFSR that produces exactly the same output sequence will need to have initial state

$$q_0^{(n-1-j)} = r_0^{(j)} + c_{n-1}q_0^{(n-1-j-1)} + \dots + c_{j+1}q_0^{(0)}$$

for $j = n - 1, n - 2, \dots, 0$.

4 Efficient Generation of Elementary Sequences for Interleaving

We first need to recall the number of (inequivalent) elementary sequences and their connection to m -sequences. The following two theorems are known results, but we make them more precise, as needed for our purposes.

Theorem 2. *Given an irreducible polynomial g of degree n and order d , there are exactly $2^n - 1$ distinct non-zero elementary sequences with minimal polynomial g . There are $q = (2^n - 1)/d$ classes of equivalence (under cyclic shifts), each having d elements.*

If β is a root of g and α is a primitive element of \mathbb{F}_{2^n} such that $\beta = \alpha^q$ then each of the classes above is of the form $\{\text{Seq}_\beta(\alpha^{i+jq}) \mid j = 0, 1, \dots, d-1\}$ for $i = 0, 1, \dots, q-1$.

Proof. The sequences can be written as $\text{Seq}_\beta(a)$ with each of the $2^n - 1$ elements $a \in \mathbb{F}_{2^n}^*$ uniquely identifying a sequence. By Lemma 1, two sequences $\text{Seq}_\beta(b_1)$ and $\text{Seq}_\beta(b_2)$ are equivalent under cyclic shifts iff there is an integer h such that $b_1 = b_2\beta^h$.

Theorem 3. *(cf. [8] and [5, Theorem 11, Ch. 8, §4]) Let $s = s_0, s_1, \dots$ be an m -sequence with composite period $2^n - 1 = dq$. Let α be a root of the minimal polynomial of s and let $a \in \mathbb{F}_{2^n}$ be such that $s = \text{Seq}_\alpha(a)$. Then the q -decimation of s (starting at positions $0, 1, \dots, q-1$) will result in the q sequences $s^{(0)}, \dots, s^{(q-1)}$ such that $s^{(j)} = \text{Seq}_\beta(a\alpha^j)$ for $j = 0, 1, \dots, q-1$ and $\beta = \alpha^q$.*

Proof. The i -th element of $s^{(j)}$ is $s_i^{(j)} = s_{j+iq} = \text{Tr}(a\alpha^{j+iq}) = \text{Tr}(a\alpha^j(\alpha^q)^i) = \text{Tr}(a\alpha^j(\beta)^i)$, which is exactly the i -th element of the sequence $\text{Seq}_\beta(a\alpha^j)$.

The result above and its proof immediately lead to the following generalisation:

Theorem 4. *Let s be an m -sequence of linear complexity n and minimal polynomial f . Assume $2^n - 1 = dq$ is a non-trivial factorisation. Let g be an irreducible polynomial of order d , degree n and let β be a root of g . Let $u^{(0)}, \dots, u^{(q-1)}$ be elementary sequences with minimal polynomial g .*

We have the following equivalence: s can be obtained by interleaving $u^{(0)}, \dots, u^{(q-1)}$ if and only if there is a primitive root α of f such that $\beta = \alpha^q$, and there is an $a \in \mathbb{F}_{2^n}$ such that $s = \text{Seq}_\alpha(a)$ and $u^{(j)} = \text{Seq}_\beta(a\alpha^j)$ for $j = 0, 1, \dots, q-1$.

Note that if we start from an irreducible non-primitive polynomial g of degree n and order d and construct the finite field \mathbb{F}_{2^n} as the algebraic extension by β , a root of g , then there are several primitive elements $\alpha \in \mathbb{F}_{2^n}$ which are solutions for $\beta = \alpha^q$ (with $q = (2^n - 1)/d$). Moreover these α need not all have the same minimal polynomial.

Our goal is, given g , to produce one elementary sequence from each of the q equivalence classes; moreover these sequences should be in the correct phase relative to each other (as described by Theorem 4) such that they may be interleaved to generate an m -sequence.

A first, straightforward approach would be to generate these sequences using q (Galois or Fibonacci) LFSRs with suitably chosen initial states. Let us examine in more detail how to compute the initial states. Assume we are given n initial terms of the target m -sequence s .

The initial states of the q Fibonacci LFSRs can be obtained by computing a further $(q-1)n$ terms of s (to give us a total of qn terms) and q -decimating the qn terms to obtain the q initial states.

The initial states of the q Galois LFSRs can be obtained by computing first the initial states of the Fibonacci LFSRs, and then transforming them into the equivalent initial states of Galois LFSRs as described at the end of Section 3. An alternative efficient way of computing the Galois initial states is described in [9] and [2].

However, we are interested in obtaining the desired sequences from less than q LFSRs. To this end, we will examine closer what sequences we can obtain from each of the registers of an LFSR, rather than just from the output register.

Theorem 5. *Let $g = x^n + c_{n-1}x^{n-1} + \dots + c_1x + c_0$ be an irreducible non-primitive polynomial with root β .*

Consider a Fibonacci and a Galois LFSR, both with characteristic polynomial g and initial states chosen such that they both produce the same output $\text{Seq}_\beta(a)$ for some given $a \in \mathbb{F}_{2^n}$.

The register $Q^{(j)}$ of the Fibonacci LFSR will produce the sequence $q^{(j)} = \text{Seq}_\beta(a\beta^j) = (\text{Seq}_\beta(a) \ll j)$, for $j = 0, 1, \dots, n-1$.

The register $R^{(j)}$ of the Galois LFSR will produce the sequence $r^{(j)} = \text{Seq}_\beta(av_j)$ where $v_j = c_{j+1} + c_{j+2}\beta + \dots + c_{n-1}\beta^{n-j-2} + \beta^{n-j-1}$, for $j = 0, 1, \dots, n-1$. Moreover, if α is a primitive element of \mathbb{F}_{2^n} such that $\beta = \alpha^q$, where $q = (2^n - 1)/\text{ord}(g)$, then $r^{(j)} = \text{Seq}_\beta(a\alpha^{h_j}) = (\text{Seq}_\beta(a\alpha^{k_j}) \ll l_j)$ where $h_j = \log_\alpha v_j = k_j + l_jq$ and $0 \leq k_j < q$.

Proof. For the Fibonacci LFSR the result is immediate and well known. For the Galois LFSR, (2) can be rewritten as $r^{(0)} = (r^{(n-1)} \gg 1)$ and $r^{(j)} = ((r^{(j-1)} + c_j r^{(n-1)}) \gg 1)$ for $1 \leq j \leq n-1$. By induction, and using Lemma 1 we obtain the required expression, similar to [7, Theorem 2].

This result tells us that for Fibonacci LFSRs all registers contain the same sequence, shifted by $1, 2, \dots, n-1$ positions, so we cannot hope to obtain sequences from different equivalence classes. The situation is more interesting for Galois LFSRs. Here, depending on the characteristic polynomial, we may obtain sequences from several classes. It all depends on the values of $h_j = \log_\alpha v_j$. If h_j is not a multiple of q (or equivalently $v_j \notin \langle \beta \rangle$) then the sequence produced in the register $R^{(j)}$ is inequivalent to (i.e. not a cyclic shift of) the output of the LFSR. Moreover if h_j and h_k are not congruent modulo q then the sequences in registers j and k are inequivalent. So we can hope to obtain several inequivalent

elementary sequences from the same Galois LFSR, and thus obtain from less than q LFSRs all the q elementary sequences needed for interleaving. Of course we also need to examine the relative shifts of the sequences, given by the integer quotients l_j in the Theorem above. Experimental data using this approach is described in Section 4.1.

An alternative approach to obtaining more than one elementary sequence from one LFSR is the following. It is well known that the set of sequences with given minimal polynomial g forms a vector space over \mathbb{F}_2 . Therefore, if we have a basis we can obtain any other sequence as a linear combination.

Theorem 6. *Let $a_0, a_1, \dots, a_{n-1} \in \mathbb{F}_{2^n}$. We have the following equivalence: a_0, a_1, \dots, a_{n-1} is a basis of \mathbb{F}_{2^n} (viewed as an n -dimensional vector space over \mathbb{F}_2) if and only if $\text{Seq}_\beta(a_0), \text{Seq}_\beta(a_1), \dots, \text{Seq}_\beta(a_{n-1})$ is a basis of the set of sequences with minimal polynomial g (viewed as an n -dimensional vector space over \mathbb{F}_2).*

Proof. Let $a \in \mathbb{F}_{2^n}$. a_0, a_1, \dots, a_{n-1} is a basis of \mathbb{F}_{2^n} iff there are unique $c_0, c_1, \dots, c_{n-1} \in \mathbb{F}_2$ such that $a = \sum_{i=0}^{n-1} c_i a_i$ iff there are unique $c_0, c_1, \dots, c_{n-1} \in \mathbb{F}_2$ such that $\text{Seq}_\beta(a) = \text{Seq}_\beta(\sum_{i=0}^{n-1} c_i a_i) = \sum_{i=0}^{n-1} c_i \text{Seq}_\beta(a_i)$ iff $\text{Seq}_\beta(a_0), \text{Seq}_\beta(a_1), \dots, \text{Seq}_\beta(a_{n-1})$ is a basis.

Corollary 1. *With the notations of Theorem 5, each of the sets of sequences $\{q^{(j)} | j = 0, 1, \dots, n-1\}$ and $\{r^{(j)} | j = 0, 1, \dots, n-1\}$ form a basis for the set of sequences with minimal polynomial g (viewed as a n -dimensional vector space over \mathbb{F}_2).*

Hence it turns out that the sequences from the n LFSR registers do form a basis (regardless whether Fibonacci or Galois LFSR) so any other sequence can be obtained as a linear combination of the registers. Examples of this approach are discussed in Section 4.2.

More generally we can combine the two approaches above. Namely, for speeding up the computation of the desired set of sequences $A = \{\text{Seq}_\beta(a\alpha^j) | j = 0, \dots, q-1\}$ we will use the following general method: we employ several (but less than q) Galois or Fibonacci LFSRs. We consider the set B of sequences generated in each register of each of the LFSRs. This set B will contain some of the sequences in A . Any remaining sequences in A can be obtained from the sequences in B by either buffering (if the sequence exists in B but is in the wrong phase) or by computing a linear combination of sequences in B . Experimental results using this combined method are a subject of further research.

4.1 Using Several Galois LFSRs

For given irreducible characteristic polynomials g we examine the Galois LFSR, computing the equivalence class and the relative shift for each register, see Theorem 5. We then identify suitable registers that produce sequences from different classes, preferably in the same phase or with a small difference of phase. We then determine the number of Galois LFSRs we need to produce all the elementary

sequences we need for interleaving. It suffices to run the experiments for only one initial state, as the classes and the shifts are computed relative to the output sequence.

We ran experiments for all irreducible polynomials g of degree n up to 14, where $2^n - 1$ is not prime. For each g we computed all the possible values of α and their minimal polynomial f (which will also be the minimal polynomial of the interleaved sequence). Table 1 shows a few examples of constructions yielding an elementary sequence in the correct phase from each equivalence class. The ‘‘Classes’’ and ‘‘Shifts’’ n -tuples display the values (k_{n-1}, \dots, k_0) and (l_{n-1}, \dots, l_0) , with the notations from Theorem 5. The list ‘‘Interleave’’ specifies which registers we need to interleave, with a triple of the form $(R^{(j)}, a\alpha^i, l_j)$ signifying that we have to use a buffer of l_j terms for register $R^{(j)}$ of a Galois LFSR initialised such that the LFSR output is $\text{Seq}_\beta(a\alpha^i)$. Finally, the LFSRs value indicates the total number of Galois LFSRs we need.

For all lengths in our experiment we were able to determine at least one efficient construction, in the sense that the number of LFSRs needed for the construction is less than q , the total number of sequences required. For each n there is at least one g which will produce 2 of the required sequences, for most n at least 3 sequences can be acquired from one LFSR. If we allow q to increase, such that the elementary sequences become shorter, we generally see that we can obtain elementary sequences from more equivalence classes from a single LFSR, thus lowering the total number of LFSRs needed for the full construction.

4.2 Using One LFSR and Linear Combinations of Its Registers

We wish to produce the required elementary sequences using only one LFSR, either Fibonacci or Galois, but allow ourselves to linearly combine the output of the registers to produce our sequences. This is possible according to Corollary 1. Let a be such that the output of our LFSR is $\text{Seq}_\beta(a)$.

To obtain a particular desired sequence $\text{Seq}_\beta(a\alpha^j)$ for $j = 1, \dots, q - 1$ as a linear combination of the registers it suffices to represent α^j in the basis $1, \beta, \dots, \beta^{n-1}$ for a Fibonacci LFSR, or in basis v_0, v_1, \dots, v_{n-1} for a Galois one. Alternatively we can consider the first n terms of the sequence $\text{Seq}_\beta(a\alpha^j)$, viewed as an element in the vector space \mathbb{F}_2^n and write them in the basis consisting of the first n elements of each of the sequences corresponding to the registers of the LFSR. Any of these approaches will amount to solving an $n \times n$ system of linear equations over \mathbb{F}_2 , which is extremely fast. In our experiments we set up the LFSRs so as to generate the m-sequence in its ‘‘impulse response’’ form, i.e. starting with initial terms 00...01.

Again we ran experiments for all irreducible polynomials g of degree n up to 14, where $2^n - 1$ is not prime. We computed, for both Fibonacci and Galois LFSRs, the linear combinations of registers needed to produce the required sequences needed for interleaving. Tables 3 and 2 show a few examples of possible constructions using Fibonacci and Galois LFSRs respectively. The tables display which registers need to be XORed in order to produce the sequences required.

Table 1. Generating m-Sequences by Interleaving Registers of Several Galois LFSRs

$n = 4$ $g = 11111$	$d = 5$ $q = 3$ $\alpha = 1110$	$f = 10011$
Classes: $(0, 1, 1, 0)$	Shifts: $(4, 4, 1, 0)$	LFSRs: 2
Interleave: $(R^{(0)}, a, 0), (R^{(1)}, a, 1), (R^{(0)}, a\alpha, 0)$		
$n = 6$ $g = 1010111$	$d = 21$ $q = 3$ $\alpha = 101$	$f = 1100001$
Classes: $(0, 2, 1, 1, 0, 0)$	Shifts: $(20, 8, 1, 0, 1, 0)$	LFSRs: 2
Interleave: $(R^{(0)}, a, 0), (R^{(2)}, a, 0), (R^{(2)}, a\alpha, 0)$		
$n = 8$ $g = 111010111$	$d = 17$ $q = 15$ $\alpha = 10011100$	$f =$
Classes = $(0, 3, 13, 13, 13, 13, 3, 0)$	Shifts = $(16, 0, 13, 12, 0, 16, 2, 0)$	LFSRs: 6
Interleave: $(R^{(0)}, a, 0), (R^{(0)}, a\alpha, 0), (R^{(0)}, a\alpha^2, 0), (R^{(7)}, a, 0), (R^{(7)}, a\alpha, 0), (R^{(7)}, a\alpha^2, 0),$ $(R^{(3)}, a\alpha^8, 0), (R^{(3)}, a\alpha^9, 0), (R^{(0)}, a\alpha^8, 0), (R^{(0)}, a\alpha^9, 0), (R^{(0)}, a\alpha^{10}, 0), (R^{(7)}, a\alpha^8, 0),$ $(R^{(7)}, a\alpha^9, 0), (R^{(3)}, a, 0), (R^{(3)}, a\alpha, 0)$		
$n = 9$ $g = 1001100101$	$d = 73$ $q = 7$ $\alpha = 111011111$	$f = 1001101111$
Classes: $(0, 0, 5, 5, 5, 4, 0, 0, 0)$	Shifts: $(72, 71, 0, 72, 71, 63, 2, 1, 0)$	LFSRs: 4
Interleave: $(R^{(0)}, a, 0), (R^{(0)}, a\alpha, 0), (R^{(6)}, a\alpha^4, 0), (R^{(6)}, a\alpha^5, 0), (R^{(0)}, a\alpha^4, 0), (R^{(6)}, a, 0), (R^{(6)}, a\alpha, 0)$		
$n = 10$ $g = 10000001111$	$d = 341$ $q = 3$ $\alpha = 1010000110$	$f = 10010000001$
Classes: $(0, 1, 2, 0, 0, 0, 0, 0, 0, 0)$	Shifts: $(340, 1, 101, 6, 5, 4, 3, 2, 1, 0)$	LFSRs: 2
Interleave: $(R^{(0)}, a, 0), (R^{(8)}, a, 1), (R^{(0)}, a\alpha^2, 0)$		

Table 2. Generating m-Sequences by XORing and Interleaving the Registers of one Galois LFSR

$n = 4$ $g = 11111$	$d = 5$ $q = 3$ $\alpha = 1010$	$f = 10011$
Classes: $(0, 2, 2, 0)$	Shifts: $(4, 2, 4, 0)$	XORs: 2
Interleave: $R^{(1)} \oplus R^{(2)} \oplus R^{(3)}, R^{(2)}, R^{(3)}$		
$n = 6$ $g = 1010111$	$d = 21$ $q = 3$ $\alpha = 101$	$f = 1000011$
Classes: $(0, 2, 1, 1, 0, 0)$	Shifts: $(20, 8, 1, 0, 1, 0)$	XORs: 2
Interleave: $R^{(5)}, R^{(3)}, R^{(1)} \oplus R^{(3)} \oplus R^{(5)}$		
$n = 8$ $g = 101110101$	$d = 85$ $q = 3$ $\alpha = 100101$	$f = 101011111$
Classes: $(0, 0, 0, 2, 2, 0, 0, 0)$	Shifts: $(84, 14, 13, 45, 49, 32, 1, 0)$	XORs: 4
Interleave: $R^{(3)} \oplus R^{(4)} \oplus R^{(6)}, R^{(2)} \oplus R^{(5)} \oplus R^{(7)}, R^{(4)}$		
$n = 8$ $g = 110100011$	$d = 85$ $q = 3$ $\alpha = 10010000$	$f = 110001101$
Classes: $(0, 1, 1, 1, 1, 1, 0)$	Shifts: $(84, 67, 66, 65, 64, 70, 69, 0)$	XORs: 4
Interleave: $R^{(3)} \oplus R^{(5)} \oplus R^{(6)}, (R^{(2)} \oplus R^{(3)}), (R^{(2)} \oplus R^{(3)}) \oplus R^{(6)}$		
$n = 10$ $g = 10010011001$	$d = 341$ $q = 3$ $\alpha = 1110011010$	$f = 10000100111$
Classes: $(0, 0, 0, 1, 1, 1, 1, 0, 0, 0)$	Shifts: $(340, 339, 338, 85, 91, 90, 89, 2, 1, 0)$	XORs: 5
Interleave: $R^{(3)} \oplus R^{(6)} \oplus R^{(8)}, R^{(8)} \oplus R^{(9)}, R^{(4)} \oplus R^{(5)} \oplus R^{(9)}$		
$n = 10$ $g = 10010011001$	$d = 341$ $q = 3$ $\alpha = 1111100010$	$f = 10001100101$
Classes: $(0, 0, 0, 2, 2, 2, 2, 0, 0, 0)$	Shifts: $(340, 339, 338, 312, 318, 317, 316, 2, 1, 0)$	XORs: 5
Interleave: $R^{(3)} \oplus (R^{(7)} \oplus R^{(8)}), R^{(4)} \oplus (R^{(7)} \oplus R^{(8)}), R^{(4)} \oplus R^{(7)} \oplus R^{(9)}$		
$n = 10$ $g = 10000001111$	$d = 341$ $q = 3$ $\alpha = 1010000110$	$f = 10000001001$
Classes: $(0, 1, 2, 0, 0, 0, 0, 0, 0, 0)$	Shifts: $(340, 1, 101, 6, 5, 4, 3, 2, 1, 0)$	XORs: 2
Interleave: $R^{(3)}, R^{(5)} \oplus R^{(6)}, R^{(7)} \oplus R^{(9)}$		

Interleaving these sequences in order yields the m-sequence with minimal polynomial f as shown in the table. Some optimisations are denoted by brackets, for example in Table 2 the fourth example shows that the computation $(R^{(2)} \oplus R^{(3)})$ can then be reused for computing $(R^{(2)} \oplus R^{(3)}) \oplus R^{(6)}$ with only one additional \oplus . A full optimisation is outside the scope of this paper. For all lengths in our

Table 3. Generating m-Sequences by XORing and Interleaving the Registers of one Fibonacci LFSR

$n = 4$	$g = 11111$	$d = 5$	$q = 3$	$\alpha = 1010$	$f = 10011$	XORs: 2
Interleave: $Q^{(2)} \oplus Q^{(0)}, Q^{(1)} \oplus Q^{(0)}, Q^{(0)}$						
$n = 6$	$g = 1010111$	$d = 21$	$q = 3$	$\alpha = 101$	$f = 1000011$	XORs: 2
Interleave: $Q^{(0)}, Q^{(2)} \oplus Q^{(0)}, Q^{(4)} \oplus Q^{(0)}$						
$n = 6$	$g = 1010111$	$d = 21$	$q = 3$	$\alpha = 111000$	$f = 1101101$	XORs: 4
Interleave: $Q^{(3)} \oplus Q^{(0)}, Q^{(3)} \oplus Q^{(1)}, Q^{(4)} \oplus Q^{(2)} \oplus Q^{(0)}$						
$n = 8$	$g = 101110111$	$d = 85$	$q = 3$	$\alpha = 1100011$	$f = 100011101$	XORs: 4
Interleave: $Q^{(3)} \oplus Q^{(1)} \oplus Q^{(0)}, Q^{(5)} \oplus Q^{(2)}, Q^{(4)} \oplus Q^{(0)}$						
$n = 8$	$g = 110001011$	$d = 85$	$q = 3$	$\alpha = 11101011$	$f = 101101001$	XORs: 4
Interleave: $Q^{(4)} \oplus Q^{(1)}, Q^{(5)} \oplus Q^{(3)}, Q^{(4)} \oplus Q^{(2)} \oplus Q^{(0)}$						
$n = 8$	$g = 111011101$	$d = 85$	$q = 3$	$\alpha = 1111110$	$f = 101110001$	XORs: 4
Interleave: $Q^{(4)} \oplus Q^{(3)} \oplus Q^{(1)}, Q^{(5)} \oplus Q^{(1)}, Q^{(3)} \oplus Q^{(0)}$						
$n = 8$	$g = 100111111$	$d = 85$	$q = 3$	$\alpha = 1111100$	$f = 110101001$	XORs: 5
Interleave: $Q^{(4)} \oplus Q^{(3)} \oplus Q^{(1)}, Q^{(5)} \oplus Q^{(2)}, (Q^{(5)} \oplus Q^{(2)}) \oplus Q^{(4)} \oplus Q^{(0)}$						
$n = 10$	$g = 10000001111$	$d = 341$	$q = 3$	$\alpha = 11000110$	$f = 11100011101$	XORs: 5
Interleave: $Q^{(6)} \oplus Q^{(5)} \oplus Q^{(1)} \oplus Q^{(0)}, (Q^{(6)} \oplus Q^{(5)}) \oplus Q^{(3)} \oplus Q^{(2)}, Q^{(0)}$						
$n = 10$	$g = 10000001111$	$d = 341$	$q = 3$	$\alpha = 1010000110$	$f = 10000001001$	XORs: 2
Interleave: $Q^{(6)}, Q^{(4)} \oplus Q^{(3)}, Q^{(2)} \oplus Q^{(0)}$						

experiment we were able to determine at least one efficient construction, in the sense that for obtaining each of the required sequences the maximum number of XORs was $\lfloor (n+3)/2 \rfloor$, but for most n this value was as low as $\lfloor (n+1)/2 \rfloor$.

5 Conclusion

We developed a general framework for efficiently producing several elementary sequences (i.e. linearly recurrent sequences with an irreducible characteristic polynomial) such that they are inequivalent under cyclic shifts and if needed are, moreover, in suitable phases so that interleaving them produces an m-sequence. Experimental data identified irreducible polynomials particularly suited for this purpose. Future work will aim to further improve the efficiency of the method by analysing in more depth the gate complexity of the different constructions, combining the different approaches, and improving the efficiency of determining the initial states.

References

1. Blackburn, S.R.: Increasing the Rate of Output of m -Sequences. Information Processing Letters 51(2), 73–77 (1994)
2. Kagaris, D.: Multiple-Seed TPG Structures. IEEE Transactions on Computers 52(12), 1633–1639 (2003)

3. Lempel, A., Eastman, W.L.: High speed generation of maximum length sequences. *IEEE Transactions on Computers* 20(2), 227–229 (1971)
4. Lidl, R., Niederreiter, H.: *Introduction to Finite Fields and Their Applications*. Cambridge University Press (1994)
5. MacWilliams, F.J., Sloane, N.J.A.: *The Theory of Error-Correcting Codes*. North-Holland (1978)
6. Robshaw, M.J.B.: Increasing the Rate of Output for m Sequences. *Electronics Letters* 27(19), 1710–1712 (1991)
7. Sălăgean, A., Gardner, D., Phan, R.: Index Tables of Finite Fields and Modular Golomb Rulers. In: Helleseth, T., Jedwab, J. (eds.) *SETA 2012*. LNCS, vol. 7280, pp. 136–147. Springer, Heidelberg (2012)
8. Surböck, F., Weinrichter, H.: Interlacing Properties of Shift-Register Sequences with Generator Polynomials Irreducible Over $\text{GF}(p)$. *IEEE Transactions on Information Theory* 24(3), 386–389 (1978)
9. Udar, S., Kagaris, D.: LFSR Reseeding with Irreducible Polynomials. In: *On-Line Testing Symposium*, pp. 293–298 (2007)

On the Homomorphic Computation of Symmetric Cryptographic Primitives

Silvia Mella and Ruggero Susella

STMicroelectronics, Agrate Brianza (MB), Italy

Abstract. We present an analysis on the homomorphic computability of different symmetric cryptographic primitives, with the goal of understanding their characteristics with respect to the homomorphic evaluation according to the BGV scheme. Specifically, we start from the framework presented by Gentry, Halevi and Smart for evaluating AES. We provide an improvement of it, then we perform a detailed evaluation on the homomorphic computation of cryptographic algorithms of different families (Salsa20 stream cipher, SHA-256 hash function and KECCAK sponge function). After the analysis, we report the performance results of the primitives we have implemented using the recently released HELib. In the conclusions we discuss our findings for the different primitives we have analyzed to draw a general conclusion on the homomorphic evaluation of symmetric cryptographic primitives.

Keywords: homomorphic encryption, cryptographic primitives, AES, SHA-256, keccak, Salsa20, HELib.

1 Introduction

Fully homomorphic encryption (FHE) allows to perform arbitrary computations on encrypted data, without ever decrypting them. The concept was introduced by Rivest, Adleman and Dertouzos shortly after the presentation of RSA [1], but it took 30 years before the first plausible candidate for FHE was presented. This result is due to Gentry [2] and since his work a number of FHE schemes have been presented [3,4,5,6,7]. Unfortunately, all of these schemes are practical only for a limited number of operations. Of particular interest is the BGV cryptosystem due to Brakerski, Gentry and Vaikuntanathan [8], which presents a good compromise among efficiency, practicability and allowed operations.

In [9], Gentry, Halevi and Smart use the BGV cryptosystem to homomorphically evaluate AES circuit. That is, to compute an AES encryption with message and secret key encrypted under the homomorphic scheme.

Following their framework, we analyze the evaluation of different families of symmetric cryptographic primitives. Specifically, we focus on the SHA-256 hash function, the Salsa20 stream cipher and the KECCAK sponge function (which has been selected as the new standard SHA-3). For each algorithm we highlight the characteristics that have an impact on the homomorphic evaluation. We describe the possible state encoding techniques, underlining the different features they

provide for the evaluation of basic operations, such as logical XOR, logical AND and integer additions, showing also how the choice of the encoding impacts on the required number of levels. Then, for each practically computable primitive, we present the performance results of our implementations together with the selected values for the BGV parameters.

Our implementations focus on optimizing the execution time of single evaluations and leverages on the recently released HELib by Halevi and Shoup [10], a publicly available library which implements the BGV cryptosystem. Apart from providing the first public results on the homomorphic evaluation of cryptographic primitives different than AES, we also present an improvement in the evaluation of AES, which allows to lower the required number of levels and thus provides the best results publicly available in literature.

In section 2 we briefly introduce the BGV cryptosystem and its properties. In sections 3, 4, 5 and 6, we describe the main block of each primitive and show how to homomorphically evaluate it, providing costs estimations. Finally, in section 7 we give some execution results, obtained using HELib.

2 The BGV Scheme

In this section we briefly describe the main characteristics of the BGV cryptosystem due to Gentry, Halevi and Smart. For more details on the scheme’s initialization and working principles, we remind to the original paper [8] and consecutive optimizations [11,9].

The plaintext space is defined as $R_2 = \mathbb{Z}_2[x]/\Phi_m(x)$, where Φ_m denotes the m -th cyclotomic polynomial. A ciphertext is of the form $(c_0, c_1) \in R_{q_t} \times R_{q_t} = \mathbb{Z}_{q_t}[x]/\Phi_m(x) \times \mathbb{Z}_{q_t}[x]/\Phi_m(x)$, for some integer q_t in the chain of moduli $q_{L-1} > \dots > q_1 > q_0$. We call the q_t ’s the “levels” of the scheme.

The decryption function is defined as $[[c_0 - c_1 \cdot s] \bmod q_t] \bmod 2$ and the term $[c_0 - c_1 \cdot s] \bmod q_t$ is called the *noise* in the ciphertext c . Performing addition and multiplication, the noise at most doubles and squares respectively. The condition for correct decryption is that the norm of the noise is sufficiently small to not wrap around q_t during operations.

A *modulus switching* technique can be exploited to keep the noise magnitude essentially constant. This technique simply consists of transforming a ciphertext c w.r.t. modulus q_t into a ciphertext c' w.r.t. modulus q_{t-1} , while preserving correctness (namely, $[[c'_0 - c'_1 \cdot s] \bmod q_{t-1}] \bmod 2 = [[c_0 - c_1 \cdot s] \bmod q_t] \bmod 2$).

In particular, the modulus switching is really necessary just before multiplication, namely when the noise magnitude is going to be squared. We say that homomorphic multiplication “consumes a level” of the scheme. During other operations it is acceptable to keep higher noise, until the noise estimate indicates that a modulus switching must be performed.

An a priori estimate of required levels is needed to set the moduli chain of the scheme.

SIMD Operations and Automorphisms. Smart and Vercauteren observed [12,13] that the algebraic structure of the plaintext space allows SIMD operations. Specifically, setting m odd, the plaintext space R_2 is isomorphic to the direct sum of ℓ copies of $\text{GF}(2^d)$. It follows that a plaintext can be viewed as a ℓ -vector of elements in $\text{GF}(2^d)$ and vice-versa. Moreover, arithmetic over plaintexts correspond to element-wise arithmetic over ℓ -vectors. Elements of this ℓ -vector are usually referred to as slots.

In [13] it is shown how to move the content of a slot in another one, using automorphisms over R_2 . Recall that for any $i \in \mathbb{Z}_m^*$ the automorphism κ_i over R_2 is defined as $\kappa_i : a(x) \mapsto a(x^i) \pmod{\Phi_m(x)}$. Up to a reordering of factors F_i , there exists an integer h such that if $a \in R$ encodes the ℓ -vector $(a_0, \dots, a_{\ell-1})$, then $\kappa_h(a)$ encodes the ℓ -vector $(a_{\ell-1}, a_0, \dots, a_{\ell-2})$. Thus, applying the automorphism κ_h results in a cyclic rotation of slots to the right. Whereas, applying $\kappa_{h^{-1}}$ results in a cyclic rotation to the left. These automorphisms belong to the group $\mathcal{H} = \mathbb{Z}_m^* / \langle 2 \rangle$ (see [11], Appendix C.2 for more details).

An arbitrary permutation of slots can be implemented combining the action of κ_h (and $\kappa_{h^{-1}}$) with addition and *select* operation [11]. The *select* operation simply consists of multiplication by constant selection vector, which consumes 0.5 levels [9] (Section 3.4).

Of particular interest are also the Frobenius automorphisms, which have the form κ_{2^j} . This automorphisms act on each slot separately. That is, the content of each slot is elevated to 2^j in the field $\text{GF}(2^d)$.

Notice that, if c encrypts the plaintext a under the key s , the map κ_i computed on each component of c gives an encryption of $\kappa_i(a)$ with respect to key $\kappa_i(s)$. It is then possible to transform $\kappa_i(c)$ into a ciphertext that encrypts $\kappa_i(a)$ with respect to the original key s , using a key-switching procedure [9] (Appendix B.3). The computation of automorphisms has no influence on the noise magnitude, but the key-switching procedure increase it somewhat [11] (Appendix D.2).

3 Homomorphic Evaluation of AES-128

The Advanced Encryption Standard [14] is a block cipher standardized by NIST in 2001. It is based on the Rijndael cipher [15], developed by Daemen and Rijmen and based on the substitution-permutation network design.

For the evaluation of AES-128, we follow the framework of Gentry, Halevi and Smart [9].

Recall on AES-128. AES-128 encryption function consists of a sequence of 10 identical rounds (except for the last one) operating on a 128-bit state, which is organized as an array of 4×4 bytes. Bytes are considered as elements of $\text{GF}(2^8)$, defined by the Rindael's polynomial $x^8 + x^4 + x^3 + x + 1$. Each round consists of the following operations, the only exception being the last round which does not compute the MixColumns.

- **AddRoundKey** simply xor the state with a 128-bit round key derived from the secret key.

- **SubBytes** computes on each byte the inversion in $\text{GF}(2^8)$ followed by an affine transformation over $\text{GF}(2)$.
- **ShiftRows** right rotates the i -th row of the state matrix by i positions.
- **MixColumns** multiplies the state matrix by a fixed matrix of the same dimension.

3.1 Evaluation of AES-128 Encryption Function

We now describe the three implementations of AES-128 encryption function presented in [9]. In particular, we show an improvement of the MixColumns computation for the packed implementation. Note that for all implementations, each round key is precomputed and encrypted with the homomorphic scheme.

Packed Implementation. First of all, the polynomial Φ_m is chosen such that it factors modulo 2 into at least 16 irreducible polynomials of degree d with $8|d$. It means that each slot can hold a byte of the state, that is an element of $\text{GF}(2^8)$ embedded in $\text{GF}(2^d)$. Notice that, because of parameters setting, the number of available slots could exceed the needed one. We can thus pad with zero the exceeding slots, or we can encode more states in the same ciphertext, obtaining parallel executions.

- **AddRoundKey** simply consists in the sum of ciphertext encrypting the state with the ciphertext encrypting the RoundKey.
- **SubBytes** is computed into 2 steps. Inversion is obtained with three Frobenius automorphisms and a depth-3 circuit of multiplications. The affine transformation is computed using seven Frobenius automorphisms and a linear combination with precomputed coefficients. The cost estimate for this operation is 3.5 levels: 3 for inversion, due to multiplication, and 0.5 for affine transformation, due to multiplication by constant.
- **ShiftRows** and **MixColumns** are computed as a single linear transformation over $\text{GF}(2^8)^{16}$, combining four permutations of the state (π_0 through π_3) via a linear operation: $2 \cdot \pi_0 + 3 \cdot \pi_1 + \pi_2 + \pi_3$.

In [9] the permutations π_i are computed combining automorphisms and select operations. Then, the linear combination with coefficients 1, x and $x + 1$ is computed. In our implementation, we compute select operations and multiplications by x and $x + 1$ simultaneously. The constant selection vectors used to compute π_0 and π_1 are actually set in order to have x and $x + 1$ (instead of 1) in the appropriate slots. This trick allows to save a constant multiplication per round, which results in a smaller amount of consumed levels for the global evaluation and thus a faster execution. In particular, this operation requires 0.5 levels, instead of 1 level as in [9]

In conclusion, we estimate a consumption of 4 levels for one round and hence of 40 levels for the whole AES function.

Byte-Sliced Implementation. This implementation uses sixteen ciphertexts, each encoding a byte of the state. As above, we need the polynomial Φ_m to factor modulo 2 into irreducible polynomial of degree d . However, in this case, we are not interested in the number of factors, since we need just a slot for ciphertext.

The operations are almost the same as in the packed implementation, the only difference being for ShiftRows and MixColumns. Actually, we do not need to compute permutation of slots, but just the multiplications by constant. Hence, we need again 4 levels per round: 3.5 for SubBytes and 0.5 for the multiplications by constant during MixColumns. Since the last round of AES does not perform MixColumns, we need a total of 39.5 levels.

Bit-Sliced Implementation. This implementation uses 128 ciphertexts (one per bit) and computes the AES function as a binary circuit. In particular, the SubBytes operation is computed using the Boyar and Peralta circuit [16], which consumes 4 levels. The other operations of AES are essentially linear. Hence, we get an estimate of 4 levels per round.

4 Homomorphic Evaluation of SHA-256

SHA-256 is a hash function developed by the NSA and published by NIST in 2001 [17], it is based on the Merkle-Damgård construction [18].

Recall on SHA-256. The hash function SHA-256 computes on 32-bit variables, combining XOR, AND, rotations and addition modulo 2^{32} .

A *message schedule* extends the 512-bit message, considered as 32-bit words $w[0], \dots, w[15]$, into other 32-bit words $w[16], \dots, w[63]$ as follows:

- $s_0 = (w[i - 15] \ggg 7) \oplus (w[i - 15] \ggg 18) \oplus (w[i - 15] \gg 3)$
- $s_1 = (w[i - 2] \ggg 17) \oplus (w[i - 2] \ggg 19) \oplus (w[i - 2] \gg 10)$
- $w[i] = w[i - 16] + s_0 + w[i - 7] + s_1$

The 64 words are then compressed applying the following loop, for i from 0 to 63, which uses 8 working variables, a through h , that have been initialized out of the function:

- $s_1 = (e \ggg 6) \oplus (e \ggg 11) \oplus (e \ggg 25)$
- $ch = (e \wedge f) \oplus (-e \wedge g)$
- $temp_1 = h + s_1 + ch + k[i] + w[i]$
- $s_0 = (a \ggg 2) \oplus (a \ggg 13) \oplus (a \ggg 22)$
- $maj = (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c)$
- $temp_2 = s_0 + maj$
- $h = g, g = f, f = e, e = d + temp_1, d = c, c = b, b = a, a = temp_1 + temp_2$

The compressed message is then added to the intermediate state h_0, \dots, h_7 :

- $h_0 = h_0 + a; h_1 = h_1 + b; h_2 = h_2 + c; h_3 = h_3 + d; h_4 = h_4 + e; h_5 = h_5 + f; h_6 = h_6 + g; h_7 = h_7 + h;$

4.1 Encoding Choice and Carry-lookahead Adders

The main problem in the homomorphic evaluation of SHA-256 is the evaluation of operations in different algebraic structures, that is XOR and AND over $\text{GF}(2^{32})$ and integer addition over $\mathbb{Z}_{2^{32}}$.

There are three main encoding techniques that we have considered, but only the last one allows to perform all operations.

- **Words in plaintext coefficients:** the modulus of the plaintext space is set as $t > 2$ (this is allowed by the BGV scheme). Each integer is broken into its binary representation and encoded in R_t as a polynomial with a bit per coefficient. Addition of two integers is simply obtained by homomorphically adding ciphertexts. The resulting plaintext will indeed give the correct result when evaluated in $x = 2$. The modulus t allows to sum up at most $t - 1$ integers, then there will be a loss of information caused by the modulo t reduction. However, this encoding choice does not allow to compute XOR and AND operations.
- **Words in slots:** the parameters of the scheme are set such that SIMD operations are allowed. In particular, we encode each word in a slot as an element of $\text{GF}(2^{32})$. This encoding choice allows to perform binary XOR by simply adding ciphertexts. However, it does not allow evaluation of binary AND, since it is not equivalent to any operation over $\text{GF}(2^{32})$. Moreover, this encoding choice does not even allow to compute integer additions.
- **Bit in slots:** the parameter of the scheme are again set to get SIMD operations and each slot of the ciphertext contains a bit. With this encoding choice, XOR and AND can be computed by simply adding and multiplying ciphertexts. Moreover, it allows also the evaluation of integer additions as in carry adders, as shown below.

We chose to use the last encoding technique, which also presents the advantage that the field $\text{GF}(2^d)$ contains $\text{GF}(2)$ for any d . It means that there are no constraints on the choice of m , as for AES evaluation.

Carry-lookahead Adder. To compute integer addition, we implement a carry-lookahead adder. Considering two 32-bit integers $x = x_{31} \dots x_0$ and $y = y_{31} \dots y_0$, sum and carry bits are defined respectively as:

- $s_i = x_i \oplus y_i \oplus c_i$,
- $c_{i+1} = G_i \vee (c_i \wedge P_i)$ with $G_i = x_i \wedge y_i$ and $P_i = x_i \oplus y_i$.

For simplicity, consider the case where x and y are encoded in two different ciphertexts, say C_x and C_y . Specifically, the LSB is contained in slot 0 and the MSB in slot 31. We want to compute a ciphertext C , which contains carry c_i in slot i , so that the integer sum of x and y can be computed homomorphically adding C , C_X and C_Y .

We first compute two ciphertexts encoding P_i and G_i in their slots, say C_P and C_G . These ciphertexts can be computed by simply adding and multiplying

C_x and C_y . Each carry c_i can now be computed using rotation, addition and multiplication of C_P and C_G .

We focus on the computation of carry c_{31} , which is the most expensive in terms of operations and consumed levels. For other carries, the computation is analogous. Substituting c_i into c_{i+1} , for any i , we get $c_{31} = G_{30} \vee (G_{29} \wedge P_{30}) \vee \dots \vee (G_0 \wedge P_1 \wedge \dots \wedge P_{30})$.

We first create copies of C_G and G_P and, applying automorphisms, we move the desired values in slot 31. The most expensive term of c_{31} is the last one, consisting of thirty AND operations, which can be computed using a depth-5 circuit of homomorphic multiplications. The other terms can be computed using from 1 to 5 levels. Then, to evaluate the OR operation, we apply the De Morgan law: $A \vee B = \neg(\neg A \wedge \neg B)$. Hence, we can compute the thirty OR operations, using additions (to get NOT) and a depth-5 circuit of multiplications. Finally, we select slot 31 from the resulting ciphertext. Thus, we estimate a total amount of 10.5 consumed levels.

Analogously, we compute ciphertexts encrypting other carries and homomorphically add them to get C .

Notice that in a bit-sliced implementation, rotations and select are not needed. Thus, 10 levels are consumed.

4.2 Evaluation of SHA-256 Function

We present here three possible implementation variants for SHA-256, providing an estimate of the number of consumed levels.

Word-Sliced Implementation. We start analyzing the message schedule, which extends sixteen 32-bit words into other forty-eight.

- The computation of s_0 and s_1 requires rotation of slots (obtained combining automorphisms and select operations) and addition of ciphertexts. Namely, it consumes just 0.5 levels.
- The computation of $w[i]$ requires four integer additions that can be implemented using a depth-2 circuit of integer additions, which requires $2 \cdot 10.5$ levels of multiplications.

Notice that the computation of $w[i]$ uses $w[i-2]$. It follows that for $i \geq 18$ the homomorphic computation involves a variable which has already been scaled down by a number of levels. Specifically, $w[i]$ is at level $L-1 - (\lfloor \frac{i-16}{2} \rfloor + 1) \cdot 21.5$.

We now analyze the evaluation of the compression function:

- The homomorphic computation of s_1 and s_0 requires rotations of slots and addition of plaintexts. Hence, it requires only 0.5 levels. Since these variables will be involved in operations with variables computed in the next point, they have to be scaled down by other 0.5 levels.
- The computation of ch and maj requires one level, since a binary AND of slots must be calculated.

- $temp_1$ and $temp_2$ are computed using integer addition. In particular, the computation of $temp_2$ requires 10.5 levels, as it consists of a single addition. The computation of $temp_1$ is the most expensive. It can be implemented using a depth-3 circuit of integer additions, which requires $3 \cdot 10.5 = 31.5$ levels of homomorphic multiplications.
- In the last step, there is a reassignment of variables, which includes the computation of integer additions for $e = d + temp_1$ and for $a = temp_1 + temp_2$. That is, additional 10.5 levels are required.

In conclusion, 43 levels are required for each round. Hence, a total of $64 \cdot 43 = 2752$ levels. Finally, additional 10.5 levels are required for the addition with the intermediate state.

Notice that, for each i , variable $w[i]$ is involved in the main loop when the computation level is already smaller than its own. It follows that the estimation of the number of required levels is given just by the estimation of the compression function and the final additions. Hence, it is $64 \cdot 43 + 10.5 = 2762.5$.

Packed Implementation. All 32-bit variables are encoded in just one ciphertext. This means that we need $80 \cdot 32$ slots.

The evaluation of the message schedule requires the same number of levels to compute s_0 and s_1 , since we rotate words consuming 0.5 levels and contemporary move them in the desired position. On the other hand, the computation of $w[i]$ requires additional rotations in order to move words in the same slots before computing integer additions.

As in the word-sliced implementation, $w[i]$ is at level $L - 1 - (\lfloor \frac{i-16}{2} \rfloor + 1) \cdot 21.5$, which means that $w[63]$ is at level $L - 1 - 516$.

The homomorphic evaluation of the compression function is essentially the same as in the word-sliced implementation, with some additional rotations to move words in the same slots before computing. In particular, in the last step we consume additional 0.5 levels to compute the reassignment of variables. In conclusion, 43.5 levels are required for each round.

In this implementation, when we evaluate the compression function, the ciphertext has already been scaled down by 516 levels. Hence, the total amount of consumed levels is $516 + 64 \cdot 43.5 + 10.5 = 3310.5$.

Bit-Sliced Implementation. This implementation uses a ciphertext for each bit, hence a total of $80 \cdot 32$ ciphertexts. The main advantage is that we avoid automorphisms and select operations, saving a number of levels.

The message schedule requires a total of 20 levels for the computation of $w[i]$. Hence, ciphertexts encoding bits of $w[i]$ are at level $L - 1 - (\lfloor \frac{i-16}{2} \rfloor + 1) \cdot 20$. That means that $w[63]$ is at level $L - 1 - 480$. The compression function requires a total of 41 levels per round and additional 10 levels for the integer addition with the intermediate state. As in the word-sliced implementation, the estimation of the number of required levels is given just by the estimation of the compression function and the final additions, since variable $w[i]$ is involved in the main loop

when the computation level is already smaller than its own. Hence, the total consumption is of $64 \cdot 41 + 10 = 2634$ levels.

5 Homomorphic Evaluation of Salsa20

Salsa20 [19] is a stream cipher designed by Bernstein and selected as part of the eSTREAM portfolio. It provides 64 bytes of output after 10 simple rounds, thus performing very fast in software implementations. For details about the specification please refer to [20].

Recall on Salsa20. Salsa20 operates on a state that can be seen as a 4×4 matrix with elements in 2^{32} . It is based on a simple function named *quarter-round*, which is applied in parallel to each column and then to each row of the state matrix, for a total of 10 rounds. This function operates on four 32-bit values. In particular, if $y = (y_0, y_1, y_2, y_3)$ then $quarter-round(y) = (z_0, z_1, z_2, z_3)$ where

- $z_1 = y_1 \oplus ((y_0 + y_3) \lll 7)$
- $z_2 = y_2 \oplus ((z_1 + y_0) \lll 9)$
- $z_3 = y_3 \oplus ((z_2 + z_1) \lll 13)$
- $z_0 = y_0 \oplus ((z_3 + z_2) \lll 18)$

5.1 Evaluation of Salsa20

Since Salsa20 requires to combine a logical operation (XOR) and addition modulo 2^{32} , the only possible encoding choice is to have a bit in each slot, as for the evaluation of SHA-256. We present here three different implementations for Salsa20, estimating the number of required levels.

Word-Sliced Implementation. The word-sliced implementation is the most intuitive one due to Salsa20 structure of applying *quarter-round* first to columns and then to rows, since each element of the state matrix is accessed through a pointer. The first element to be computed in *quarter-round* is z_1 , and its computation requires an integer addition and a rotation, which means 10.5 levels due to the adder and half level due to the select used in the rotation, for a total of 11 levels. Then, since all the sequent elements computed by *quarter-round* recursively depend on each other, the last value to be computed (z_0) will require $11 \times 4 = 44$ levels. This result can be applied to the other half round in the same way. This results in 44×2 consumed levels for each round, for a total of 880 levels.

Packed Implementation. In the packed implementation the state is encoded in a single ciphertext, thus requiring 512 slots. The *quarter-round* function is applied in parallel, this means that in the first step all the additions of the first and last elements of each column (or row) are performed together. However, to write the result back to the ciphertext we need to select only the column (or row) on which we operated, therefore each *quarter-round* must be followed by a select, resulting in 92 levels per round or 920 total levels.

Bit-Sliced Implementation. In the bit-sliced implementation we require 512 ciphertexts but we remove any cost due to the rotations of words, therefore *quarter-round* will cost 10×4 just due to additions and the total execution would require 800 levels.

6 Homomorphic Evaluation of KECCAK

KECCAK [21,22] is a cryptographic primitive, based on the sponge construction [23], which allows the construction of hash functions, symmetric encryption and authentication functions, reseeder pseudo-random bit generator, etc. It has been recently selected by NIST as the winner of the SHA-3 competition.

Recall on KECCAK. The main block of KECCAK is the KECCAK- $f[b]$ permutation, which consists in a sequence of identical rounds operating on a b -bit state. A state A is organized as an array of 5×5 lanes each of length $w \in \{1, 2, 4, 8, 16, 32, 64\}$. Hence the size of the state is $b = 25w$, i.e. $b \in \{25, 50, 100, 200, 400, 800, 1600\}$.

The number of rounds n_r depends on the state size and is given by $n_r = 12 + 2\ell$ where $\ell = \log_2 w$. A round consists of a sequence of invertible steps on the state $A[x][y][z]$:

- θ **step** is a linear map and can be divided into three steps:

$$C[x] = A[x, 0] \oplus A[x, 1] \oplus A[x, 2] \oplus A[x, 3] \oplus A[x, 4] \quad \text{for } x = 0, \dots, 4$$

$$D[x] = C[x - 1] \oplus (C[x + 1] \ggg 1) \quad \text{for } x = 0, \dots, 4$$

$$A[x, y] = A[x, y] \oplus D[x] \quad \text{for } x, y = 0, \dots, 4$$

- ρ **step** consists in a bitwise cyclic shift operation within the lanes.

- π **step** is a transposition of the lanes.

- χ **step** is the only non-linear map of the round, defined as:

$$A[x, y] = B[x, y] \oplus (\neg B[x + 1, y] \wedge B[x + 2, y]) \quad \text{for } x, y = 0, \dots, 4$$

- ι **step** consists in a single bitwise XOR between the lane $A[0, 0]$ and a constant lane RC defined for each round.

6.1 Evaluation of KECCAK- f

We describe here three different implementations.

As we have explained in Section 4.1, the binary AND operation does not allow to encode multiple bits in the same slot. For this reason, in all implementations each slot contains a bit.

Packed Implementation. The whole state A is encoded in a single plaintext a , using the map $A[x][y][z] \mapsto a[wx + 5wy + z]$.

- θ **step** consists of rotations of slots and addition of ciphertexts. To compute $C[x]$ for each column, we need to move lanes corresponding to the same column in the same slots. Hence, we need to perform four left-rotations (no

select operation is needed) and additions. Notice that we are interested only in the content of slots 0 through $5w$ but we do not select them now in order to save one level of multiplications. By performing again rotations (with select), we get $C[x - 1]$ and $C[x + 1] \ggg 1$. At this point, we contemporary select slots 0 through $5w$, which are the ones we are interested in. A homomorphic addition is then performed to obtain D . Finally, four right-rotations are computed to copy $D[x]$ in slots corresponding to all lanes of column x and a final addition is performed. The main level cost of θ lies in select operations. Hence, 0.5 levels are required.

- ρ and π steps are simultaneously computed as a big permutation of slots, using automorphisms and select operations. The total cost for the steps is 0.5 levels, due to constant multiplication of select operations.
- χ step is the only operation which requires multiplication of ciphertexts. To get $B[x + 1, y]$ and $B[x + 2, y]$ rotations are computed as usual. The NOT operation is obtained by addition with a constant vector encoding a 1 in each slot, whereas the AND by multiplication of ciphertexts. Then, a final homomorphic addition is computed
- ι step consists of a single addition with a constant. In particular, the constant RC is precomputed encoding it in the first w slots of a ciphertext.

The estimated number of levels needed by the round function is 2.5, therefore KECCAK- $f[b]$ should require $2.5(12 + 2\ell)$ levels.

Lane-Sliced Implementation. In this implementation, we use 25 different ciphertexts, each encrypting a lane. In particular the slot z contains the bit $A[\cdot][\cdot][z]$.

- θ step requires a smaller number of rotations than the packed implementation. Computation of $C[x]$ for each column simply consists in homomorphic addition of ciphertexts corresponding to different rows of the same column. Then, we compute $D[x]$ by performing lane rotations (to get $C[x + 1] \ggg 1$) and homomorphic additions. Finally, we loop on each ciphertext homomorphically adding the relative $D[x]$. The level cost of this step is again 0.5, due to lane rotations.
- ρ step is obtained performing rotations over each ciphertext. The level cost is 0.5.
- π step is performed without any homomorphic computation, in fact we just reassign a vector of pointers to our ciphertexts.
- χ step does not require rotation as in packed implementation. We perform addition of ciphertexts with constant vectors to obtain the NOT. Then, we need multiplications to compute the AND and a final addition to get the result. Hence, this step consumes 1 level.
- ι step consists of a single addition with a constant.

This implementation requires a smaller number of rotations with respect to the packed implementation. In particular, the π step is for free. The total number of required levels is estimated to be 2 for each round. Hence, the evaluation of KECCAK- $f[b]$ should consume $2(12 + 2\ell)$.

Bit-Sliced Implementation. In this implementation, we use b different ciphertexts, each encrypting a bit of the state. This implementation requires no rotations, but just additions and multiplications of ciphertexts. All \ggg operations, as ρ and π steps, can actually be computed by reassigning a vector of pointers to our ciphertexts. The only step which consumes 1 level is χ . Hence, the evaluation of the whole KECCAK- $f[b]$ permutation should consume $12 + 2\ell$.

7 Implementation Results Using HELib

For our implementations we used the recently released HELib by Halevi and Shoup [10].

HELib is a NTL-based C++ library, distributed under the GPL license, and it is the first publicly available library to implements the BGV homomorphic encryption scheme. It provides all the necessary functionalities to perform our computations, as additions and multiplications of ciphertexts, automorphisms, additions and multiplications by constant values. Moreover, HELib sets the moduli chain of the scheme, given the number of levels and the security level. We set the security level equal to 80 bits, which translates to a security level matching that of AES-128 [24].

For all implementations, we focus on obtaining a fast execution for a single evaluation. It means that we set the polynomial Φ_m as small as possible to support a single evaluation. As stated in [9], Φ_m could be chosen larger in order to allow parallel evaluations. This will amortizes the time for single blocks, but the global execution will require a larger amount of time. In particular, we choose m as the smaller one satisfying the following conditions:

- $\phi(m) \geq N$ where N is a lower bound on the degree of Φ_m , depending mainly on the security level and the number of required levels. In particular $N > (L(\log N + 23) - 8.5)(k + 110)/7.2$. For details on the computation of N , we remind to [9] (Appendix C);
- ℓ equal to or greater than the the number of slots we need;
- the group $\mathcal{H} = \mathbb{Z}_m^*$ of automorphisms, used for rotations of slots, has at most one generator.

The last constraint is strictly related to HELib implementation. We can briefly say that with two or more generators, the automorphism computation implies multiplications by constants [25]. Due to the large number of performed automorphisms (to compute rotations of slots), this property results in a significant increase of the number of required levels.

We tested our implementations using the HELib on a server with an Intel Xeon X5675 running at 3GHz.

7.1 AES-128

We have implemented packed and byte-sliced variants, since the bit-sliced implementation is less attractive in terms of time and memory requirements [9].

Recall that the packed implementation uses just one ciphertext to encode the whole state. Since each slot contains a byte of the state, we need at least 16 slots. In addition, we need d to be a multiple of 8. The estimate for the number of required levels is 4 per round.

On the other hand, the byte-sliced implementation uses sixteen ciphertexts, one for each byte. The number of required levels is again 4 per round and 39.5 for the whole execution, since the last round of AES does not perform MixColumns.

Table 1. Parameters and execution costs for the evaluation of AES encryption function

	N	m	$\phi(m)$	$\log_2 q_{L-1}$	L	ℓ	Timings	Ctxt size	# Ctxts
packed	26482	30977	30976	928.76	40	16	22m	7.2MB	1
byte-sliced	26482	26497	26496	939.59	40	12	2h47m	6.2MB	16

In Tab. 1 we provide the dimensions of the scheme and the execution costs of the evaluation. In particular, we first give the actual number L of consumed levels, which exactly corresponds to the theoretical estimate we made in section 3 (note that half levels are not supported). Then, we give the smaller value of m which satisfies our conditions, together with the degree of Φ_m and its lower bound N . Then, the dimension of the larger modulo of the scheme q_{L-1} is provided.

In the last columns, we give execution time and ciphertexts size. The ciphertext size is estimated according to the formula: $2 \cdot \varphi(m) \cdot \log_2 q_{L-1}$.

Both implementations require a number of levels equal to 40. For the packed implementation we need to set m larger than for the byte-sliced implementation, because we need at least sixteen slots per ciphertext. However, the byte-sliced implementation has larger execution time and memory requirements, since we need to manipulate sixteen ciphertexts. In fact, we need to compute sixteen different evaluations of SubBytes and AddRoundKey per round, one for each ciphertext, and this results in a significant slow down of the execution.

7.2 SHA-256 and Salsa20

Both SHA-256 and Salsa20 make use of integer addition, which is the main limitation in their homomorphic evaluation. In fact, recall that each integer addition, computed through carry-lookahead adders, requires 10.5 levels (10 in the bit-sliced variant).

The total amount of required levels is very high for both algorithms and parameters should therefore be set very large, as shown in Tab. 2 and Tab. 3. In other words, we need to manipulate polynomials two orders of magnitude greater than the ones used for AES evaluation. This would make an implementation very inefficient.

We don't expect our server to be able to compute it in any reasonable amount of time. For this reason, we did not concretely implement the evaluation of SHA-256 and Salsa20.

Table 2. Parameter settings for the evaluation of SHA-256

	L	N	m	ℓ
packed	3311	2098115	2099863	48834
word-sliced	2763	1751048	1751689	72
bit-sliced	2634	1669348	1669351	6

Table 3. Parameter settings for the evaluation of Salsa20

	L	N	m	ℓ
packed	920	583815	586697	682
word-sliced	880	558482	558757	83
bit-sliced	800	507815	507821	1

7.3 KECCAK

Recall that each implementation has no requirement on d , since each slot contains a bit.

In the packed implementation, the number of necessary slots is given by the state size $b \in \{25, 50, 100, 200, 400, 800, 1600\}$. In the lane-sliced it corresponds to the lane length $w \in \{1, 2, 4, 8, 16, 32, 64\}$, whereas in the bit-sliced implementation it is always 1.

Recall that the packed implementation has an estimate of 2.5 levels per round, whereas lane-sliced of 2 levels (the only exception being KECCAK- f [25] which requires 1 level per round) and bit-sliced of just 1 level.

Table 4. Parameters and execution costs for the evaluation of KECCAK- f permutation

	b	N	m	$\phi(m)$	$\log_2 q_{L-1}$	L	ℓ	Timings	Ctxt size	# Ctxts
packed	25	16982	17467	17466	563.36	25	41	27m	2,5MB	1
	50	19515	20191	19800	661.80	29	330	1h1m	3,3MB	1
	100	22048	22111	22110	743.10	33	110	1h30m	4,1MB	1
	200	24582	25351	25000	860.81	37	250	2h8m	5,4MB	1
	400	27115	32377	32376	958.18	41	568	2h45m	7,8MB	1
	800	29648	43691	43690	1069.48	45	1285	5h9m	11,7MB	1
	1600	32182	49981	49500	1198.11	49	1650	13h53m	14,8MB	1
lane-sliced	25	9382	9391	9390	282.16	13	2	3m	0,7MB	25
	50	19515	19543	19542	642.93	29	2	31m	3.1MB	25
	100	22048	22051	22050	753.35	33	25	50m	4.2MB	25
	200	24582	24931	24592	854.88	37	25	1h39m	53.MB	25
	400	27115	27409	27408	971.03	41	48	1h54m	6.7MB	25
	800	29648	30269	30268	1054.86	45	84	2h44m	8MB	25
	1600	32182	32377	32376	1153.97	49	568	4h	9.3MB	25
bit-sliced	25	9382	9391	9390	282.16	13	2	3m	0,7MB	25
	50	10648	10651	10650	330.67	15	3	10m	0.9MB	50
	100	11915	11923	11922	375.13	17	3	25m	1.1MB	100
	200	13182	13183	13182	411.54	19	1	1h10m	1.4MB	200
	400	14448	14461	14460	462.90	21	1	2h36m	1.7MB	400
	800	15715	15727	15726	514.51	23	6	7h17m	2MB	800
	1600	16982	16987	16986	562.31	25	3	21h59m	2.4MB	1600

In Tab. 4 we provide parameter settings and execution costs for each implementation. In particular, the actual number of required levels in the packed

implementation is smaller than the estimated one. Whereas, for the other implementations it is the estimated number plus 1. This can be explained by the fact that theoretical estimation considers only multiplications and multiplication by constant, whereas also the other operations (as additions, key switching and modulo switching) increase the noise magnitude somewhat.

Analyzing execution timings, the packed implementation results the slower one except for KECCAK- f [800] and KECCAK- f [1600] where the bit-sliced implementation requires a very large amount of time, due to the large number of ciphertexts we manipulate.

However, the packed implementation is the one with the smallest memory requirements, from 2.5MB for KECCAK- f [25] to 14.8MB for KECCAK- f [1600]. In fact, the other implementations operate on multiple ciphertexts. In particular, the lane-sliced implementation requires from 17.5MB for KECCAK- f [25] to 232.5MB for KECCAK- f [1600] and the bit-sliced implementation requires from 17.5MB for KECCAK- f [25] to 3.8GB for KECCAK- f [1600].

8 Conclusions

After summarizing the BGV homomorphic encryption scheme, we performed detailed analysis of the homomorphic evaluation of cryptographic primitives of different families: AES as a substitution permutation network based block cipher, SHA-256 as an hash function, Salsa20 as a stream cipher, and KECCAK as a sponge function.

This analysis showed how certain characteristics of the primitives have an impact on their homomorphic implementation. Specifically we saw in sections 4 and 5 that the evaluation of SHA-256 and Salsa20 requires a number of levels which is too large to allow an efficient implementation. In both cases the issue relies on the requirement to compute addition modulo 2^n together with logical operations. As shown in section 4.1, the only feasible way of implementation is by using an encoding in $\text{GF}(2)$ and performing addition through carry adders, but this results in a tremendous increment in the number of levels. This result can be easily extended to all ARX based cryptographic primitives such as (among others): MD5 [26], SHA-1 [17], Threefish [27], HC-128 [28] and BLAKE [29].

Also in section 4.1 we showed that although there are several possible ways to encode the state of a primitive, the choice might be constrained due to need of computing the logical AND operation, which can be computed only when a slot encodes a single bit while all other logical operations can be computed when slots are element of $\text{GF}(2^n)$ allowing for a more advantageous encoding.

For all computable primitives we showed how the different encoding of the state would impact the memory, in terms of number of ciphertexts, and level requirements. The result of this analysis is that AES is best suited to be implemented homomorphically not only due to its low number of rounds, but especially on the fact that it requires neither integer operations nor logical ANDs, allowing for a better packed encoding. Another algorithm which can be quite efficiently implemented homomorphically is KECCAK due the simplicity of its round that requires only logical operations.

For these two primitives, the ones that are practically computable, we successfully tested our implementations using the recently release HELib. We have carefully chosen the BGV parameters to improve the execution times, by using the lowest possible number of levels and cyclotomic polynomials as small as possible and with group \mathcal{H} with at most one generator. Finally, in section 7 we showed the performance results of our implementations, including the first available performance data of a homomorphic implementation of KECCAK , and the best result to date for a homomorphic computation of AES.

References

1. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms, pp. 169–177. Academic Press (1978)
2. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC, pp. 169–178. ACM (2009)
3. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
4. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: Ostrovsky, R. (ed.) FOCS, pp. 97–106. IEEE (2011)
5. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
6. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. IACR Cryptology ePrint Archive
7. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
8. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS, pp. 309–325. ACM (2012)
9. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the aes circuit. IACR Cryptology ePrint Archive
10. Halevi, S., Shoup, V.: HELib (2013), <http://github.com/shaih/HELib>
11. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
12. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
13. Smart, N., Vercauteren, F.: Fully homomorphic simd operations. IACR Cryptology ePrint Archive
14. National Institute for Science, Technology (NIST): Advanced Encryption Standard (FIPS PUB 197) (November 2001), <http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
15. Daemen, J., Rijmen, V.: The block cipher rijndael. In: Schneier, B., Quisquater, J.-J. (eds.) CARDIS 1998. LNCS, vol. 1820, pp. 277–284. Springer, Heidelberg (2000)

16. Boyar, J., Peralta, R.: A depth-16 circuit for the aes s-box. IACR Cryptology ePrint Archive
17. National Institute for Science, Technology (NIST): Secure hash standard (shs) (fips pub 180-4) (March 2012), <http://csrc.nist.gov/publications/PubsFIPS.html>
18. Merkle, R.: Secrecy, authentication, and public key systems. PhD thesis, Stanford University (1979)
19. Bernstein, D.J.: The salsa20 family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) *New Stream Cipher Designs*. LNCS, vol. 4986, pp. 84–97. Springer, Heidelberg (2008)
20. Bernstein, D.J.: Salsa20 specification (2005), <http://cr.yp.to/snuffle/spec.pdf>
21. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The KECCAK reference (2011), <http://keccak.noekeon.org>
22. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V., Keer, R.V.: KECCAK implementation overview (2012), <http://keccak.noekeon.org>
23. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Cryptographic sponge functions (2011), <http://keccak.noekeon.org>
24. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. *J. Cryptology* 14(4), 255–293 (2001)
25. Halevi, S., Shoup, V.: HElib (2013), <http://github.com/shaih/HElib/blob/master/doc/designDocument/HElibrary.pdf>
26. Rivest, R.: Rfc 1321: The md5 message-digest algorithm (1992), <http://tools.ietf.org/html/rfc1321>
27. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The skein hash function family (2010), <http://www.skein-hash.info/sites/default/files/skein1.3.pdf>
28. Wu, H.: The stream cipher hc-128 (2004), http://www.ecrypt.eu.org/stream/p3ciphers/hc/hc128_p3.pdf
29. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.: Sha-3 proposal blake (2010), <https://131002.net/blake/blake.pdf>

Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme^{*}

Joppe W. Bos¹, Kristin Lauter¹, Jake Loftus², and Michael Naehrig¹

¹ Microsoft Research

{jbos,klauter,mnaehrig}@microsoft.com

² University of Bristol

loftus@cs.bris.ac.uk

Abstract. In 1996, Hoffstein, Pipher and Silverman introduced an efficient lattice based encryption scheme dubbed **NTRUencrypt**. Unfortunately, this scheme lacks a proof of security. However, in 2011, Stehlé and Steinfeld showed how to modify **NTRUencrypt** to reduce security to standard problems in ideal lattices. In 2012, López-Alt, Tromer and Vaikuntanathan proposed a fully homomorphic scheme based on this modified system. However, to allow homomorphic operations and prove security, a non-standard assumption is required. In this paper, we show how to remove this non-standard assumption via techniques introduced by Brakerski and construct a new fully homomorphic encryption scheme from the Stehlé and Steinfeld version based on standard lattice assumptions and a circular security assumption. The scheme is scale-invariant and therefore avoids modulus switching and the size of ciphertexts is one ring element. Moreover, we present a practical variant of our scheme, which is secure under stronger assumptions, along with parameter recommendations and promising implementation results. Finally, we present an approach for encrypting larger input sizes by extending ciphertexts to several ring elements via the CRT on the message space.

1 Introduction

Fully homomorphic encryption (FHE) is a powerful form of encryption which allows an untrusted server to carry out arbitrary computation on encrypted data on behalf of a client. Introduced in [21] by Adleman, Dertouzos and Rivest, the problem of constructing a scheme which can evaluate any function on encrypted data remained open until 2009, when Gentry constructed an FHE scheme based on ideal lattices [10]. Gentry's scheme effectively laid down a blueprint for constructing FHE schemes and paved the way for many further constructions [26,3,4,6,5,24,20,11,9]. The main focus of the cryptologic research community has been on improving the efficiency of FHE and basing its security on standard assumptions.

^{*} Most of this work was done while the third author was an intern in the Cryptography Research group at Microsoft Research.

Recently, López-Alt et al. [16] proposed a (multi-key) FHE scheme based on the work by Stehlé and Steinfeld [25] in which a provably secure version of NTRUEncrypt [13] is presented with security based on standard problems in ideal lattices. Unfortunately, the FHE scheme from [16] needs to make an additional assumption relating to the uniformity of the public key, the so-called decisional small polynomial ratio (DSPR) assumption, to allow homomorphic operations and remain semantically secure. We show how to avoid this additional assumption and transform the results from [25] into a fully homomorphic encryption scheme based on standard lattice assumptions only. This is achieved by limiting noise growth during homomorphic operations via a tensoring technique recently introduced by Brakerski [3]. Besides this theoretical advantage, our scheme has other attractive properties. Firstly, this new scheme is *scale-invariant* in the sense of [3], i.e. it avoids the modulus-switching technique of Brakerski, Gentry and Vaikuntanathan [4]. Secondly, we keep the property of the scheme in [16] that a ciphertext consists of only a single ring element as opposed to the two or more ring elements for schemes based purely on the (ring) learning with errors (RLWE) assumption [17]. This decreases the ciphertext size since parameters are comparable in both settings. Finally, we present a technique to increase the size of the input space by working with separate, small plaintext moduli in ciphertexts of multiple ring elements, which are later combined via the Chinese remainder theorem into a larger plaintext modulus. For some applications, this additional flexibility to increase the message space without changing parameters at the cost of increasing ciphertext size can prove especially useful.

Our main contribution is an FHE scheme based on the schemes by Stehlé and Steinfeld [25] and López-Alt et al. [16] that does not need the DSPR assumption and thus is secure under the RLWE and circular security assumptions only. The public key in both schemes is the fraction $h = gf^{-1} \pmod q$ of two polynomials f and g in a cyclotomic polynomial ring modulo an integer modulus q that are sampled from a discrete Gaussian distribution. The DSPR assumption is the assumption that such a fraction is indistinguishable from uniform random in the ring modulo q . Stehlé and Steinfeld show that this assumption holds if the Gaussian is wide enough. Unfortunately, the scheme by López-Alt et al. cannot use such a wide Gaussian for key generation. Since the norms of f and g contribute to the noise growth during homomorphic multiplication, using a wide enough Gaussian means that the scheme is not guaranteed to be capable of doing even a single multiplication. We solve this problem by using decompositions and Brakerski's [3] tensoring technique. During the homomorphic multiplication procedure which includes a key switching step, we decompose the polynomial f into its bit decomposition, i.e. into a vector of polynomials with binary coefficients. This technique replaces the ring product of polynomials by a scalar product of binary decomposition vectors with vectors of polynomials multiplied by powers of 2 modulo q . The noise growth introduced in such a scalar product is bounded by a polynomial in $\log(q)$ and the degree of f , replacing the square of the norm of f in the bounds of the original scheme. Noise growth is much smaller now and it is possible to sample from a wide Gaussian to ensure the Stehlé-Steinfeld

conditions. As noted in Appendix A.1 of [16], any FHE scheme is inherently a multi-key scheme for a constant number of parties, but this construction is rather inefficient. The original scheme in [16], however, directly yields the multi-key property for a non-constant number of parties, which is much more efficient. Our scheme is not a multi-key scheme in that sense because decryption of a multi-key ciphertext would require a multiplication by the product of all keys that were involved in the generation of the ciphertext. With keys generated in the setting of Stehlé and Steinfeld, multiplying by a product of only two keys would already lead to a noise overflow, making it impossible to decrypt correctly.

The second part of the paper describes a more practical variant of the above scheme, along with details on parameter selection and implementation results. The price for obtaining security without the DSPR assumption in the above scheme lies in a large evaluation key and a complicated key switching procedure, both of which are a consequence of using the tensoring approach. Any possibility, which we are aware of, to avoid the tensor products, leads to an increase in the noise bounds that makes it necessary to reintroduce the DSPR assumption. However, if one is willing to make this assumption, there are several efficiency advantages and possible trade-offs as shown in our more practical variant. This variant keeps the general characteristics of the scheme, but simplifies key switching and avoids tensor products. A much shorter evaluation key can be achieved by using base- w instead of base-2 decompositions for a $w > 2$, e.g. $w = 2^{32}$. This increases noise growth, but ensures that the evaluation key contains only a few ring elements. Since the key switching is the main cost in homomorphic multiplication, the choice of w provides an important trade-off between homomorphic capability and multiplication efficiency. We also point out that it is possible to weaken the DSPR assumption by allowing the polynomial g to be sampled from a wider Gaussian than f . The proofs of most lemmas and theorems are given in the full version of this paper [2].

2 Preliminaries

In this section, we define all basic notation that is needed in the paper. The most important structure is the ring R . Let d be a positive integer and define $R = \mathbb{Z}[X]/(\Phi_d(X))$ as the ring of polynomials with integer coefficients modulo the d -th cyclotomic polynomial $\Phi_d(X) \in \mathbb{Z}[X]$. The degree of Φ_d is $n = \varphi(d)$, where φ is Euler's totient function. The elements of R can be uniquely represented by all polynomials in $\mathbb{Z}[X]$ of degree less than n . Arithmetic in R is arithmetic modulo $\Phi_d(X)$, which is implicit whenever we write down terms or equalities involving elements in R . An arbitrary element $a \in R$ can be written as $a = \sum_{i=0}^{n-1} a_i X^i$ with $a_i \in \mathbb{Z}$ and we identify a with its vector of coefficients $(a_0, a_1, \dots, a_{n-1})$. In particular, a can be viewed as an element of the \mathbb{R} -vector space \mathbb{R}^n . We choose the maximum norm on \mathbb{R}^n to measure the size of elements in R . The maximum norm of a is defined as $\|a\|_\infty = \max_i \{|a_i|\}$.

When multiplying two elements $g, h \in R$, the norm of their product gh expands with respect to the individual norms of g and h . The maximal norm

expansion that can occur is $\delta = \sup \{\|g \cdot h\|_\infty / (\|g\|_\infty \|h\|_\infty) : g, h \in R\}$, which is a ring constant. When d is a power of 2 and thus $\Phi_d(X) = X^n + 1$, we have $\delta = n$ [10, Section 3.4]. To keep the exposition more general, we do not restrict to this special case and work with general δ in most of what follows.

Let χ be a probability distribution on R . We assume that we can efficiently sample elements from R according to χ , and we use the standard notation $a \leftarrow \chi$ to denote that $a \in R$ is sampled from χ . The distribution χ on R is called B -bounded for some $B > 0$ if for all $a \leftarrow \chi$ we have $\|a\|_\infty < B$, i.e. a is B -bounded (see [4, Def. 3] and [16, Def. 3.1 and 3.2]). Let us introduce a specific example of a distribution on R . First, define the discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ with mean 0 and standard deviation σ over the integers, which assigns a probability proportional to $\exp(-\pi|x|^2/\sigma^2)$ to each $x \in \mathbb{Z}$. When d is a power of 2 and $\Phi_d(X) = X^n + 1$, we can take χ to be the spherical discrete Gaussian $\chi = \mathcal{D}_{\mathbb{Z}^n,\sigma}$, where each coefficient of the polynomial is sampled according to the one-dimensional distribution $D_{\mathbb{Z},\sigma}$ (see [17] for more details and why $\chi = \mathcal{D}_{\mathbb{Z}^n,\sigma}$ is the right choice in that case). The distribution χ is used in many fully homomorphic encryption schemes based on RLWE to sample random error polynomials that have small coefficients with high probability. Such polynomials are a significant part of the noise terms used in the encryption process. To deduce meaningful bounds on noise size and noise growth during homomorphic operations, we assume that the distribution we are working with is B -bounded for some B . For the discrete Gaussian, this is a reasonable assumption since sampled elements tend to be small with high probability. By rejecting samples with norm larger than B , we can sample from a truncated Gaussian distribution that is statistically close to the true discrete Gaussian if B is chosen large enough. For example, if we take $B = 6\sigma$, all samples are B -bounded with very high probability [18, Lemma 4.4].

Although the principal object of interest for our scheme is the ring R , and all polynomials that we deal with are considered to be elements of R , we often reduce polynomial coefficients modulo an integer modulus q . We denote the map that reduces an integer x modulo q and uniquely represents the result by an element in the interval $(-q/2, q/2]$ by $[\cdot]_q$. We extend this map to polynomials in $\mathbb{Z}[X]$ and thus also to elements of R by applying it to their coefficients separately, i.e. $[\cdot]_q : R \rightarrow R$, $a = \sum_{i=0}^{n-1} a_i X^i \mapsto \sum_{i=0}^{n-1} [a_i]_q X^i$. Furthermore, we extend this notation to vectors of polynomials by applying it to the entries of the vectors separately. Sometimes we reduce an integer modulo q and uniquely represent the result by an element in $[0, q)$. In this case, we write $r_q(x)$ to mean the reduction of x into $[0, q)$. A polynomial $f \in R$ is invertible modulo q if there exists a polynomial $f^{-1} \in R$ such that $ff^{-1} = \tilde{f}$, where $\tilde{f}(X) = \sum_i a_i X^i$ with $a_0 = 1 \pmod q$ and $a_j = 0 \pmod q$ for all $j \neq 0$. Our homomorphic encryption scheme uses two different moduli. In addition to a modulus q that is used to reduce the coefficients of the elements that represent ciphertexts, there is a second modulus $t < q$ that determines the message space R/tR , i.e. messages are polynomials in R modulo t . We make frequent use of the quantity $\Delta = [q/t]$ and it is readily verified that $q - r_t(q) = \Delta \cdot t$.

In [3], functions called BitDecomp and PowersOfTwo are used. We slightly generalize these to an arbitrary base and describe our notation next. Fix a positive integer $w > 1$ that is used to represent integers in a radix- w system. Let $\ell_{w,q} = \lceil \log_w(q) \rceil + 2$, then a non-negative integer $z < q$ can be written as $\sum_{i=0}^{\ell_{w,q}-2} z_i w^i$ where the z_i are integers such that $0 \leq z_i < w$. If z is an integer in the interval $(-q/2, q/2]$, it can be written uniquely as $\sum_{i=0}^{\ell_{w,q}-1} z_i w^i$ with $z_i \in (-w/2, w/2]$. With this, an element $x \in R$ with coefficients in $(-q/2, q/2]$ can be written as $\sum_{i=0}^{\ell_{w,q}-1} x_i w^i$, where $x_i \in R$ with coefficients in $(-w/2, w/2]$. Since then $x_i = [x_i]_w$, we write $x = \sum_{i=0}^{\ell_{w,q}-1} [x_i]_w w^i$ to make clear that the norm of the coefficient polynomials x_i is at most $w/2$. With this notation, define

$$D_{w,q} : R \rightarrow R^{\ell_{w,q}}, \quad x \mapsto ([x_0]_w, [x_1]_w, \dots, [x_{\ell_{w,q}-1}]_w) = ([x_i]_w)_{i=0}^{\ell_{w,q}-1},$$

this function for $w = 2$ is called BitDecomp in [3]. We define a second function

$$P_{w,q} : R \rightarrow R^{\ell_{w,q}}, \quad x \mapsto ([x]_q, [xw]_q, \dots, [xw^{\ell_{w,q}-1}]_q) = ([xw^i]_q)_{i=0}^{\ell_{w,q}-1},$$

which is called PowersOfTwo in [3] for $w = 2$. For any two $x, y \in R$, we see that the scalar product of the vectors $D_{w,q}(x)$ and $P_{w,q}(y)$ is the same as the product xy modulo q , because

$$\langle D_{w,q}(x), P_{w,q}(y) \rangle = \sum_{i=0}^{\ell_{w,q}-1} [x_i]_w [y w^i]_q \equiv y \sum_{i=0}^{\ell_{w,q}-1} [x_i]_w w^i \equiv xy \pmod{q}.$$

Note that when $\|f\|_\infty < B$ for some $B < q$, then only the $\ell_{w,B} := \lceil \log_w(B) \rceil + 2$ least significant polynomials in $D_{w,q}(f)$ can be non-zero. We use the tensor product of two vectors in the usual way, i.e. for a positive integer ℓ and two vectors $a, b \in R^\ell$, the tensor $a \otimes b \in R^{\ell^2}$ is the concatenation of the $a_i b$ for $i \in \{1, 2, \dots, \ell\}$. We extend the functions $D_{w,q}$ and $P_{w,q}$ to vectors. For $v = (v_1, v_2, \dots, v_\ell) \in R^\ell$ denote the vector $(D_{w,q}(v_1), \dots, D_{w,q}(v_\ell)) \in R^{\ell \cdot \ell_{w,q}}$ by $D_{w,q}(v)$, likewise we extend $P_{w,q}$.

Several operations in the scheme require scaling by rational numbers such that the resulting polynomials do not necessarily belong to R but instead have rational coefficients. In that case, a rounding procedure is applied to get back to integer coefficients. The usual rounding of a rational number a to the nearest integer is denoted by $\lfloor a \rfloor$.

The Ring Learning With Errors (RLWE) Problem. Our scheme relies on the hardness of the (decisional) ring learning with errors problem, which was first introduced by Lyubashevsky, Peikert and Regev [17].

Definition 1 (Decision-RLWE). Given a security parameter λ , let d and q be integers depending on λ , let $R = \mathbb{Z}[X]/(\Phi_d(X))$ and let $R_q = R/qR$. Given a distribution χ over R_q that depends on λ , the Decision-RLWE $_{d,q,\chi}$ problem is to distinguish the following two distributions. The first distribution consists of pairs (a, u) , where $a, u \leftarrow R_q$ are drawn uniformly at random from R_q . The second

distribution consists of pairs of the form $(a, a \cdot s + e)$. The element $s \leftarrow R_q$ is drawn uniformly at random and is fixed for all samples. For each sample, $a \leftarrow R_q$ is drawn uniformly at random, and $e \leftarrow \chi$. The Decision-RLWE $_{d,q,\chi}$ assumption is that the Decision-RLWE $_{d,q,\chi}$ problem is hard.

In [17], it was shown that the hardness of RLWE can be established by a quantum reduction to worst-case shortest vector problems in ideal lattices over the ring R , see also [4, Thm. 2]. It is known that the *search* variant of RLWE $_{d,q,\chi}$, in which we are required to explicitly find the secret s given an RLWE $_{d,q,\chi}$ instance, is equivalent to the decision problem [17]. There are a number of variants of RLWE which are as hard as RLWE, for example we can restrict the sampling of a and e to invertible elements only [25]. And we can also choose s from χ without incurring any loss of security [1].

The Decisional Small Polynomial Ratio (DSPR) Problem. In [16], López-Alt, Tromer and Vaikuntanathan introduced the decisional small polynomial ratio problem. They describe a multi-key fully homomorphic encryption scheme with security based on the assumption that the DSPR problem is hard in the ring R_q where $R = \mathbb{Z}[x]/(x^n + 1)$ for n a power of 2 and $t = 2$. We state a more general form of the problem for any cyclotomic ring $R = \mathbb{Z}[x]/(\Phi_d(x))$ and general $1 < t < q$. Let $h = tg/f \pmod{q}$ where $f = 1 + tf'$ and $f', g \leftarrow \chi$ where χ is a truncated Gaussian distribution. In [16], the problem of distinguishing such an element h from a uniformly random element of $R_q = R/qR$ was formalized as the DSPR problem. Assuming the hardness of DSPR and RLWE, the scheme in [16] is secure. To state the problem, define the following: for a distribution χ on R_q and $z \in R_q$ we define $\chi_z = \chi + z$ to be the distribution shifted by z . Also, let R_q^\times be the set of all invertible elements in R_q .

Definition 2 (DSPR). For security parameter λ , let d and q be integers, let $R = \mathbb{Z}[X]/(\Phi_d(X))$ and $R_q = R/qR$ and let χ be a distribution over R_q , all depending on λ . Let $t \in R_q^\times$ be invertible in R_q , $y_i \in R_q$ and $z_i = -y_i t^{-1} \pmod{q}$ for $i \in \{1, 2\}$. The DSPR $_{d,q,\chi}$ problem is to distinguish elements of the form $h = a/b$ where $a \leftarrow y_1 + t \cdot \chi_{z_1}$, $b \leftarrow y_2 + t \cdot \chi_{z_2}$ from uniformly random elements of R_q . The DSPR $_{d,q,\chi}$ assumption is that the DSPR $_{d,q,\chi}$ problem is hard.

Theorem 4.1 in the full version of [25] shows that DSPR $_{d,q,\chi}$ is hard when the χ_{z_i} are shifted versions of a discrete Gaussian distributions χ which is $\mathcal{D}_{\mathbb{Z}^n, \sigma}$ restricted to R_q^\times for a large enough deviation σ . For convenience, we state the theorem in the full version of this paper [2, Appendix A]. A discrete Gaussian on R_q^\times can be obtained from a discrete Gaussian on R_q by rejecting non-invertible elements.

3 Basic Scheme

In this section, we describe the basic public key encryption scheme that is the foundation for the leveled schemes of the next sections. The scheme is parameterized by a modulus q and a plaintext modulus $1 < t < q$. Ciphertexts are elements

of $R = \mathbb{Z}[X]/(\Phi_d(X))$ and plaintexts are elements of R/tR (see Section 2). Secret keys and errors are generated from different distributions, for example Gaussian distributions of different width. The secret key is derived from the distribution χ_{key} , and errors are sampled from the distribution χ_{err} . We use “Regev-style” encryption as in [3] and [9]. The scheme consists of the following algorithms.

- **Basic.ParamsGen**(λ): Given the security parameter λ , fix a positive integer d that determines R , moduli q and t with $1 < t < q$, and distributions $\chi_{\text{key}}, \chi_{\text{err}}$ on R . Output $(d, q, t, \chi_{\text{key}}, \chi_{\text{err}})$.
- **Basic.KeyGen**($d, q, t, \chi_{\text{key}}, \chi_{\text{err}}$): Sample $f', g \leftarrow \chi_{\text{key}}$ and let $f = [tf' + 1]_q$. If f is not invertible modulo q , choose a new f' . Compute the inverse $f^{-1} \in R$ of f modulo q and set $h = [tgf^{-1}]_q$. Output the public and private key pair $(\text{pk}, \text{sk}) = (h, f) \in R^2$.
- **Basic.Encrypt**(h, m): The message space is R/tR . For a message $m + tR$, choose $[m]_t$ as its representative. Sample $s, e \leftarrow \chi_{\text{err}}$, and output the ciphertext $c = [\lfloor q/t \rfloor [m]_t + e + hs]_q \in R$.
- **Basic.Decrypt**(f, c): To decrypt a ciphertext c , compute

$$m = \left[\left[\frac{t}{q} \cdot [fc]_q \right] \right]_t \in R.$$

In the following, we often refer to a message as an element m in the ring R although the message space is R/tR , keeping in mind that encryption always takes place on the representative $[m]_t$ and that by decrypting, all that can be recovered is m modulo t .

Correctness. The following lemma states conditions for a ciphertext c such that the decryption algorithm outputs the message m that was originally encrypted.

Lemma 1. *Let q, t , and $\Delta = \lfloor q/t \rfloor$ be as above and let $c, f, m \in R$. If there exists $v \in R$ such that*

$$fc = \Delta[m]_t + v \pmod{q} \text{ and } \|v\|_\infty < (\Delta - r_t(q))/2,$$

then $\text{Basic.Decrypt}(f, c) = [m]_t$, i.e. c decrypts correctly under the secret key f .

Of course, for any given c, f and m , there always exists a $v \in R$ such that $fc = \Delta[m]_t + v \pmod{q}$. But only a v of small norm allows one to recover $[m]_t$ from c . Since we are always free to vary v modulo q , i.e. to add any multiple of q to it, we choose v to be the canonical element $[v]_q$. This means that we choose v with the smallest possible norm among all polynomials that satisfy the equation. We call this specific v the *inherent noise in c with respect to m and f* . The previous lemma says that if the inherent noise in a ciphertext is small enough, then decryption works correctly.

Inherent Noise in Initial Ciphertexts. The following lemma derives a bound on the inherent noise in a freshly encrypted ciphertext output by **Basic.Encrypt**, assuming bounds B_{key} on the key and B_{err} on the error distributions. Note that since $f', g \leftarrow \chi_{\text{key}}$ we have $\|f'\|_\infty, \|g\|_\infty < B_{\text{key}}$ and it follows that $\|tg\|_\infty < tB_{\text{key}}$ and $\|f\|_\infty = \|1 + tf'\|_\infty < tB_{\text{key}}$ since $t \geq 2$.

Lemma 2. *Let the key and error distributions be B_{key} -bounded and B_{err} -bounded, respectively. Given $m \in R$, a public key $h = [tgf^{-1}]_q \in R$ with secret key $f = [1 + tf']_q$, $f', g \leftarrow \chi_{\text{key}}$, and let $c = \text{Basic.Encrypt}(h, m)$. There exists $v \in R$ such that $fc = \Delta[m]_t + v \pmod{q}$ and*

$$\|v\|_\infty < \delta t B_{\text{key}} \left(2B_{\text{err}} + \frac{1}{2}r_t(q) \right).$$

In particular, by Lemma 1, decryption works correctly if $2\delta t B_{\text{key}}(2B_{\text{err}} + \frac{1}{2}r_t(q)) + r_t(q) < \Delta$.

4 Levelled Homomorphic Scheme

In this section, we state our levelled homomorphic encryption scheme YASHE¹ based on the Basic scheme from the previous section. We then analyze the homomorphic operations and deduce bounds on the noise growth that occurs during these operations.

- **YASHE.ParamsGen**(λ): Given the security parameter λ , output the parameters $(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$, where $(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}) \leftarrow \text{BasicParamsGen}(\lambda)$ and $w > 1$ is an integer.
- **YASHE.KeyGen**($d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w$): Compute

$$h, f \leftarrow \text{Basic.KeyGen}(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}).$$

Sample $e, s \leftarrow \chi_{\text{err}}^{\ell^3_{w,q}}$, compute

$$\gamma = [f^{-1}P_{w,q}(D_{w,q}(f) \otimes D_{w,q}(f)) + e + h \cdot s]_q \in R^{\ell^3_{w,q}},$$

and output $(\text{pk}, \text{sk}, \text{evk}) = (h, f, \gamma)$.

- **YASHE.Encrypt**(pk, m): Encrypt $m \in R$ by $c \leftarrow \text{Basic.Encrypt}(\text{pk}, m) \in R$.
- **YASHE.Decrypt**(sk, c): Output the message $m \leftarrow \text{Basic.Decrypt}(\text{sk}, c) \in R$.
- **YASHE.KeySwitch**($\tilde{c}_{\text{mult}}, \text{evk}$): Output $[\langle D_{w,q}(\tilde{c}_{\text{mult}}), \text{evk} \rangle]_q \in R$.
- **YASHE.Add**(c_1, c_2): Compute the addition of c_1 and c_2 as $c_{\text{add}} = [c_1 + c_2]_q$.
- **YASHE.Mult**(c_1, c_2, evk): Compute

$$\tilde{c}_{\text{mult}} = \left[\left[\frac{t}{q} P_{w,q}(c_1) \otimes P_{w,q}(c_2) \right] \right]_q \in R^{\ell^2_{w,q}},$$

and output $c_{\text{mult}} = \text{YASHE.KeySwitch}(\tilde{c}_{\text{mult}}, \text{evk})$.

Since encryption and decryption are the same as in the Basic scheme from Section 3, the correctness bound does not change and Lemmas 1 and 2 hold for YASHE as well. Next, we analyze the homomorphic operations **YASHE.Add** and **YASHE.Mult**.

¹ Yet Another Somewhat Homomorphic Encryption scheme.

Homomorphic Addition. Given two ciphertexts $c_1, c_2 \in R$, which encrypt two messages m_1, m_2 with inherent noise terms v_1, v_2 , their sum *modulo* q , $c_{\text{add}} = [c_1 + c_2]_q$, encrypts the sum of the messages *modulo* t , $[m_1 + m_2]_t$. Indeed, we can write $[m_1]_t + [m_2]_t = [m_1 + m_2]_t + tr_{\text{add}}$ for some $r_{\text{add}} \in R$ with $\|r_{\text{add}}\|_\infty \leq 1$. Since

$$\begin{aligned} f[c_1 + c_2]_q &= fc_1 + fc_2 = \Delta([m_1]_t + [m_2]_t) + (v_1 + v_2) \\ &= \Delta([m_1 + m_2]_t + tr_{\text{add}}) + (v_1 + v_2) \pmod{q}, \end{aligned}$$

we obtain $f[c_1 + c_2]_q = \Delta[m_1 + m_2]_t + (v_1 + v_2 - r_t(q)r_{\text{add}}) \pmod{q}$ because $\Delta t \equiv -r_t(q) \pmod{q}$. This means that the size of the inherent noise v_{add} of c_{add} is bounded by

$$\|v_{\text{add}}\|_\infty \leq \|v_1\|_\infty + \|v_2\|_\infty + r_t(q). \quad (1)$$

Up to the term $r_t(q) < t$, the inherent noise terms are added during homomorphic addition.

Homomorphic Multiplication. The homomorphic multiplication operation is divided into two parts. The first part describes a basic procedure to obtain an intermediate ciphertext that encrypts the product $[m_1 m_2]_t$ modulo t of two messages m_1 and m_2 . However, the intermediate ciphertext can not be decrypted with `Basic.Decrypt` using the secret key f . The second part performs a procedure which allows a public transformation of this intermediate ciphertext to a ciphertext that can be decrypted with f . This latter procedure was introduced in [6] in the form of relinearization and was later expanded in [4] into a method called key switching, which transforms a ciphertext decryptable under one secret key to one decryptable under any other secret key. For our analysis, we assume that χ_{key} and χ_{err} are B_{key} - and B_{err} -bounded, respectively. Even if we work with unbounded Gaussian distributions, this is a valid assumption since elements drawn from either distribution have bounded norm for suitable bounds with high probability. The deduction of noise bounds mostly follows the basic multiplication section of [9], since ciphertexts and the decryption algorithm in YASHE have a very similar structure to those in [9].

First Step. Let $c_1, c_2 \in R$ be ciphertexts that encrypt messages $m_1, m_2 \in R$. In the first step of the homomorphic multiplication operation, we compute

$$\tilde{c}_{\text{mult}} = \left[\left[\frac{t}{q} P_{w,q}(c_1) \otimes P_{w,q}(c_2) \right] \right].$$

The following theorem shows that $\langle \tilde{c}_{\text{mult}}, D_{w,q}(f) \otimes D_{w,q}(f) \rangle = \Delta[m_1 m_2]_t + \tilde{v}_{\text{mult}} \pmod{q}$, and it provides a bound for the size of \tilde{v}_{mult} . Thus, \tilde{c}_{mult} can be viewed as an encryption of $[m_1 m_2]_t$ under $D_{w,q}(f) \otimes D_{w,q}(f)$ if the inherent noise term \tilde{v}_{mult} is small enough.

Theorem 1 (Multiplication Noise). *Let $c_1, c_2 \in R$ be ciphertexts encrypting $m_1, m_2 \in R$, decryptable with the secret key f . Let $v_1, v_2 \in R$ be the inherent noise terms in c_1, c_2 and let $V > 0$ such that $\|v_i\|_\infty \leq V < \Delta/2$,*

$i \in \{1, 2\}$. Let \tilde{c}_{mult} be the intermediate ciphertext in YASHE.Mult , and let $\ell_{w,tB_{\text{key}}} = \lceil \log_w(tB_{\text{key}}) \rceil + 2$. Then $\langle \tilde{c}_{\text{mult}}, D_{w,q}(f) \otimes D_{w,q}(f) \rangle = \Delta[m_1 m_2]_t + \tilde{v}_{\text{mult}} \pmod{q}$ where

$$\|\tilde{v}_{\text{mult}}\|_{\infty} < \delta t(2 + \delta \ell_{w,tB_{\text{key}}} w)V + \frac{\delta t^2}{2}(3 + \delta \ell_{w,tB_{\text{key}}} w) + \frac{1}{8}(\delta \ell_{w,tB_{\text{key}}} w)^2 + \frac{1}{2}.$$

Starting with two ciphertexts at a given inherent noise level, the first step of the multiplication increases the inherent noise level by a multiplicative factor of roughly $\delta^2 t \ell_{w,tB_{\text{key}}} w$ and an additive term of $\frac{\delta^2}{2} \ell_{w,tB_{\text{key}}} w(t^2 + \frac{1}{4} \ell_{w,tB_{\text{key}}} w)$.

Key Switching. The second part in the homomorphic multiplication procedure is a key switching step, which transforms the ciphertext \tilde{c}_{mult} into a ciphertext c_{mult} that is decryptable under the original secret key f . We use the evaluation key

$$\text{evk} = [f^{-1} P_{w,q}(D_{w,q}(f) \otimes D_{w,q}(f)) + \mathbf{e} + h \cdot \mathbf{s}]_q,$$

output by YASHE.KeyGen where $\mathbf{e}, \mathbf{s} \leftarrow \chi_{\text{err}}^{\ell_{w,q}^3}$ are vectors of polynomials sampled from the error distribution χ_{err} and $[\cdot]_q$ is applied to each coefficient of the vector. Note that this key is a vector of quasi-encryptions of $f^{-1} P_{w,q}(D_{w,q}(f) \otimes D_{w,q}(f))$ that depend on the secret key f , under its corresponding public key and that it is made public because it is needed for the homomorphic multiplication operation. Therefore, we need to make a circular security assumption, namely that the scheme is still secure even given that evk is publicly known (see Section 4.2). The following lemma deduces a bound on the noise caused by the key switching procedure and states an overall bound on the noise growth during a single homomorphic multiplication operation.

Lemma 3. *Let notation be as in Theorem 1 and as above. In particular, let \tilde{c}_{mult} be the intermediate ciphertext in YASHE.Mult with inherent noise term \tilde{v}_{mult} . Let evk be the evaluation key and $c_{\text{mult}} = \text{YASHE.KeySwitch}(\tilde{c}_{\text{mult}}, \text{evk})$. Then $f c_{\text{mult}} = \Delta[m_1 m_2]_t + v_{\text{mult}} \pmod{q}$, where*

$$\|v_{\text{mult}}\|_{\infty} < \|\tilde{v}_{\text{mult}}\|_{\infty} + \delta^2 t \ell_{w,q}^3 w B_{\text{err}} B_{\text{key}}.$$

Theorem 1 and Lemma 3 give an overall upper bound on the noise growth during a homomorphic multiplication. This clearly dominates the noise growth for homomorphic addition.²

4.1 Correctness

This section discusses the correctness of YASHE and shows that it is a leveled homomorphic encryption scheme. We state correctness by giving an asymptotic bound on the number of multiplicative levels in an arithmetic circuit that can be correctly evaluated. For this, we concretely focus on a parameter setting such

² As noted in [3] the number of elements in $D_{w,q}(f) \otimes D_{w,q}(f)$ can be reduced from $\ell_{w,q}^2$ to $\binom{\ell_{w,q}}{2}$ which correspondingly reduces the number of ring elements in evk .

that the assumptions of the theorem by Stehlé and Steinfeld (see [2, Appendix A]) hold. This means that the DSPR problem is hard in R_q . We therefore fix the following parameters: let d be a power of 2, $n = \varphi(d)$, $\epsilon \in (0, 1)$, $k \in (1/2, 1)$ and let $q = 2^{d^\epsilon}$ be a prime such that $\Phi_d(X) = X^n + 1$ splits into n irreducible factors modulo q . Let χ_{key} be a discrete Gaussian distribution on R_q with deviation $\sigma_{\text{key}} \geq d\sqrt{\log(8dq)} \cdot q^k$, and let χ_{err} be an asymptotically $\omega(\sqrt{d\log(d)})$ -bounded Gaussian distribution on R where d tends to infinity. Finally, we fix $w = 2$ and $t = 2$, but note that similar results hold for general w, t – this restriction is merely for the purpose of exposition.

Theorem 2 (Correctness of YASHE). *For the parameter choices above, YASHE can evaluate any circuit of depth*

$$L = \mathcal{O}\left(\frac{(1-k)\log(q)}{\log(\log(q)) + \log(d)}\right).$$

4.2 Security

To prove security of YASHE, we need to assume that IND-CPA security can be maintained even when an adversary has access to elements of the evaluation key evk . Due to the way we construct evk it is not sufficient to simply replace f by L distinct secret keys f_i , as has been done in previous works – a specific assumption is still required. This is a form of key dependent message security, for the family of functions defining the evaluation key. Under this “circular security” assumption, the IND-CPA security of YASHE follows from the IND-CPA security of the scheme **Basic** described in Section 3 and the RLWE assumption.

Theorem 3 (Security of YASHE). *The scheme YASHE is IND-CPA secure under the $\text{RLWE}_{d,q,\chi_{\text{err}}}$ assumption and the assumption that the scheme remains IND-CPA secure, even when an adversary has access to evk output by $\text{YASHE.KeyGen}(d, q, 2, \chi_{\text{key}}, \chi_{\text{err}}, 2)$.*

Proof. Since $\sigma_{\text{key}} \geq d\sqrt{\log(8dq)} \cdot q^k$ for some $k > 1/2 + \nu$ with $\nu > 0$, the conditions of [2, Theorem 7] (see also [25]) are satisfied. Hence the public key is indistinguishable from a uniform element of R_q^\times . It follows from [25] that the scheme **Basic** is IND-CPA secure under the $\text{RLWE}_{d,q,\chi_{\text{err}}}$ assumption in R_q . Under the circular security assumption outlined above, the IND-CPA security of YASHE follows. \square

For the proof of Theorem 3, we only need parameters that satisfy the assumptions in [2, Theorem 7]. For the parameters outlined at the beginning of this subsection, the RLWE assumption is believed to be hard based on standard worst-case lattice problems.

4.3 From Leveled to Fully Homomorphic Encryption

In [10], Gentry showed how a fully homomorphic scheme can be obtained from a leveled homomorphic scheme supporting computation of circuits of sufficient

depth. If a scheme can evaluate its own decryption circuit and one additional multiplication, then that scheme can be converted to a fully homomorphic scheme. The only caveat is that we have to make an additional assumption: to execute the bootstrapping procedure, it is necessary to augment the public key with encryptions $\text{YASHE.Encrypt}(\text{pk}, \text{sk}[j])$ of the bits of the secret key, under its corresponding public key. Similarly to the assumption on the evaluation key, we need to make an additional assumption that including encryptions of bits of the secret key does not affect security.

To achieve a fully homomorphic scheme, we simply view the decryption circuit as a circuit computed on the bits of the secret key at a ciphertext c we wish to refresh. The noise in the resulting *fresh* ciphertext will be of fixed size depending on the noise in the encryptions of the bits of the secret key. In the full version [2, (Lemma 6, Theorem 8)] we show that YASHE can be bootstrapped to a fully homomorphic scheme.

5 A More Practical Variant of the Scheme

In this section, we propose a more practical variant YASHE' of YASHE. The difference to YASHE lies in the homomorphic multiplication procedure. In YASHE' , an intermediate ciphertext is simply a single polynomial while it is a vector of polynomials in YASHE. This results in an evaluation key that consists of only $\ell_{w,q}$ polynomials instead of $\ell_{w,q}^3$ for YASHE and thus in a simpler key switching procedure. We now state the scheme and discuss the noise growth during the simplified homomorphic multiplication operation $\text{YASHE}'.\text{Mult}$.

- $\text{YASHE}'.\text{ParamsGen}(\lambda)$: Output $(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}) \leftarrow \text{BasicParamsGen}(\lambda)$.
- $\text{YASHE}'.\text{KeyGen}(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$: Compute

$$h, f \leftarrow \text{Basic.KeyGen}(d, q, t, \chi_{\text{key}}, \chi_{\text{err}}).$$

Sample $e, s \leftarrow \chi_{\text{err}}^{\ell_{w,q}}$, compute $\gamma = [P_{w,q}(f) + e + h \cdot s]_q \in R^{\ell_{w,q}}$. and output $(\text{pk}, \text{sk}, \text{evk}) = (h, f, \gamma)$.

- $\text{YASHE}'.\text{Encrypt}(\text{pk}, m)$: Encrypt $m \in R$ as $c \leftarrow \text{Basic.Encrypt}(\text{pk}, m) \in R$.
- $\text{YASHE}'.\text{Decrypt}(\text{sk}, c)$: Output the message $m \leftarrow \text{Basic.Decrypt}(\text{sk}, c) \in R$.
- $\text{YASHE}'.\text{KeySwitch}(\tilde{c}_{\text{mult}}, \text{evk})$: Output the ciphertext $[[D_{w,q}(\tilde{c}_{\text{mult}}), \text{evk}]]_q$.
- $\text{YASHE}'.\text{Add}(c_1, c_2)$: Output $c_{\text{add}} \leftarrow \text{YASHE}.\text{Add}(c_1, c_2) = [c_1 + c_2]_q$.
- $\text{YASHE}'.\text{Mult}(c_1, c_2, \text{evk})$: Output the ciphertext

$$c_{\text{mult}} = \text{YASHE}'.\text{KeySwitch}(\tilde{c}_{\text{mult}}, \text{evk}), \text{ where } \tilde{c}_{\text{mult}} = \left[\left[\begin{array}{c} t \\ -c_1 c_2 \end{array} \right] \right]_q.$$

For two ciphertexts $c_1, c_2 \in R$ that encrypt $m_1, m_2 \in R$, the intermediate ciphertext \tilde{c}_{mult} during homomorphic multiplication $\text{YASHE}'.\text{Mult}$ satisfies $f^2 \tilde{c}_{\text{mult}} = \Delta[m_1 m_2]_t + \tilde{v}_{\text{mult}} \pmod{q}$ as shown in the following theorem. This means that \tilde{c}_{mult} is an encryption of $[m_1 m_2]_t$ under f^2 . The theorem also provides an upper bound on the inherent noise term in the intermediate ciphertext. We assume that the error distribution χ_{err} is B_{err} -bounded and that the key distribution χ_{key} is B_{key} -bounded.

Theorem 4 (Multiplication Noise). *Let $c_1, c_2 \in R$ be ciphertexts encrypting $m_1, m_2 \in R$, which are decryptable with the secret key f . Let $v_1, v_2 \in R$ be the inherent noise terms in c_1, c_2 and let $V > 0$ such that $\|v_i\|_\infty \leq V < \Delta/2$, $i \in \{1, 2\}$. Let \tilde{c}_{mult} be the intermediate ciphertext in YASHE'.Mult.*

Then $f^2 \tilde{c}_{\text{mult}} = \Delta[m_1 m_2]_t + \tilde{v}_{\text{mult}} \pmod{q}$ where

$$\|\tilde{v}_{\text{mult}}\|_\infty < \delta t(4 + \delta t B_{\text{key}})V + \delta^2 t^2 B_{\text{key}}(B_{\text{key}} + t).$$

Key Switching. The key switching algorithm now transforms such an intermediate encryption into a ciphertext that can be decrypted with f itself. The evaluation key is $\text{evk} = [P_{w,q}(f) + \mathbf{e} + h \cdot \mathbf{s}]_q$, where $\mathbf{e}, \mathbf{s} \leftarrow \chi_{\text{err}}^{\ell_{w,q}}$ are vectors of polynomials sampled from the error distribution χ_{err} . Again, this key is a vector of quasi-encryptions of the secret key f under its corresponding public key. It is required for the homomorphic multiplication operation and is therefore made public. This means, we need to make a circular security assumption as for YASHE, namely that the scheme is still secure even given that evk is publicly known. The following lemma gives a bound on the key switching noise.

Lemma 4. *Let \tilde{c}_{mult} be the intermediate ciphertext in YASHE'.Mult. Its inherent noise term is denoted by \tilde{v}_{mult} . Let γ be the evaluation key from above and $c_{\text{mult}} = \text{YASHE'.KeySwitch}(\tilde{c}_{\text{mult}}, \gamma)$. Then $f c_{\text{mult}} = \Delta[m_1 m_2]_t + v_{\text{mult}} \pmod{q}$, where*

$$\|v_{\text{mult}}\|_\infty < \|\tilde{v}_{\text{mult}}\|_\infty + \delta^2 t \ell_{w,q} w B_{\text{err}} B_{\text{key}}.$$

5.1 Correctness and Security of YASHE'

In the following theorem, we give an explicit bound for correctness of a homomorphic evaluation of an arithmetic circuit in R/tR of multiplicative depth L that is organized in a leveled tree structure of multiplications without any additions. At each level all ciphertexts are assumed to have inherent noise terms of roughly the same size. The bounds that we obtain might be too large and could be significantly reduced for computations that involve more additions and less multiplications as well as multiplications of ciphertexts with imbalanced inherent noise terms. In favor of simplicity, we restrict to the above setting.

Theorem 5 (Correctness of YASHE'). *Let $\epsilon_1 = 4(\delta t B_{\text{key}})^{-1}$. The scheme YASHE' can correctly evaluate an arithmetic circuit consisting of L -levels of multiplications in R/tR on ciphertexts with inherent noise of size at most V that are arranged in a binary tree of L levels of multiplications if*

$$2(1 + \epsilon_1)^{L-1} \delta^{2L} t^{2L-1} B_{\text{key}}^L ((1 + \epsilon_1)tV + L(tB_{\text{key}} + t^2 + \ell_{w,q} w B_{\text{err}})) < \Delta - r_t(q).$$

Appendix K in [2] gives detailed bounds on the increase of the inherent noise terms in ciphertexts during homomorphic addition and multiplication. One can take these bounds to deduce overall bounds for the exact computation that is supposed to be carried out on encrypted data. The obtained bounds can then be used to deduce tailored parameters for the scheme to ensure correctness and

security for that particular setting, possibly resulting in more efficient parameters for the specific computation.

The security of YASHE' is based on the RLWE assumption and a circular security assumption similar to the one for YASHE . The price we pay for a simpler homomorphic multiplication operation lies in an additional security assumption. Since YASHE' only works for a much narrower key distribution that does not satisfy the requirements for applying the Stehlé and Steinfeld result ([25, Thm. 4.1]), security also relies on the Decisional Small Polynomial Ratio (DSPR) assumption, as stated in Section 2. In YASHE , this assumption could be avoided by making the scheme work with a key distribution as demanded by [25]. Following the same hybrid argument as in [16], one can prove that the scheme described in this section is secure under the DSPR assumption and the RLWE assumption (see [16, Section 3.3]). If a, b are two elements sampled from a Gaussian with very small standard deviation or from a different distribution that yields polynomials with very small coefficients only, the ratio $h = a/b$ can clearly not be uniform because the number of elements for a and b is too small and produces only a small number of values for h when compared to all elements in R_q . Still, a computationally bounded adversary might not be able to distinguish such a case from uniform randomly chosen h .

Theorem 6 (Security of YASHE'). *Let d be a positive integer, q and $t < q$ be two moduli, w be a fixed positive integer, and let χ_{key} and χ_{err} be distributions on R . The scheme YASHE' is IND-CPA secure under the $\text{RLWE}_{d,q,\chi_{\text{err}}}$ assumption, the $\text{DSPR}_{d,q,\chi_{\text{key}}}$ assumption, and the assumption that the scheme remains IND-CPA secure even when the evaluation key evk which is output by $\text{YASHE}'.\text{KeyGen}(d, q, t, \chi_{\text{key}}, \chi_{\text{err}})$ is known to the adversary.*

Remark 1. The $\text{DSPR}_{d,q,\chi_{\text{key}}}$ assumption can be replaced by a weaker assumption $\text{DSPR}_{d,q,\chi_f,\chi_g}$, where the elements f and g that are used for the public key $h = [tgf^{-1}]_q$ are sampled from distributions of different width with bounds B_f and B_g , respectively. This new assumption can be made weaker than the original assumption since the element g can be sampled from a much wider distribution than f . Introducing these two distributions means that the noise bound for the inherent noise in a fresh ciphertext is changed to $\delta t(B_{\text{err}}(B_f + B_g) + r_t(q)B_f/2)$. The proofs of the noise bounds for $\text{YASHE}'.\text{Mult}$ show that the bound B_g only influences the constant C_2 in Lemma 9 in the full version [2]. The contributions of B_g in the noise bounds for L levels of multiplications are merely a constant factor independent of L . Therefore, the scheme is still leveled homomorphic with the weaker assumption.

Remark 2. For YASHE' , since private keys are sampled with very small norm, the circular security assumption can be avoided in the usual way by providing a different public/private key pair (h_i, f_i) for each level i of multiplications for $0 \leq i \leq L$. The evaluation key has to be extended to L vectors $\gamma_i = [P_{w,q}(f_{i-1}^2) + e + h_i \cdot \mathbf{s}]_q$, $1 \leq i \leq L$, such that the key switching step $\text{YASHE}'.\text{KeySwitch}(\tilde{c}_{\text{mult}}, \text{evk}_i)$ transforms the intermediate ciphertext \tilde{c}_{mult} decryptable under f_{i-1}^2 (obtained from two ciphertexts at level $i - 1$) into one decryptable under f_i at level i .

Table 1. Parameters that guarantee security of $\lambda = 80$ bits against the distinguishing attack with advantage $\epsilon = 2^{-80}$. We fix $w = 2^{32}$, the key distribution is assumed to be bounded by $B_{\text{key}} = 1$, and we use $\sigma_{\text{err}} = 8$ and $B_{\text{err}} = 6\sigma_{\text{err}}$. Either for fixed sizes of q , we give the minimal degree n_{min} (left part), or for fixed dimension n , we give the maximal size $\log(q_{\text{max}})$ (right part). For each pair (q, n) according to the given sizes, and different values of t , correctness is guaranteed for at most L_{max} multiplicative levels.

$\lceil \log(q) \rceil$	n_{min}	t	L_{max}	n	$\log(q_{\text{max}})$	t	L_{max}
128	3329	2	3	2^{12}	157	2	4
		256	2			256	2
		1024	1			1024	2
192	5018	2	5	2^{13}	312	2	9
		256	3			256	6
		1024	3			1024	5
256	6707	2	7	2^{14}	622	2	19
		256	5			256	13
		1024	4			1024	11
512	13463	2	15	2^{15}	1243	2	37
		256	10			256	25
		1024	9			1024	23
1024	26974	2	31	2^{16}	2485	2	71
		256	21			256	50
		1024	19			1024	46

5.2 Parameters

In this section, we give suggestions for choosing concrete parameters which can be used as a guideline to instantiate practical schemes with varying complexity. There are multiple parameters one can adjust, so we restrict ourselves to a subset of choices which we think are most relevant. We consider two settings. In the first, we fix a specific size for the modulus q . This is interesting for instance when a fast modular multiplication implementation (in either hard- or software) is already available, and one prefers to use this to boost the scheme's performance. We fix different sizes for the modulus q starting from 128 bits up to 1024 bits. The other setting focuses on special-purpose polynomial arithmetic. Here, we fix the degree $n = \varphi(d)$ to be a power of 2 between 2^{12} and 2^{16} .

The parameters presented in Table 1 are obtained by following the security analysis of Lindner and Peikert [15] under the assumption that the results from [15] in the LWE setting carry over to the RLWE setting, and assuming that the assumptions in Section 5.1 hold. This analysis is similar to the ones from [12,9,14] and we refer to [12] for a more complete discussion of assumptions made in deriving parameters. Note that recent results by Chen and Nguyen [7] are considered to be more accurate for estimating the security of specific parameters using the simulation of the BKZ 2.0 algorithm for assessing the runtime of lattice basis reduction. Selecting parameters for YASHE' with this method is ongoing work at the time of writing this paper. However, it is expected that the parameters presented in this paper which are obtained by using the Lindner-Peikert method are more conservative than those obtained with the BKZ 2.0 simulation.

Next, we discuss in more detail the parameter selection recommendations made in Table 1. We use $B_{\text{key}} = 1$, in other words we are assuming that even when the polynomials f', g have coefficients in $\{-1, 0, 1\}$, the public key $h = [tgf^{-1}]_q$ is indistinguishable from uniform. The standard deviation of the error distribution is fixed at $\sigma_{\text{err}} = 8$; this is consistent with [19]. The high probability bound on the size of the coefficients of errors drawn from Gaussian distributions is chosen as $6\sigma_{\text{err}}$.

To distinguish with an advantage of ϵ in the RLWE problem, an adversary is required to find vectors of length at most $\alpha \cdot (q/\sigma)$ where $\alpha = \sqrt{\ln(1/\epsilon)/\pi}$. In our specific parameter examples, we use $\epsilon = 2^{-80}$, which results in $\alpha \approx 4.201$. We refer to [15] for a more complete description of a distinguishing attack and the precise lattices we are required to find short vectors in. Running Schnorr-Euchner's BKZ [22], the best known lattice reduction algorithm in practice, and its successor BKZ 2.0 [7] for security parameter λ (following [12] we use $\lambda = 80$) one expects to find vectors of length $2^{2\sqrt{n \log_2(q) \log_2(\delta_{\text{RHF}})}}$ in time $T_{\text{BKZ}} = 2^\lambda$ where δ_{RHF} is the so-called root Hermite factor. This latter quantity is the overwhelming factor determining the quality of the basis which can be achieved in a given time and is computed as in [15] from

$$\log_2(T_{\text{BKZ}}) = 1.8/\log_2(\delta_{\text{RHF}}) - 110.$$

It is currently infeasible to achieve a target root Hermite factor $\delta_{\text{RHF}} < 1.005$ [7]. To guarantee security, we require that the shortest vector obtained through lattice reduction is longer than a vector which could give an adversary a non-negligible advantage ϵ in the Ring-LWE distinguishing problem. This means that for security we thus require

$$\alpha \cdot q/\sigma < 2^{2\sqrt{n \log_2(q) \log_2(\delta_{\text{RHF}})}}.$$

For fixed parameters α and δ_{RHF} , this inequality provides bounds on the remaining parameters q , σ_{err} and n . Fixing σ_{err} too ($\sigma_{\text{err}} = 8$ here), we get a dependency between q and n that is expressed in the two settings discussed above as follows. When we fix q , we obtain a lower bound n_{min} for the dimension n to guarantee security against the distinguishing attack. For the example values for the sizes of q given in the first column of the left part of Table 1, we list this minimal degree in the second column. We used the worst case bound for a modulus q of that size. Vice versa, first fixing the degree n means that we get an upper bound q_{max} for q . We display the relation between n and the size $\log(q_{\text{max}})$ in the first two columns of the right part of Table 1.

For guaranteeing correctness, we use the noise bounds derived in the previous section. As mentioned in Section 2, when d is a power of 2 and thus $\Phi_d(X) = X^n + 1$, the expansion factor is $\delta = n$. Then, by Lemma 1 and Lemma 9 in the full version [2] we know that our scheme can correctly evaluate a depth L circuit as long as

$$(1 + \epsilon_1)^{L-1} n^{2L} t^{2L-1} B_{\text{key}}^L ((1 + \epsilon_1)tV + L(t(B_{\text{key}} + t) + \ell_{w,q}wB_{\text{err}}))$$

is less than $(\Delta - r_t(q))/2$, where $\epsilon_1 = 4(ntB_{\text{key}})^{-1}$ and $V = ntB_{\text{key}}(2B_{\text{err}} + r_t(q)/2)$ is the inherent noise of fresh ciphertexts by Lemma 2. For each row in

either the left or the right part of Table 1, we take the given values for q and n together with different values for t and check what is the maximum number of levels L_{\max} for which the correctness bound still holds. Note that in the left part, we take the minimal degree n_{\min} . This means that when choosing a power of 2 for the degree, the values for L_{\max} might change. In the right part, we take the largest possible value for q with the given maximal bit size.

It is important to ensure that the security bounds as well as the correctness bounds are both satisfied. Note that the authors of [9] failed to check their parameters presumably obtained from the correctness bound in the security bound, too, resulting in insecure parameters of $q = 2^{1358}$ and $n = 2^{10}$.

5.3 Implementation

Currently there are not many known implementation results for FHE schemes. Some of those which have been published demonstrate that the current state-of-the-art's performance is still rather unsatisfactory, see for example the implementations which are capable of computing AES homomorphically [12,8]. Other people have focused on implementing relatively simple schemes that require only a few levels of multiplications [14]. When using the ring $R = \mathbb{Z}[X]/(X^{4096} + 1)$, $t = 2^{10}$ and a 130-bit prime q , the authors of [14] present implementation results on an Intel Core 2 Duo running at 2.1 GHz. Encryption takes 756 ms, addition of ciphertexts 4 ms, multiplication of ciphertexts 1590 ms (this includes the degree reduction) and decryption 57 ms.

We have implemented the YASHE' variant proposed in Section 5 in a C-library. All the arithmetic has been built from scratch and we do not depend on any external number theory library. Using almost the same parameters (we use a 127-bit prime q) with $w = 2^{32}$ we obtained the following results on an Intel Core i7-3520M at 2893.484 MHz with hyperthreading turned off and over-clocking ("turbo boost") disabled. Encryption runs in 79.2 million cycles (27 ms), addition of ciphertexts in 70 thousand cycles (0.024 ms), multiplication of ciphertexts (including the key-switching) in 90.7 million cycles (31 ms) and decryption in 14.1 million cycles (5 ms).

This performance increase by at least one order of magnitude (for the decryption) to two orders of magnitude (for the addition of ciphertexts) can be partially explained by the fact that we are running on a more recent processor and that we implemented the scheme directly in C (avoiding the overhead incurred by using a computer algebra system as in [14]). The remainder of the speed-up is due to our newly proposed scheme, in particular due to a simpler multiplication operation on ciphertexts that uses a more compact evaluation key consisting of only 4 elements. These performance numbers highlight the fact that HE is much more practical for schemes which do not require very deep circuits (like AES) but instead only need a few (around 2^2 to 2^5) multiplications.

5.4 Truncating Ciphertext Words

Brakerski [3, Section 4.2] first suggested for his scale-invariant LWE scheme to discard some least significant bits of the ciphertext. Based on this idea, we

describe an optimization to our scheme which significantly reduces both the ciphertext length and the number of elements in the evaluation key. By aligning the number of bits we discard with a multiple of w used in `YASHE.KeySwitch`, the number of elements required to switch keys is reduced per multiplication.

Define `YASHE.Discardw(c, i)` as the function which takes as input a ciphertext and the number $0 \leq i < \ell_{w,q}$ of w -words to be truncated and outputs $c' = \text{YASHE.Discard}_w(c, i) = \lfloor w^{-i}c \rfloor$. Then, $w^i c'$ is equal to c with the i least significant w -words of c being set to 0. If $cf = \Delta m + v \pmod{q}$, then $w^i c' f = \Delta m + v' \pmod{q}$ with $\|v'\|_\infty \leq \|v\|_\infty + \frac{1}{2}\delta w^i \|f\|_\infty$. For a constant $B > 0$ such that $2B > \delta \|f\|_\infty / 2$, if we discard $\log_w(2B) - \log_w(\delta \|f\|_\infty)$ words, we incur an additional noise term of size B , but the ciphertext can now be represented by $\log_w(q/B) + \log_w(\delta \|f\|_\infty / 2)$ words. This means that, with discarding, the length of ciphertexts does not depend on the absolute value of q but only on the ratio of q to the noise in the ciphertext. Perhaps more importantly, this means that when we consider $D_{w,q}(c)$ for a ciphertext c with coefficients represented by roughly $\log_w(q/B)$ words, all the lowest $\log_w(B)$ words are now zero. If c is a ciphertext decryptable under f^2 , in the key switching step, we only need the top $\log_w(q/B)$ elements from the evaluation key to carry out the switch.

5.5 Encoding Input Data via the CRT

For our leveled homomorphic encryption scheme, we have given bounds on parameters and input data to ensure correctness and security. For applications such as outsourcing of storage and computation on private data to the cloud, it could be the case that the user requires a flexible system which allows for additional computation, more computation than was planned for when setting system parameters. We propose a way to extend the system to allow additional computation without resetting the parameters. For computations on integer values, the encoding of larger integers using the Chinese Remainder theorem allows for either greater precision of computation or larger integer inputs, using the same underlying field size and lattice dimension but at the cost of increasing the number of ciphertexts to be operated on.

Integer computations with results up to a bound B are done by encoding each input as a collection of integers modulo coprime t_i via the CRT. Computations are then carried out on the collection and correctly reflect the integer operations not involving any modular reductions, as long as the product of the t_i is greater than B . Each integer in the collection is encrypted as a separate ciphertext with respect to its corresponding plain text modulus t_i and those ciphertexts can be processed in parallel to return encrypted collections. After they are decrypted, the CRT is used to recover the output as an actual integer.

This approach is different than the ones introduced in [23] and [8], since in contrast to these schemes, we do not use the CRT to pack information into different plain text slots of a single ciphertext. Instead, we simply encrypt each part of the CRT encoding in a separate ciphertext with respect to its plain text modulus t_i . This introduces a different way of flexibility. Ciphertexts now consist of several ring elements, but can be processed in parallel. For example, this allows

to work on integers of double bit length by keeping the same parameters, only extending to two ciphertexts with different values for t_0 and t_1 .

6 Conclusions

We have proposed a new fully homomorphic encryption scheme based on the scheme by Stehlé and Steinfeld which removes the non-standard decisional small polynomial ratio assumption needed in the homomorphic encryption scheme by López-Alt, Tromer and Vaikuntanathan. Hence, the security is solely based on standard lattice assumptions and a circular security assumption. Our new scheme avoids modulus switching and keeps the size of ciphertexts to a single ring element. Furthermore, we have presented a more practical variant of our scheme which does not need the decisional small polynomial ratio assumption. For this latter scheme we presented parameters and implementation results.

Acknowledgments. The authors thank Adriana López-Alt for many useful suggestions and discussions, in particular for pointing out the possibility of a weaker assumption in Remark 1, Tancrede Lepoint for his comments and for noticing an error in an earlier version of Table 1, Nigel P. Smart for helpful advice and the anonymous reviewers for their constructive feedback.

References

1. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
2. Bos, J.W., Lauter, K., Loftus, J., Naehrig, M.: Improved security for a ring-based fully homomorphic encryption scheme (full version). Cryptology ePrint Archive, Report 2013/075 (2013), <http://eprint.iacr.org/>
3. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. In: ITCS, pp. 309–325 (2012)
5. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS, pp. 97–106 (2011)
6. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
7. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
8. Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013)
9. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), <http://eprint.iacr.org/>

10. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
11. Gentry, C., Halevi, S.: Implementing Gentry’s fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
12. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)
13. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
14. Lauter, K., Naehrig, M., Vaikuntanathan, V.: Can homomorphic encryption be practical?. In: Cachin, C., Ristenpart, T. (eds.) CCSW, pp. 113–124. ACM (2011)
15. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
16. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC, pp. 1219–1234 (2012)
17. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
18. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: SIAM J. on Comp., pp. 372–381. IEEE Computer Society (2004)
19. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009)
20. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011)
21. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, vol. 4, pp. 169–180. Academic Press, New-York (1978)
22. Schnorr, C.-P.: A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.* 53, 201–224 (1987)
23. Smart, N., Vercauteren, F.: Fully homomorphic SIMD operations. *Cryptology ePrint Archive, Report 2011/133* (2011), <http://eprint.iacr.org/>
24. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
25. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011)
26. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)

On the Relationship between Functional Encryption, Obfuscation, and Fully Homomorphic Encryption

Joël Alwen¹, Manuel Barbosa², Pooya Farshim³, Rosario Gennaro⁴, S. Dov Gordon⁵, Stefano Tessaro^{6,7}, and David A. Wilson⁷

¹ ETH Zurich

² HASLab – INESC TEC and Universidade do Minho

³ Fachbereich Informatik, Technische Universität Darmstadt

⁴ City University of New York

⁵ Applied Communication Sciences

⁶ University of California, Santa Barbara

⁷ MIT

Abstract. We investigate the relationship between Functional Encryption (FE) and Fully Homomorphic Encryption (FHE), demonstrating that, under certain assumptions, a Functional Encryption scheme supporting evaluation on two ciphertexts implies Fully Homomorphic Encryption. We first introduce the notion of Randomized Functional Encryption (RFE), a generalization of Functional Encryption dealing with randomized functionalities of interest in its own right, and show how to construct an RFE from a (standard) semantically secure FE. For this we define the notion of entropically secure FE and use it as an intermediary step in the construction. Finally we show that RFEs constructed in this way can be used to construct FHE schemes thereby establishing a relation between the FHE and FE primitives. We conclude the paper by recasting the construction of RFE schemes in the context of obfuscation.

Keywords: Functional encryption, Randomized functionality, Obfuscation, Homomorphic encryption.

1 Introduction

The goal of this work is to draw connections between FE and FHE. The motivation is twofold. On the one hand, from a purely theoretical point of view it is interesting to understand the relationship between these new concepts, both of which promise to guide much future research in the area. On the other hand, from a constructive point of view, we hope that by exploring this relationship we can also discover connections that will help us progress towards new constructions of FHE schemes, taking advantage of techniques used in constructing FE schemes. This motivation is even stronger in light of very recent developments in the area, which indicate that constructing efficient FE schemes supporting complex functionalities may be within our reach in the near future. For example, an FE for arbitrary functionalities has been recently proposed in [13], as well as functional encryption for regular languages in [19]. Goldwasser et al. recently demonstrated that FHE together with attribute-based encryption implies FE [12]; here

we study the opposite problem, asking what additional assumptions can be used to build FHE from FE.

To explain our high-level approach to accomplishing this goal, let us consider the main differences between FE and FHE. For any function $f(\cdot)$, an FHE scheme allows us to compute *an encryption* of $f(x)$ from an encryption of x ; whereas an FE scheme allows us to compute, *in the clear*, $f(x)$ from an encryption of x . Our intuition is that an FE scheme supporting functions of the type Enc_f , where $\text{Enc}_f(x) = \text{Enc}(f(x))$ is a re-encryption of $f(x)$, would be very close to constructing an FHE scheme. However, an immediate problem inherent to this approach is how to provide the tokens needed for the computation of Enc_f , when f can be arbitrary. We will demonstrate that this can be done if one considers FE schemes that support functions on two inputs $f(x_1, x_2)$. In this case, it is sufficient to publish the token for $\text{Enc}_{\text{NAND}}(x_1, x_2) = \text{Enc}(\neg(x_1 \wedge x_2))$, as any Boolean function can be computed using a circuit of NAND gates.

When we set out to formally prove this intuition we stumbled across two major definitional issues: First, the “re-encryption” functionality described above is a “randomized” functionality, a case which is not treated in the original definitions of FE [7,16]. Second, this approach to constructing FHEs from FEs relies on a functionality that re-encrypts under the original encryption key. However, this implies extending the syntax of functional encryption schemes to allow for functionalities that take the domain parameters as an additional input (i.e., where the functionality can depend on the master public key).

RANDOMIZED FUNCTIONAL ENCRYPTION. In Randomized Functional Encryption an encryptor is able to hide an input within a ciphertext in such a way that authorized decryptors can only recover the result of applying a randomized function to it. Intuitively, the encryption and/or decryption operations can take *additional randomness* that is fed into the randomized function upon decryption, ensuring that the recovered image $f(x)$ is distributed as if the randomized function was calculated on fresh randomness r as $f(x; r)$. This is a correctness criterion. As for functional encryption, the security goal in RFE is to ensure that the decryption operation leaks no more about x than that which could be inferred from the output of the randomized functionality with private randomness. We introduce a natural indistinguishability-based notion of security for randomized functional encryption to capture this intuition, and show that it suffices to establish a relation to fully homomorphic encryption. Indeed, we show that an IND-CPA-secure RFE supporting the *NAND re-encryption* functionality $\text{Enc}_{\text{NAND}}(x_1, x_2)$ described above can be used as a secure fully homomorphic encryption scheme.

Randomized Functional Encryption is an interesting cryptographic primitive in its own right. In the full version of this paper we show that various existing public-key encryption schemes can actually be seen as particular instances of this primitive, e.g., those supporting re-randomization of ciphertexts and re-encryption. That said, our primary motivation in introducing the notion of RFE is to enable the construction of FHE from FE, so we do not attempt to fully explore the subtleties of RFE. In particular, we provide and use a very strong indistinguishability notion for RFE which is

likely *too* strong to admit many interesting functionalities, but suffices for building FHE. We leave further exploration of RFE to future work.

FROM FUNCTIONAL ENCRYPTION TO RFE AND FHE. We show how FE schemes can be used to build RFE schemes, which, in turn, can be used to construct a fully homomorphic encryption scheme, as described above. To this end, we begin by extending the syntax and security definitions for functional encryption schemes as follows: 1) we modify the syntax to allow functionalities to take multiple inputs, as well as the domain parameters as an extra input; 2) we introduce a new indistinguishability notion of security for FE schemes that we call entropic security in the spirit of the eponymous definition of Russel and Wang [18]; and 3) we extend the semantic security definition of Gorbunov, Vaikuntanathan, and Wee [13] to accommodate our new syntactic extensions. We also discuss the relation between the two security notions we propose, and show that, under certain conditions, semantic security implies entropic security.

We then present a generic construction of an RFE scheme supporting a given randomized functionality RF_n of arity n , from an entropically secure FE supporting a specific functionality F_n of the same arity. Interestingly, for binary functionalities we require a PRF that resists a particular class of related-key attacks [5,4] in order to prove the security of our construction. When combined with the semantically secure FE construction of Gorbunov, Vaikuntanathan, and Wee [13], our results for the unary case yield positive feasibility results for RFE supporting complex functionalities.

Finally, we present two constructions of FHE from functional encryption. The direct construction from an RFE supporting NAND re-encryption mentioned above gives us that entropically secure functional encryption supporting a binary functionality (and taking also the domain parameters) implies FHE. However, we cannot show that the required RFE scheme can be constructed by assuming semantic security alone. Our second construction uses two RFEs and a PKE, and it enables leveled homomorphic computation using an adaptation of the bootstrapping technique of Gentry [11] to the functional setting. We can relate the security of this construction to semantically secure functional encryption supporting functionalities of arity 2 (there is no need for the functionality to take the domain parameters). As is typical of bootstrapping techniques, an extra assumption akin to key-dependent message (KDM) security [11] allows us to use this construction as an FHE scheme.

CONNECTION WITH OBFUSCATION. As our final contribution, we explore the problem of obfuscating specific re-encryption functionalities, introducing new notions extending those proposed in earlier works on re-encryption [14,9]. We then explain how to use such obfuscated circuits to obtain RFE schemes suitable for FHE constructions.

STRUCTURE OF THE PAPER. In Section 2 we recall the syntax and security of FE schemes, and introduce a set of extensions that we require for our results. In Section 3 we introduce randomized functional encryption. In Section 4 we introduce the notion of entropic security and, in Section 5, we present generic constructions of fully homomorphic encryption schemes from RFE schemes. Finally, the connection with obfuscated re-encryption is explored in Section 6.

2 Functional Encryption

NOTATION. We start by settling the notation. We write $x \leftarrow y$ for assigning value y to variable x . We write $x \leftarrow_{\S} X$ for sampling x from set X uniformly at random. If A is a probabilistic algorithm we write $y \leftarrow_{\S} A(x_1, \dots, x_n)$ for the action of running A on inputs x_1, \dots, x_n with random coin chosen uniformly at random, and assigning the result to y . We use “:” for appending to a list, and denote the empty string by ϵ . For random variable X we denote by $[X]$ the support of X , i.e., the set of all values that X takes with nonzero probability. All algorithms in this paper are probabilistic polynomial-time. We say $\eta(\lambda)$ is negligible if $|\eta(\lambda)| \in \lambda^{-\omega(1)}$. Finally in various security games we use the notation $\mathcal{A}^{\mathcal{O}}$ and adversary \mathcal{A} having oracle access to all oracles specified within the same figure as the game.

We now formalize the syntax and security of an FE scheme, extending it with respect to several features that were proposed in [7].

DETERMINISTIC FUNCTIONALITY. A deterministic functionality is an algorithm Fn implementing an n -ary function f over a parameter space Fn.Prms , a key space Fn.KeySp , and a plaintext space Fn.MsgSp^n :

$$f : \text{Fn.Prms} \times \text{Fn.KeySp} \times \text{Fn.MsgSp}^n \longrightarrow \text{Fn.Rng}.$$

We require that the key space contains a special key called the *empty key* denoted k_{ϵ} . The empty key abstracts the information that publicly leaks from a ciphertext. This function in most natural cases would only depend on a single argument, although one may envisage the more general case where it has arity n . Throughout the paper we assume the messages in the message space are of equal length, and whenever the empty key is not explicitly defined, it is assumed that it returns the length of its first input.

SYNTAX. A *functional encryption (FE) scheme* FE for the functionality Fn (of arity n) is specified by four algorithms as follows.

1. $\text{FE.Setup}(1^\lambda)$: This is the setup algorithm. On input the security parameter, this algorithm outputs a master secret key Msk and master public key Mpk , a key space, and a message space. We require that the set of all possible master public keys output by the setup algorithm, the key space, and the message space are identical to Fn.Prms , Fn.KeySp , and Fn.MsgSp respectively.
2. $\text{FE.TKGen}(k, \text{Msk})$: This is the token-generation algorithm. On input a key $k \in \text{Fn.KeySp}$, and a master secret key Msk , it outputs a token TK . We assume, without loss of generality, that the token for the empty functionality is the empty string.
3. $\text{FE.Enc}(m, \text{Mpk})$: This is the encryption algorithm. On input a message $m \in \text{Fn.MsgSp}$ and a master public key Mpk , it outputs a ciphertext c .
4. $\text{FE.Dec}(c_1, \dots, c_n, \text{TK}, \text{Mpk})$: This is the deterministic decryption (or evaluation) algorithm. On input n ciphertexts, a token TK , and a master public key Mpk , it outputs a message m or a special failure symbol \perp .

CORRECTNESS. We call scheme FE correct if, for $(\text{Msk}, \text{Mpk}) \leftarrow_{\S} \text{FE.Setup}(1^\lambda)$, any $m_1, \dots, m_n \in \text{Fn.MsgSp}$, randomly chosen $c_i \leftarrow_{\S} \text{FE.Enc}(m_i, \text{Mpk})$ for $1 \leq i \leq n$,

any $k \in \text{Fn.KeySp}$, and randomly chosen $\text{TK} \leftarrow_{\S} \text{FE.TKGen}(k, \text{Msk})$, $\Pr [\text{FE.Dec}(c_1, \dots, c_n, \text{TK}, \text{Mpk}) \neq \text{Fn}(\text{Mpk}, k, m_1, \dots, m_n)]$ is negligible as a function of λ .

SECURITY. We now discuss security notions for FE that match our syntactic extensions to the primitive. Given the limitations of indistinguishability-based notions already identified in [7,16] and further discussed in the full version of this paper, we adopt a semantic (or simulation-based) security notion akin to those in [7,16,13,3,6,2]. More precisely, we adopt of [13] extended to multiple encryption queries. Our choice is due to the conceptual simplicity of the model and to the general feasibility result we automatically obtain for unary functionalities. We restrict our attention to TNA queries to avoid impossibility results such as those presented in [7,6,2]. The definition formalizes the intuition that in a secure FE scheme, no information beyond that which is leaked through extracted tokens is available to an adversary. The fine details of the semantic security model are not important to our results, as long as the model is strong enough to imply an indistinguishability-based notion (Definition 8) that we introduce later in the paper.

Definition 1 (Semantic security). *Let games $\text{SS-R}_{\text{FE},\mathcal{A},\mathcal{D}}$ and $\text{SS-I}_{\text{FE},\mathcal{A},\mathcal{S},\mathcal{D}}$ be as in Figure 1. Let FE be an (n -ary) functional encryption scheme. We say FE is semantically secure if, for any adversary \mathcal{A} , there exists a simulator \mathcal{S} such that for all distinguishers \mathcal{D} the following definition of advantage is negligible.*

$$\text{Adv}_{\text{FE},\mathcal{A},\mathcal{S},\mathcal{D}}^{\text{ss-cpa}}(\lambda) := \Pr [\text{SS-R}_{\text{FE},\mathcal{A},\mathcal{D}}(1^\lambda) \Rightarrow \text{T}] - \Pr [\text{SS-I}_{\text{FE},\mathcal{A},\mathcal{S},\mathcal{D}}(1^\lambda) \Rightarrow \text{T}]$$

<p>Game $\text{SS-R}_{\text{FE},\mathcal{A},\mathcal{D}}(1^\lambda)$: $\text{KList} \leftarrow [k_\epsilon]; m \leftarrow \perp$ $(\text{Msk}, \text{Mpk}) \leftarrow_{\S} \text{FE.Setup}(1^\lambda)$ $\alpha \leftarrow_{\S} \mathcal{A}^{\text{Token}, \text{Enc}_0}(\text{Mpk})$ Return $\mathcal{D}(\text{MList}, \alpha)$</p>	<p>oracle $\text{Token}(k)$: $\text{TK} \leftarrow_{\S} \text{FE.TKGen}(k, \text{Msk})$ $\text{KList} \leftarrow k : \text{KList}$ Return TK</p>	<p>oracle $\text{Func}(\mathcal{I}, k, m_1, \dots, m_n)$: For $(i, j) \in \mathcal{I}$ do $m_i \leftarrow \text{MList}[j]$ Return $\text{Fn}(\text{Mpk}, k, m_1, \dots, m_n)$</p>
<p>Game $\text{SS-I}_{\text{FE},\mathcal{A},\mathcal{S},\mathcal{D}}(1^\lambda)$: $\text{KList} \leftarrow [k_\epsilon]; m \leftarrow \perp$ $(\text{Msk}, \text{Mpk}) \leftarrow_{\S} \text{FE.Setup}(1^\lambda)$ $\alpha \leftarrow_{\S} \mathcal{A}^{\text{Token}, \text{Enc}_1}(\text{Mpk})$ Return $\mathcal{D}(\text{MList}, \alpha)$</p>	<p>oracle $\text{Enc}_0(m)$: $c \leftarrow_{\S} \text{FE.Enc}(m, \text{Mpk})$ $\text{MList} \leftarrow m : \text{MList}$ Return c</p>	<p>oracle $\text{Enc}_1(m)$: $c \leftarrow_{\S} \mathcal{S}^{\text{Func}}(\text{Mpk}, \text{KList})$ $\text{MList} \leftarrow m : \text{MList}$ Return c</p>

Fig. 1. Games defining the semantic security of an FE scheme. \mathcal{A} is legitimate if it does not call **Token** after calling **Enc_b**. In the single-message model \mathcal{A} can call **Enc_b** exactly once. Simulator \mathcal{S} is legitimate if it queries **Func** with $k \in \text{KList}$ only.

THE Func ORACLE. Note that providing a ciphertext and a set of tokens to the adversary in the real world allows the adversary to compute many images of the functionality simply by encrypting fresh ciphertexts and decrypting (evaluating) them along with the

provided ciphertext. Our definition provides an oracle to the simulator in the ideal world that gives it essentially the same power. Here \mathcal{I} denotes a set of index pairs (i, j) which indicates hidden message j should be used in position i of the functionality. Note that in the single-message setting and for unary functionalities, this oracle is equivalent to directly providing a list of images to the simulator, as in the definition proposed in [13]

FEASIBILITY. The feasibility results presented in [13,16] directly carry over to our definition for unary functionalities in the single-message model, where the adversary may only request a single challenge ciphertext. This can be extended to the multi-message model (for the TNA case) through a composition theorem, but we do not discuss the details here due to space constraints. For the multi-ary case, however, the feasibility problems resurface even in this more restricted scenario. More precisely, a problem similar to that identified for noninteractive noncommitting encryption [15] may arise. However, we do not know of any impossibility result that excludes the construction of an FE scheme for the *specific* functionalities that we require for building FHE schemes and leave a detailed treatment to future research.

3 Randomized Functional Encryption

In this section we propose a new cryptographic primitive that generalizes FE to randomized functionalities. We start by formally introducing the syntax and security of randomized functional encryption schemes. In the full version we describe several examples of cryptographic primitives that can be seen as particular cases of RFE.

RANDOMIZED FUNCTIONALITY. A randomized functionality is a probabilistic algorithm RFn implementing an n -ary function f that also takes a set of random coins from a randomness space RFn.RndSp :

$$f : \text{RFn.Prms} \times \text{RFn.KeySp} \times \text{RFn.MsgSp}^n \times \text{RFn.RndSp} \longrightarrow \text{RFn.Rng} .$$

We assume the random coins are uniformly distributed over RFn.RndSp .

SYNTAX. A *randomized functional encryption (RFE) scheme* for the randomized functionality RFn (of arity n) is specified by four algorithms as follows.

1. $\text{RFE.Setup}(1^\lambda)$: This is the setup algorithm. On input the security parameter, it outputs a master secret key Msk and a master public key Mpk , a key space, and a message space. We require that the set of all possible master public keys output by the setup algorithm, the key space, and the message space are identical to RFn.Prms , RFn.KeySp , and RFn.MsgSp respectively.
2. $\text{RFE.TKGen}(k, \text{Msk})$: This is the token-generation algorithm. On input a key $k \in \text{RFn.KeySp}$ and a master secret key Msk , it outputs a token TK .
3. $\text{RFE.Enc}(m, \text{Mpk})$: This is the encryption algorithm. On input a message $m \in \text{RFn.MsgSp}$ and a master public key Mpk , it outputs a ciphertext c .
4. $\text{RFE.Dec}(c_1, \dots, c_n, \text{TK}, \text{Mpk})$: This is the (possibly probabilistic) decryption (or evaluation) algorithm. On input n ciphertexts, a token TK , and a master public key Mpk , it outputs a message m or a special symbol \perp .

CORRECTNESS. Intuitively, the correctness requirement imposes that the distribution of a decrypted value (over the random coins of the encryption and decryption algorithms) is computationally indistinguishable from that obtained by sampling the randomized functionality directly on the encrypted message. We formalize this property next.

Definition 2 (Correctness). Let game $\text{COR}_{\text{RFE}, \mathcal{A}}$ be as in Figure 2. Let RFE be a randomized functional encryption scheme. We say RFE is correct if, for any adversary \mathcal{A} , the following definition advantage is negligible.

$$\text{Adv}_{\text{RFE}, \mathcal{A}}^{\text{cor}}(\lambda) := 2 \cdot \Pr [\text{COR}_{\text{RFE}, \mathcal{A}}(1^\lambda) \Rightarrow \text{T}] - 1$$

<p>Game $\text{COR}_{\text{RFE}, \mathcal{A}}(1^\lambda)$:</p> <p>$b \leftarrow_{\S} \{0, 1\}$ $(\text{Mpk}, \text{Msk}) \leftarrow_{\S} \text{RFE.Setup}(1^\lambda)$ $b' \leftarrow_{\S} \mathcal{A}^{\text{O}}(\text{Mpk}, \text{Msk})$ Return $(b = b')$</p>	<p>oracle $\text{Func}(i_1, \dots, i_n, j)$:</p> <p>For $\ell = 1$ to n do $(m_\ell, c_\ell) \leftarrow \text{MList}[i_\ell]$ $(\text{TK}, k) \leftarrow \text{KList}[j]$ $y_0 \leftarrow_{\S} \text{RFE.Dec}(\text{TK}, c_1, \dots, c_n)$ $y_1 \leftarrow_{\S} \text{RFn}(\text{Mpk}, k, m_1, \dots, m_n)$ Return y_b</p>
<p>oracle $\text{Token}(k)$:</p> <p>$\text{TK} \leftarrow_{\S} \text{RFE.TKGen}(k, \text{Msk})$ $\text{KList} \leftarrow (\text{TK}, k) : \text{KList}$ Return TK</p>	<p>oracle $\text{Enc}(m)$:</p> <p>$c \leftarrow_{\S} \text{RFE.Enc}(m, \text{Mpk})$ $\text{MList} \leftarrow (m, c) : \text{MList}$</p>

Fig. 2. Game defining the correctness of an RFE scheme. An adversary is legitimate if all **Func** queries are distinct.

We emphasize that the adversary in the correctness game has access to Msk (this is needed to model correctness for Msk-dependent messages), and that it may force some of the ciphertexts to be repeatedly used in the input to the decryption algorithm.

SECURITY. Our proposed notions of security for RFE are indistinguishability based. We will formalize the definitions and then discuss why the models do not necessarily suffer from the same limitations as IND-CPA security for FE.

Definition 3 (Indistinguishability). Let game $\text{IND-CPA}_{\text{RFE}, \mathcal{A}}$ be as shown in Figure 3. We say RFE is IND-CPA secure if, for any adversary \mathcal{A} , the following definition of advantage is negligible.

$$\text{Adv}_{\text{RFE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda) := 2 \cdot \Pr [\text{IND-CPA}_{\text{RFE}, \mathcal{A}}(1^\lambda) \Rightarrow \text{T}] - 1$$

We observe that, unless the functionality has special characteristics, the images recovered by the adversary may allow it to trivially win the game; for instance, the above definition is unrealizable for nontrivial *deterministic* functionalities. To deal with this, we will consider restricted classes of *legitimate* adversaries that cannot “trivially” win the game. We start with a definition which characterizes when image values can be used to win the IND-CPA game.

Game $\text{IND-CPA}_{\text{RFE}, \mathcal{A}}(1^\lambda)$: $b \leftarrow_{\S} \{0, 1\}$ $(\text{Msk}, \text{Mpk}) \leftarrow_{\S} \text{RFE.Setup}(1^\lambda)$ $b' \leftarrow_{\S} \mathcal{A}^{\text{O}}(\text{Mpk})$ Return $(b = b')$	oracle $\text{LR}(m_0, m_1)$: $c \leftarrow_{\S} \text{RFE.Enc}(m_b, \text{Msk})$ Return c	oracle $\text{Token}(k)$: $\text{TK} \leftarrow_{\S} \text{RFE.TKGen}(k, \text{Msk})$ Return TK
---	--	--

Fig. 3. Game defining the IND-CPA security of an RFE. In the TNA model \mathcal{A} may not query **Token** after **LR**.

Definition 4 (Message-hiding RFE). Let game $\text{MH}_{\text{RFE}, \mathcal{A}}$ be as in Figure 4. We say RFE is message hiding if, for any adversary \mathcal{A} , the following definition of advantage is negligible.

$$\text{Adv}_{\text{RFE}, \mathcal{A}}^{\text{mh}}(\lambda) := 2 \cdot \Pr [\text{MH}_{\text{RFE}, \mathcal{A}}(1^\lambda) \Rightarrow \text{T}] - 1$$

Game $\text{MH}_{\text{RFE}, \mathcal{A}}(1^\lambda)$: $b \leftarrow_{\S} \{0, 1\}$ $(\text{Msk}, \text{Mpk}) \leftarrow_{\S} \text{RFE.Setup}(1^\lambda)$ $b' \leftarrow_{\S} \mathcal{A}^{\text{O}}(\text{Mpk})$ Return $(b = b')$	oracle $\text{Func}(\mathcal{I}, m_1, \dots, m_n, k)$: For $(i, j) \in \mathcal{I}$ do $(m^0, m^1) \leftarrow \text{MList}[j]$ $m_i \leftarrow m^b$ $y \leftarrow_{\S} \text{RFn}(\text{Mpk}, k, m_1, \dots, m_n)$ Return y
oracle $\text{LR}(m_0, m_1)$: $\text{MList} \leftarrow (m_0, m_1) : \text{MList}$	oracle $\text{Token}(k)$: $\text{TK} \leftarrow_{\S} \text{RFE.TKGen}(k, \text{Msk})$ $\text{KList} \leftarrow k : \text{KList}$ Return TK

Fig. 4. Game defining the message-hiding property. An adversary \mathcal{A} is legitimate if it queries **Func** with $k \in \text{KList}$ only. In the TNA model \mathcal{A} may not query **Token** after querying **LR**.

The message-hiding property detects whether an adversary can distinguish messages queried to the **LR** oracle by looking at images of the functionality. Note that this may depend intrinsically on the way in which the domain parameters are sampled by the setup algorithm of the RFE scheme, and hence we have honest parameter generation. We can now introduce variants of the IND-CPA definition where we exclude trivial attacks by adversaries that win the game by exploiting information leaked via the images. To this end, we formalize the notion of an *associated adversary* \mathcal{B} that mimics \mathcal{A} 's capabilities in the MH game.

Definition 5 (Associated adversary). Let RFE be an RFE scheme, and let \mathcal{A} be an adversary in the $\text{IND-CPA}_{\text{RFE}, \mathcal{A}}$ game. Let also \mathcal{B} be an adversary in the $\text{MH}_{\text{RFE}, \mathcal{B}}$ game. Define the traces of \mathcal{A} and \mathcal{B} to be $(\text{Mpk}, \text{MList}, \text{KList}, \text{TKList})$ in their respective games, where TKList is the list of tokens returned by the **Token** oracle. We say \mathcal{B}

is an adversary associated to \mathcal{A} if the traces of \mathcal{A} and \mathcal{B} are computationally indistinguishable. Weak associated adversary is defined analogously, where TKList is omitted from the traces.

We now set legitimacy criteria for IND-CPA adversaries, excluding MH attacks.

Definition 6 (Legitimacy). Let RFE be a randomized functional encryption scheme, and let \mathcal{A} be an $\text{IND-CPA}_{\text{RFE}, \mathcal{A}}$ adversary. We say \mathcal{A} is legitimate if the advantage of any adversary \mathcal{B} associated to \mathcal{A} in the $\text{MH}_{\text{RFE}, \mathcal{B}}$ game is negligible.

Let us now introduce our weaker indistinguishability notions before discussing the legitimacy conditions.

Definition 7 (Restricted indistinguishability). Let RFE be a randomized functional encryption scheme. The R-IND-CPA security of RFE requires the advantage of any legitimate adversary \mathcal{A} in the IND-CPA game to be negligible.

The following implications are easy to verify: $\text{IND-CPA} \implies \text{R-IND-CPA}$; $\text{R-IND-CPA} + \text{MH} \implies \text{IND-CPA}$. Note also that, if scheme RFE is IND-CPA secure, then the supported functionality must be MH secure. (If the functionality is not message hiding, then the adversary could trivially break the security of the RFE scheme by simply distinguishing the images it recovers using legitimately retrieved tokens). As we shall see later in the paper, our constructions of a homomorphic scheme from an RFE impose that the adversary is *unrestricted* in its challenge query. One way to achieve this is to concentrate on RFEs that are *provably* message hiding. The potential limitations of this indistinguishability-based definition are discussed in the full version of this paper.

4 Entropic Security

We now turn our attention to a particular class of (standard) functional encryption schemes that we will use as a stepping-stone between the standard notions of functional encryption and randomized functional encryption. We call such functional encryption schemes *entropically secure*.

Intuitively, entropic security imposes that an adversary has a negligible advantage in distinguishing encryptions *provided that a part of the encrypted message is sampled uniformly at random*. To this end, we must restrict our attention to functionalities Fn for which the plaintext space Fn.MsgSp is partitioned as $\text{Fn.MsgSp}_m \times \text{Fn.MsgSp}_r$. For FE schemes supporting such functionalities, we can define a natural adaptation of the IND-CPA model as follows.

Definition 8 (Entropic indistinguishability). Let game $\text{IND-ECPA}_{\text{FE}, \mathcal{A}}$ be as in Figure 5. Let FE be a functional encryption scheme with a portioned message space. We say FE is IND-ECPA secure if for any adversary \mathcal{A} , the following definition of advantage is negligible.

$$\text{Adv}_{\text{FE}, \mathcal{A}}^{\text{ind-ecpa}}(\lambda) := 2 \cdot \Pr [\text{IND-ECPA}_{\text{FE}, \mathcal{A}}(1^\lambda) \Rightarrow \text{T}] - 1$$

Game $\text{IND-CPA}_{\text{FE}, \mathcal{A}}(1^\lambda)$: $b \leftarrow_{\S} \{0, 1\}$ $(\text{Msk}, \text{Mpk}) \leftarrow_{\S} \text{FE.Setup}(1^\lambda)$ $b' \leftarrow_{\S} \mathcal{A}^{\text{O}}(\text{Mpk})$ Return $(b = b')$	oracle $\text{LR}(m_0, m_1)$: $r \leftarrow_{\S} \text{FE.MsgSp}_r$ Return c	oracle $\text{Token}(k)$: $\text{TK} \leftarrow_{\S} \text{FE.TKGen}(k, \text{Msk})$ Return TK
---	--	---

Fig. 5. Game defining the IND-ECPA security of an FE scheme. In the TNA model \mathcal{A} may not query **Token** after querying **LR**.

Unlike the standard definition of IND-CPA, the above definition is meaningful for functionalities of arbitrary arity. Note also that unless the functionality has special characteristics the images recovered by the adversary may allow it to trivially win the game. We therefore adopt a similar strategy to that presented for RFE security, define an *entropic message-hiding* property, and impose a legitimacy condition on the adversary to obtain a weaker flavor of the definition. We present these definitions in the full version of the paper. In entropic message hiding, the two differences to the model presented for RFEs are the following. Firstly, the **LR** oracle now samples part of the hidden message, as in the IND-ECPA definition. Secondly, the **Func** oracle allows the adversary to evaluate the functionality choosing all the r inputs except those that correspond to **LR** queries. This means that the EMH property (contrarily to message hiding for RFEs, where fresh images are sampled on each query to **Func**) entails a form of security under *randomness reuse* across the different functions k that are queried by the adversary. Note also that for multi-ary functionalities the adversary can *maliciously* choose part of the randomness components in the input to Fn . Later in the paper we will see how to build entropically message-hiding FEs. The definition of legitimacy is analogous to that presented for RFEs. We call the resulting definition *restricted* entropic indistinguishability. We also note that the legitimacy of an IND-ECPA adversary is a natural generalization of the restriction imposed in the IND-CPA model for FEs that the functions queried to the **Token** oracle must all collide on the challenge messages.

Constructing RFEs from FEs We now show how to construct RFEs for arbitrary functionalities starting from FE schemes. Technically, we will show that specific entropically secure FE schemes suffice, and then establish a connection to semantically secure FE.

The intuition behind our construction is the following. Suppose our goal is to construct an RFE scheme for a functionality RF_n of arity n . We begin by defining a derandomized version of RF_n , denoted Fn , of arity n , which we call the deterministic functionality *associated* to RF_n . We then show that the simple generic construction in Figure 6 converts a functional encryption scheme FE into a correct and secure RFE scheme for RF_n , provided that the underlying scheme FE is correct and entropically secure for Fn . Observe that the only modification that is made to the original functional encryption scheme is that the encryption algorithm samples extra randomness that is encrypted along with the message. This naturally relates to entropic security, where we considered functionalities where the message space was partitioned as $\text{MsgSp} = \text{MsgSp}_m \times \text{MsgSp}_r$. We now formalize this intuition.

algo. RFE.Setup(1^λ): Return FE.Setup(1^λ)	algo. RFE.TKGen(k, Msk): Return FE.TKGen(k, Msk)
algo. RFE.Enc(m, Mpk): $r \leftarrow_{\S} \text{Fn.MsgSp}_r$ $c \leftarrow_{\S} \text{FE.Enc}((m, r), \text{Mpk})$ Return c	algo. RFE.Dec($c_1, \dots, c_n, \text{TK}, \text{Mpk}$): Return FE.Dec($c_1, \dots, c_n, \text{TK}, \text{Mpk}$)

Fig. 6. Generic construction of an RFE scheme from an FE scheme

CORRECTNESS. We begin by defining what it means to *correctly derandomize* a randomized functionality. We require that Fn and RFn are computationally indistinguishable to an adversary with oracle access that can choose all message inputs to Fn/RFn (except of course the components coming from Fn.MsgSp_r).

Definition 9 (Derandomized functionality). Let game $\text{DRND}_{\text{RFn}, \text{Fn}, \mathcal{A}}$ be as shown in Figure 7. Let RFn be an n -ary randomized functionality. Let Fn be a deterministic functionality with the same arity, parameter space, and key space as those of RFn . Suppose further that the message space of Fn is partitioned as $\text{Fn.MsgSp} = \text{RFn.MsgSp} \times \text{Fn.MsgSp}_r$. We say Fn correctly derandomizes RFn if, for any adversary \mathcal{A} , the following definition of advantage is negligible.

$$\text{Adv}_{\text{RFn}, \text{Fn}, \mathcal{A}}^{\text{drnd}}(\lambda) := 2 \cdot \Pr [\text{DRND}_{\text{RFn}, \text{Fn}, \mathcal{A}}(1^\lambda) \Rightarrow \text{T}] - 1$$

Game $\text{DRND}_{\text{RFn}, \text{Fn}, \mathcal{A}}(1^\lambda)$: $b \leftarrow_{\S} \{0, 1\}$ $b' \leftarrow_{\S} \mathcal{A}^{\text{O}}(1^\lambda)$ Return $(b = b')$	oracle $\text{Rand}(m)$: $r \leftarrow_{\S} \text{Fn.MsgSp}_r$ List $\leftarrow (m, r) : \text{List}$	oracle $\text{Func}(\text{Mpk}, i_1, \dots, i_n, k)$: For $\ell = 1$ to n do $(m_\ell, r_\ell) \leftarrow \text{List}[i_\ell]$ $y_0 \leftarrow \text{Fn}(\text{Mpk}, k, (m_1, r_1), \dots, (m_n, r_n))$ $y_1 \leftarrow_{\S} \text{RFn}(\text{Mpk}, k, m_1, \dots, m_n)$ Return y_b
--	---	---

Fig. 7. Game defining correct derandomization of RFn by Fn . An adversary is legitimate if all Func queries are distinct.

Observe that the derandomized functionality must simulate independent samplings of RFn , whilst reusing the same input randomness. We now state the correctness result. The proof is a direct reduction and is given in the full version.

Proposition 1. Let RFn and Fn be a randomized and a deterministic functionality, respectively. Suppose that Fn correctly derandomizes RFn . Then, any correct FE supporting Fn yields, by the construction in Figure 6, a correct RFE scheme supporting RFn .

BUILDING DERANDOMIZED FUNCTIONALITIES. We now consider how unary and binary functionalities can be derandomized. (The techniques used to deal with the binary

case naturally extend to arity greater than 2.) For each randomized functionality that we want to support in the generic RFE construction of Figure 6, we define an associated deterministic functionality as follows.

Definition 10 (Associated deterministic functionality). *Let RFn be a randomized functionality.*

Unary RFn : *Let $\text{PRF} : \text{PRF.KeySp} \times \text{RFn.KeySp} \rightarrow \text{RFn.RndSp}$ be a pseudorandom function. The associated unary deterministic functionality Fn is*

$$\text{Fn}(\text{Mpk}, k, (m, r)) := \text{RFn}(\text{Mpk}, k, m; \text{PRF}_r(k)) .$$

The key space and the parameter space of Fn match those of RFn and the message space of Fn is $\text{RFn.MsgSp} \times \text{Fn.MsgSp}_r$, where $\text{Fn.MsgSp}_r := \text{PRF.KeySp}$.

Binary RFn : *Let $\text{PRF} : \text{PRF.KeySp} \times (\{0, 1\}^{2\lambda} \times \text{RFn.KeySp}) \rightarrow \text{RFn.RndSp}$ be a pseudorandom function with PRF.KeySp forming an abelian group with operation \circ . The associated binary deterministic functionality Fn is defined as*

$$\text{Fn}(\text{Mpk}, k, (m_1, r_1, s_1), (m_2, r_2, s_2)) := \text{RFn}(\text{Mpk}, k, m_1, m_2; \text{PRF}_{r_1 \text{ or } r_2}(s_1 || s_2, k)) .$$

The key space and the parameter space of Fn match those of RFn and the message space of Fn is $\text{RFn.MsgSp}_m \times \text{Fn.MsgSp}_r$, where $\text{Fn.MsgSp}_r := \text{PRF.KeySp} \times \{0, 1\}^\lambda$.

We first show that an associated deterministic functionality, as defined as above, satisfies the correctness derandomization criterion. The proof of the following theorem is a direct reduction and can be found in the full version.

Theorem 1 (Derandomization via PRFs). *Let RFn be a unary or a binary randomized functionality. Let Fn be the associated deterministic functionality to RFn that uses the pseudorandom function PRF . Suppose that PRF satisfies the standard notion of PRF security. Then Fn correctly derandomizes RFn .*

SECURITY. We can now discuss the security of the construction. The next theorem shows that an FE scheme supporting the associated deterministic functionality to RFn suffices to obtain a secure RFE with an analogous security level.

Theorem 2 (Security of the RFE construction). *Let RFn be a randomized functionality and let Fn be its associated deterministic functionality. Let FE be a functional encryption scheme supporting Fn . Then if FE is IND-ECPA secure, the RFE scheme resulting from the generic construction in Figure 6 is IND-CPA secure. A similar result holds for the restricted indistinguishability game if the PRF is Φ -RKA secure, where*

$$\Phi := \left\{ \begin{array}{l} \phi_i^{\downarrow} : (K_1, \dots, K_n) \mapsto K_i \circ L \mid L \in \text{PRF.KeySp} \\ \phi_{i,j}^* : (K_1, \dots, K_n) \mapsto K_i \circ K_j \mid i, j \in \mathbb{N} \wedge i \leq j \end{array} \right\} .$$

The proof is a direct reduction. However, for the restricted case, one needs to prove that the resulting IND-ECPA adversary is legitimate. The most challenging part of this argument consists of establishing that any successful EMH adversary against FE gives rise to a successful MH adversary against RFE. The intuition is as follows. The RKA

security of the PRF ensures that its outputs look random even if one of the inputs to the functionality is maliciously chosen. This allows us to make a transition from the entropically message-hiding game to another game where the random coins of Fn are generated via a truly random function. The reduction to the legitimacy then consists of proving that any adversary succeeding in this game would either be breaking the MH property, or triggering the unlikely event of guessing input s , when this is sampled uniformly at random. Details of the proof may be found in the full version.

Constructing Entropically Secure FE In this section we discuss how to construct entropically secure FEs from semantically secure functional encryption schemes. In our constructions we will be relying on FE schemes supporting functionalities that do *not* depend on the domain parameters, as this is the standard FE notion from [7,13] (modulo the arity extension). Therefore we first discuss how to construct a functionality that may depend on the parameters from one with a larger key space that no longer does. This then leads us to a natural transformation of FE schemes converting an FE scheme supporting the constructed parameter-independent functionality to one which supports the original parameter-dependent functionality. We show that this transformation preserves semantic security, and conclude the section by showing that any semantically secure FE scheme is also entropically secure.

REMOVING PARAMETER DEPENDENCY. Let Fn be a parameter-dependent functionality. We define a parameter-independent functionality as

$$\overline{\text{Fn}}(\epsilon, (\text{Mpk}, \text{key}), m_1, \dots, m_n) := \text{Fn}(\text{Mpk}, \text{key}, m_1, \dots, m_n).$$

Therefore, the key space of $\overline{\text{Fn}}$ is $\text{Fn.Prms} \times \text{Fn.KeySp}$. Let us look at a concrete example useful for our purposes. Consider an FE scheme in which we would like to support the following binary deterministic parameter-dependent functionality:

$$\text{Fn}_{\text{NAND}}(\text{pk}, \text{k}_{\text{NAND}}, (b_1, (r_1, s_1)), (b_2, (r_2, s_2))) := \text{PKE.Enc}(\neg(b_1 \wedge b_2), \text{pk}; \text{PRF}_{r_1 \circ r_2}(s_1 || s_2, \text{k}_{\text{NAND}})),$$

where PKE is an encryption scheme and k_{NAND} is the only key supported by the functionality. The converted functionality has key space identical to the public key space of the PKE, and is given by

$$\overline{\text{Fn}}_{\text{NAND}}(\epsilon, (\text{pk}, \text{k}_{\text{NAND}}), (b_1, (r_1, s_1)), (b_2, (r_2, s_2))) := \text{PKE.Enc}(\neg(b_1 \wedge b_2), \text{pk}; \text{PRF}_{r_1 \circ r_2}(s_1 || s_2, \text{k}_{\text{NAND}})).$$

We present our transformation in Figure 8, where we build scheme FE for Fn_{NAND} from scheme $\overline{\text{FE}}$ for $\overline{\text{Fn}}_{\text{NAND}}$. Note that although $\overline{\text{FE}}_{\text{NAND}}$ permits extracting tokens for encryption under *all* public keys in the underlying PKE scheme, FE_{NAND} samples a *single* public key at set-up, which it publishes along with the master public key. The fact that the public key is sampled honestly means that not only we can rely on the security properties of the PKE, but also that we can include (pk, sk) in the master key for FE_{NAND} . This means that the holder of the master public key is capable of recovering encrypted messages from Fn_{NAND} images, a feature that we will use later on.

<p>algo. $\text{FE}_{\text{NAND}}.\text{Gen}(1^\lambda)$: $(pk, sk) \leftarrow_{\S} \text{PKE}.\text{Gen}(1^\lambda)$ $(Msk', Mpk') \leftarrow_{\S} \overline{\text{FE}}_{\text{NAND}}.\text{Setup}(1^\lambda)$ $Mpk \leftarrow (Mpk', pk)$ $Msk \leftarrow (Msk', sk, pk)$ Return (Msk, Mpk)</p> <p>algo. $\text{FE}_{\text{NAND}}.\text{TKGen}(Msk, k_{\text{NAND}})$: $(Msk', sk, pk) \leftarrow Msk$ $TK \leftarrow_{\S} \overline{\text{FE}}_{\text{NAND}}.\text{TKGen}(k_{pk}, Msk')$ Return TK</p>	<p>algo. $\text{FE}_{\text{NAND}}.\text{Enc}(m_1, m_2, Mpk)$: $(Mpk', pk) \leftarrow Mpk$ $c \leftarrow_{\S} \overline{\text{FE}}_{\text{NAND}}.\text{Enc}(m_1, m_2, Mpk')$ Return c</p> <p>algo. $\text{FE}_{\text{NAND}}.\text{Dec}(c, TK, Mpk)$: $(Mpk', pk) \leftarrow Mpk$ Return $\overline{\text{FE}}_{\text{NAND}}.\text{Dec}(c, TK, Mpk')$</p>
---	---

Fig. 8. Scheme FE_{NAND} for Fn_{NAND} based on scheme $\overline{\text{FE}}_{\text{NAND}}$ for $\overline{\text{Fn}}_{\text{NAND}}$.

The following result establishes that the above transformation yields a correct and secure FE scheme. The proof can be found in the full version. The intuition is that FE_{NAND} uses only a subset of the functionality of $\overline{\text{FE}}_{\text{NAND}}$, and so the simulator implied by the hypothesis can be used to establish the semantic security of the construction.

Proposition 2. *If scheme $\overline{\text{FE}}_{\text{NAND}}$ is correct (for $\overline{\text{Fn}}_{\text{NAND}}$) and semantically secure, then scheme FE_{NAND} is correct for Fn_{NAND} and semantically secure.*

An important aspect of this result is that it goes through for the case in which the semantic security adversary places a single extraction query. This is important for feasibility, as the construction in [13] of FE schemes for general functionalities imposes the restriction that the semantic security adversary places a bounded number of such queries. Looking ahead, the FHE constructions we will present rely on FEs supporting functionalities with a single key.

FROM SEMANTIC TO ENTROPIC SECURITY. We now show that semantic security is strong enough to imply entropic security. Given that our definition of semantic security is restricted to the TNA scenario, we obtain entropic security in a similar setting. Luckily, entropic security under TNA attacks suffices for the results in the next section.

Theorem 3. *Let FE be a (deterministic) functional encryption scheme. Then if FE is semantically secure, it is also entropically secure in the TNA model.*

The intuition of the proof (given in the full version) is as follows. Any IND-ECPA attacker can be recast as a semantic security attacker that wins the real-world game with the same probability. The key observation is that the ideal world environment matches to entropic message hiding game, when this is played by an associated adversary to the original IND-ECPA attacker. However, the legitimacy condition on the attacker implies that this associated adversary cannot be successful, and that the real-world advantage (and hence the IND-ECPA advantage) must also be negligible.

Putting the above results together we obtain a path to constructing an RFE scheme for a randomized functionality RFn as follows. 1) Construct Fn , the associated deterministic functionality to RFn ; 2) Construct scheme FE that is correct for Fn ; 3) Prove that

FE is semantically secure, and hence, by the above theorem, it is R-IND-ECPA secure; 4) Let RFE be the randomized scheme associated to FE. By Theorem 2 it is R-IND-CPA secure; 5) Finally, to achieve security against unrestricted adversaries (IND-CPA security), establish that RFE is message hiding. We will be using this strategy in the following section.

5 Relating Homomorphic and Functional Encryption Schemes

We saw earlier in the paper that homomorphic public-key encryption can be seen as a particular instance of a randomized functional encryption scheme. We now combine this observation with our results from the previous section to formalize a relation between FHE and FE.

FHE FROM ENTROPICALLY SECURE FE. We restrict our attention to homomorphic public-key encryption schemes that support encrypting bits (rather than strings) and which allow the homomorphic computation of an arbitrary number of NAND gates. We note that since NAND is complete, we can support the evaluation of arbitrary functions by first representing the function as a circuit of NAND gates and using bit-wise encryption on the inputs.

We start from an RFE scheme supporting the following binary randomized functionality RFn , which we call *NAND re-encryption*, and is given by

$$\text{RFn}(\text{Mpk}, k_{\text{NAND}}, b_1, b_2) := \text{RFE.Enc}(\neg(b_1 \wedge b_2), \text{Mpk}).$$

This functionality has message space $\{0, 1\}$ and it supports a single key k_{NAND} (in addition to the empty key). We also assume, for the sake of correctness, that RFE supports a special decryption operation $\text{Dec}(\text{Msk}, c)$ akin to that of PKEs.

Our first construction of a fully homomorphic encryption scheme is as follows. The key generation algorithm generates RFE domain parameters and further extracts the token for k_{NAND} , which is added to the public key. Encryption is simply RFE encryption, and decryption is carried out using the special algorithm $\text{Dec}(\text{Msk}, c)$. Evaluation of a single NAND gate on two ciphertexts is carried out by running $\text{RFE.Dec}(c_1, c_2, \text{TK}_{k_{\text{NAND}}})$. Furthermore, the correctness of the underlying RFE scheme ensures that one can keep computing over the encrypted results, as the evaluated ciphertext will be with overwhelming probability in the co-domain of the RFE encryption circuit. In this way one can evaluate circuits of arbitrary size. Finally, it is easy to see that the IND-CPA security of the resulting FHE directly reduces to the IND-CPA security of the RFE scheme for single-message, TNA attacks, where by definition the adversary is *unrestricted*. We also observe that, by construction, this FHE is compact and function hiding. This is because the result of any computation is indistinguishable from a fresh encryption of the result of the computation, even to the holder of the decryption key. This construction and Theorems 1 and 2 immediately yield the following result.

Theorem 4. *Entropically secure FE with respect to unrestricted adversaries in the single message, TNA model and supporting the deterministic functionality associated with NAND re-encryption implies fully homomorphic encryption.*

We note that the underlying FE must be entropically secure against *unrestricted* adversaries. However, as we are dealing with bit-wise encryption, this is really the minimal assumption that one could have: the scheme should be secure when the adversary is allowed to choose challenge messages $m_0 \neq m_1$.

FHE FROM SEMANTICALLY SECURE FE. The previous construction reveals an interesting relation between entropically secure FE, RFE, and FHE. However, it does not give a relation between semantically secure FE and FHE. It would be tempting to try to build an RFE scheme such as the one described above from semantically secure FE, using Theorem 3 to obtain security against restricted adversaries, and then proving that the resulting RFE is message hiding to obtain unrestricted security. However, this approach fails: the fact that the randomized functionality is defined using the same RFE scheme that supports it introduces a circular dependency that we cannot overcome. Intuitively, assuming that self-re-encryption is message hiding amounts to assuming that RFE construction is secure to begin with. (Note that in Theorem 4 this circular argument is broken by explicitly assuming security against unrestricted adversaries.)

We present an alternative, slightly more involved construction to overcome the above difficulty. We require two RFE schemes supporting the following binary functionalities, each having a single key (in addition to the empty key).

RFE_{NAND} supports NAND re-encryption, with the caveat that re-encryption targets a standard public-key encryption scheme. More precisely,

$$\text{RFn}_{\text{NAND}}((\text{Mpk}, \text{pk}), k_{\text{NAND}}, b_1, b_2) := \text{PKE}.\text{Enc}(\neg(b_1 \wedge b_2), \text{pk}).$$

We also impose that the master secret key for RFE_{NAND} includes the secret key sk corresponding to pk .

RFE_{boot} enables a functional analogue of the bootstrapping technique of Gentry [11]. It permits functionally decrypting a ciphertext under PKE and re-encrypting it under RFE_{NAND} :

$$\text{RFn}_{\text{boot}}((\text{Mpk}, \text{Mpk}_{\text{NAND}}), k_{\text{boot}}, c, \text{sk}) := \text{RFE}_{\text{NAND}}.\text{Enc}(\text{PKE}.\text{Dec}(c, \text{sk}), \text{Mpk}_{\text{NAND}}).$$

We also impose that the master secret key for RFE_{boot} includes the master secret key Msk_{NAND} corresponding to Mpk_{NAND} .

We observe that these RFE schemes can be constructed from parameter-independent FE schemes as discussed in Section 4. Indeed, RFE_{NAND} can be constructed directly from the FE_{NAND} construction in Figure 6. Also, from Theorems 2 and 3, these RFEs will be R-IND-CPA secure in the TNA model if the underlying FE schemes are semantically secure. To establish IND-CPA security, it suffices to prove the message-hiding property. We address this for RFE_{NAND} , as a similar argument follows for RFE_{boot} . The following result follows from the observation that the message-hiding game reduces to the IND-CPA security of the underlying PKE scheme.

Theorem 5. *If the underlying PKE is IND-CPA secure, then the RFE scheme supporting RFn_{NAND} that results from plugging the FE scheme in Figure 8 in the generic construction in Figure 6 is message hiding.*

Using these RFE schemes, we construct an FHE scheme as follows. To encrypt or decrypt, one simply perform the equivalent operation using a pk for the underlying PKE. Evaluation of a single NAND gate proceeds as follows. The two ciphertexts are independently re-encrypted under RFE_{boot} , and then they are independently functionally decrypted and re-encrypted under RFE_{NAND} using TK_{boot} . To enable this operation, sk is encrypted under Mpk_{boot} and published in the public key, as is customary in bootstrapped constructions. Given any two PKE ciphertexts, one can therefore convert them into encryptions under RFE_{NAND} from which one can evaluate a NAND gate and re-encrypt again under the PKE. We obtain a ciphertext that is indistinguishable from a fresh encryption, which means that the construction is compact and function hiding.

The construction is correct if the underlying RFE schemes are correct. For security, note that one can easily reduce the IND-CPA security of the resulting FHE scheme to that of RFE_{boot} if the secret key encrypted under c_{boot} does not correspond to the public key pk used inside RFE_{NAND} . This is typical in bootstrapping techniques [11], and the additional assumption that the construction securely encrypts key-dependent messages (i.e., that it is KDM secure) is necessary to use the scheme with the same pk . On the other hand, without relying on this assumption, this result implies a leveled homomorphic encryption scheme [11], allowing a bounded number of cascaded NAND computations through multiple independent instances of RFE_{boot} . The last theorem in the paper follows from this discussion.

Theorem 6. *Semantically secure FE schemes supporting the deterministic functionalities associated to RF_{NAND} and RF_{boot} imply fully homomorphic encryption, under a key-dependent message security assumption.*

6 Relation to Obfuscated Re-encryption

FUNCTIONAL RE-ENCRYPTION IN THE CIRCUIT MODEL. For concreteness, we consider the functional re-encryption notions of Section 5 in the randomized circuit model. Specifically, we consider a family of re-encryption circuits $\mathcal{R}^f = \{R_{sk, pk'}^f\}$, called the *f-re-encryption functionality* from PKE to PKE', which decrypts each input (which is a ciphertext encrypted with PKE) using key sk , applies the function f to the resulting values, and outputs the encryption of the result under key pk' using PKE'. (In the full version, we consider the general discussion of functionalities that do not necessarily re-encrypt with fresh randomness, but here we restrict ourselves to such “canonical” functionalities.) Intuitively, a “secure” obfuscation of this circuit should serve as an evaluation key for a functional re-encryption scheme for function f .

OBFUSCATION FOR *f*-RE-ENCRYPTION. We define new notions of secure obfuscation as specifically applied to a *f*-re-encryption functionality \mathcal{R}^f . Following earlier work on obfuscation [20,10,1,14,8], we want the obfuscated circuit to perform the same computation as the original circuit without revealing any information beyond its input-output behavior.

Our notion differs from the one proposed by Chandran et al [9], while still following the same average-case viewpoint. We attempt to capture at the same time the fact

that the obfuscated re-encryption functionality does not reveal *any* information beyond black-box access to the functionality *and* the fact that black-box access to the functionality does not reveal any information about the messages being encrypted. Still, our notion is connected to (and in many cases implied by) the notion defined in these earlier work, as we explain below.

Definition 11 (Re-encryption Obfuscation). *Let Obf be a PPT algorithm whose input and output are both circuits. We say Obf securely obfuscates the f -re-encryption functionality \mathcal{R}^f from PKE to PKE' if the following properties hold:*

Correctness: *For any circuit $C = R_{\text{sk}, \text{pk}'}^f \in \mathcal{R}^f$ and for all inputs x , the statistical distance $\Delta(\text{Obf}(C)(x), C(x))$ is negligible.*

Simulatability: *There exists a PPT simulator S such that for all PPT distinguishers \mathcal{D} and security parameter λ ,*

$$|\Pr[\mathcal{D}(\text{pk}, \text{pk}', \text{Obf}(R_{\text{sk}, \text{pk}'}^f)) = 1] - \Pr[\mathcal{D}(\text{pk}, S(\text{pk})) = 1]| < \text{negl}(\lambda)$$

where $(\text{sk}, \text{pk}) \leftarrow_{\S} \text{Gen}(1^\lambda)$, $(\text{pk}', \text{sk}') \leftarrow_{\S} \text{Gen}'(1^\lambda)$, and the probabilities are taken over the coins of Gen and S .

We stress that the definition provides a very strong guarantee, in that it says that an attacker, given pk, pk' and the obfuscation $\text{Obf}(R_{\text{sk}, \text{pk}'}^f)$ does not learn *anything* beyond the public key pk of the source scheme. In particular, the simulator simulates the public key pk' . Note that the obfuscation may be a randomized circuit itself, and that the correctness requirements assumes *honest* evaluation of the circuit, i.e., using honestly generated random coins. In the full version, we also consider a stronger simulatability requirement, where pk' is generated honestly and the simulator does not learn sk' .

RELATION TO EARLIER DEFINITIONS OF OBFUSCATION. As mentioned above, previous works on re-encryption [14,9] consider a different notion of average-case obfuscation which appears at first incomparable to ours, in which the simulator must simulate $\text{Obf}(R_{\text{sk}, \text{pk}'}^f)$, given *black-box* access to $R_{\text{sk}, \text{pk}'}^f$ and knowing the public keys pk, pk' . Formally, when translated to our setting, the requirement of these earlier works is as follows:

Virtual Black-Boxness: There exists a PPT simulator S such that for all PPT distinguishers \mathcal{D} and security parameter λ ,

$$|\Pr[\mathcal{D}^{R_{\text{sk}, \text{pk}'}^f}(\text{pk}, \text{pk}', \text{Obf}(R_{\text{sk}, \text{pk}'}^f)) = 1] - \Pr[\mathcal{D}^{R_{\text{sk}, \text{pk}'}^f}(\text{pk}, \text{pk}', S^{R_{\text{sk}, \text{pk}'}^f}(\text{pk}, \text{pk}')) = 1]| < \text{negl}(\lambda)$$

where $(\text{sk}, \text{pk}) \leftarrow_{\S} \text{Gen}(1^\lambda)$, $(\text{pk}', \text{sk}') \leftarrow_{\S} \text{Gen}'(1^\lambda)$, and the probabilities are taken over the coins of Gen and S .

This definition of strong virtual black-boxness implies our obfuscation notion above for natural re-encryption functionalities, hence making it a somewhat stronger notion.

More concretely, we say that the f -re-encryption functionality $\mathcal{R}^f = \{R_{\text{sk}, \text{pk}'}^f\}$ is *simulatable* if there exists a simulator S' such that for all PPT distinguishers \mathcal{D} , we have

$$|\Pr[(\mathcal{D}^{R_{\text{sk}, \text{pk}'}^f}(\text{pk}, \text{pk}') = 1)] - \Pr[\mathcal{D}^{S'(\text{pk}, \text{pk}')}(\text{pk}, \text{pk}') = 1]| < \text{negl}(\lambda).$$

For example, the canonical re-encryption functionality is simulatable by semantic security, provided we can efficiently test if a ciphertext input to the functionality is decryptable given pk only. In the full version, we prove the following:

Lemma 1. *Assume that the obfuscator satisfies the virtual black-boxness property and the f -re-encryption functionality \mathcal{R}^f is private. Then, the obfuscator satisfies the simulatability property.*

RELATION TO FUNCTIONAL ENCRYPTION. The main result of this section connects the notions of obfuscated re-encryption with functional encryption.

Lemma 2. *Any securely obfuscatable functional re-encryption scheme for function f where the underlying public-key scheme(s) are semantically secure implies an IND-CPA-secure randomized functional encryption scheme with a single non-empty key for f -re-encryption.*

Proof (Sketch). We consider RFE defined as follows:

1. RFE.Setup(1^λ): Run $\text{PKE.Gen}(1^\lambda)$ and $\text{PKE}'.\text{Gen}(1^\lambda)$ to obtain (pk, sk) and (pk', sk') . Let $\text{Mpk} = (\text{pk}, \text{pk}')$ and $\text{Msk} = \text{sk}$. Let the message space of RFE be the message space of PKE, and the key space contain k_f (and k_e).
2. RFE.TKGen(k_f, Msk): Return $\text{Obf}(R_{\text{sk}, \text{pk}'}^f)$.
3. RFE.Enc(m, Mpk): Return $\text{PKE.Enc}(m, \text{pk})$.
4. RFE.Dec($c_1, \dots, c_n, \text{TK}, \text{Mpk}$): Return $\text{Obf}(R_{\text{sk}, \text{pk}'}^f)(c_1, \dots, c_n)$; that is, the output of the obfuscated circuit applied to the ciphertexts.

Details about the security proof are deferred to the full version. □

Combined with the results of Section 5, we therefore conclude that secure obfuscators for circuits computing RF_{NAND} and RF_{boot} imply a fully-homomorphic encryption scheme. Such obfuscations can be constructed based on the LWE assumption, starting from the encryption scheme of Regev [17]. We defer the details of this construction to the full version.

Acknowledgments. Tessaro and Wilson wish to thank Shafi Goldwasser for insightful feedback. Their work was partially supported by NSF Contract CCF-1018064 and is based on research sponsored by DARPA under agreement numbers FA8750-11-C-0096 and FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

1. Adida, B., Wikström, D.: How to Shuffle in Public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007)
2. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: new perspectives and lower bounds. Cryptology ePrint Archive, Report 2012/468 (2012)
3. Barbosa, M., Farshim, P.: On the semantic security of functional encryption schemes. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 143–161. Springer, Heidelberg (2013)
4. Bellare, M., Cash, D.: Pseudorandom functions and permutations provably secure against related-key attacks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 666–684. Springer, Heidelberg (2010)
5. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
6. Bellare, M., O’Neill, A.: Semantically-secure functional encryption: possibility results, impossibility results and the quest for a general definition. Cryptology ePrint Archive, Report 2012/515 (2012)
7. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
8. Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of Hyperplane Membership. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 72–89. Springer, Heidelberg (2010)
9. Chandran, N., Chase, M., Vaikuntanathan, V.: Functional re-encryption and collusion-resistant obfuscation. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 404–421. Springer, Heidelberg (2012)
10. Dodis, Y., Smith, A.: Correcting Errors Without Leaking Partial Information. In: Gabow, H.N., Fagin, R. (eds.) STOC 2005, pp. 654–663. ACM Press (May 2005)
11. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)
12. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Succinct functional encryption and applications: reusable garbled circuits and beyond. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) STOC 2013, pp. 555–564. ACM (2013)
13. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012)
14. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely Obfuscating Re-encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 233–252. Springer, Heidelberg (2007)
15. 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)
16. O’Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010)
17. Regev, O.: On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In: Gabow, H.N., Fagin, R. (eds.) STOC 2005, pp. 84–93. ACM Press (May 2005)
18. Russell, A.Y., Wang, H.: How to Fool an Unbounded Adversary with a Short Key. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 133–148. Springer, Heidelberg (2002)
19. Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012)
20. Wee, H.: On Obfuscating Point Functions. In: Gabow, H.N., Fagin, R. (eds.) STOC 2005, pp. 523–532. ACM Press (May 2005)

On Minimal and Quasi-minimal Linear Codes

G erard D. Cohen^{1,4}, Sihem Mesnager², and Alain Patey^{1,3,4}

¹ T el ecom ParisTech, CNRS LTCI

{gerard.cohen,alain.patey}@telecom-paristech.fr

² Department of Mathematics, University of Paris VIII, LAGA (Laboratoire Analyse, G eometrie et Applications), UMR 7539, CNRS, and University of Paris XIII, Sorbonne Paris Cit e

smesnager@univ-paris8.fr

³ Morpho

alain.patey@morpho.com

⁴ Identity and Security Alliance (The Morpho and T el ecom ParisTech Research Center)

Abstract. Minimal linear codes are linear codes such that the support of every codeword does not contain the support of another linearly independent codeword. Such codes have applications in cryptography, e.g. to secret sharing. We here study minimal codes, give new bounds and properties and exhibit families of minimal linear codes. We also introduce and study the notion of quasi-minimal linear codes, which is a relaxation of the notion of minimal linear codes, where two non-zero codewords have the same support if and only if they are linearly dependent.

Keywords: minimal codes, quasi-minimal codes, intersecting codes, secret sharing.

1 Introduction

A *minimal codeword* [15,16] c of a linear code C is a codeword such that its support (set of non-zero coordinates) does not contain the support of another linearly independent codeword. Minimal codewords are useful for defining access structures in secret sharing schemes using linear codes. Determining the set of minimal codewords is difficult for general linear codes, although this has been studied for some classes of specific linear codes. This led to work on how to find codes where all codewords are minimal, in order to facilitate the choice of access structures. The problem of finding a code satisfying this condition, called a *minimal linear code* has first been envisioned in [10] and later studied in [20,4].

Interestingly, in [4], the motivation for finding minimal linear codes is no longer secret sharing but in a new proposal for secure two-party computation, where it is required that minimal linear codes are used to ensure privacy.

It is pointed out in [4] that minimal codes are close to the notions of intersecting and separating codes [7,6]. Such codes have been suggested for applications to oblivious transfer [2], secret sharing [1,10,20] or digital fingerprinting [18].

In the binary case, the notions of intersecting and minimal linear codes coincide. Intersecting codes have already received a lot of attention [7,19,8,2,11]. For instance, [7] gives definitions, some generic constructions and non-constructive bounds on rates; [19] gives explicit constructions for small dimensions and summarizes bounds on minimum distance; [8] gives an explicit constructive sequence of intersecting codes with high rate, and so on. We will not here focus on the binary case, but on the q -ary case, where the notion of minimal codes is more restrictive than the notion of separating codes. Secret-sharing and secure two-party computations both crucially hinge on a large alphabet; thus, one cannot rely on the well-understood binary case only.

We thus pursue in Section 2 the study of [4] on bounds and criteria for minimal linear codes and exhibit families of minimal codes with better rates (but still asymptotically zero). We also exhibit in Section 3 new constructions of minimal codes using trace functions, following the works of [10,20]. Finally, in Section 4, we relax the notion of minimal codes and introduce *quasi-minimal* linear codes. Quasi-minimal linear codes are codes where two non-zero codewords have the same support if and only if they are linearly dependent. This slight relaxation enables to exhibit families with non-zero asymptotic rates.

2 Minimal Codes – Bounds and Constructions

2.1 Definitions – Notations

We denote by $|F|$ the cardinality of a set F . Let $q = p^h$, where p is a prime number and $h \in \mathbb{N}^*$. An $[n, k, d]_q$ code is a vector subspace of \mathbb{F}_q^n of dimension k with minimum Hamming distance d ; d_{max} is the maximal distance between two codewords of \mathcal{C} . Normalized parameters will be denoted by $R = k/n$, $\delta = d/n$, $\delta_{max} = d_{max}/n$; R is called the *rate* of \mathcal{C} .

The *support* of a codeword $c \in \mathcal{C}$ is the set $supp(c) = \{i \in \{1, \dots, n\} | c_i \neq 0\}$. The *Hamming weight* of a codeword $c \in \mathcal{C}$ denoted by $wt(c)$ is the cardinality of its support : $wt(c) = |supp(c)|$. A codeword c *covers* a codeword c' if $supp(c') \subset supp(c)$.

Definition 1 (Minimal codeword). [15] *A codeword c is minimal if it only covers $\mathbb{F}_q \cdot c$, i. e. if $\forall c' \in \mathcal{C}, (supp(c') \subset supp(c)) \implies (c, c')$ linearly dependent.*

Definition 2 (Minimal linear code). [10] *A linear code \mathcal{C} is minimal if every non-zero codeword $c \in \mathcal{C}$ is minimal.*

A code \mathcal{C} is *intersecting* if $\forall c \neq 0, c' \neq 0 \in \mathcal{C}, supp(c) \cap supp(c') \neq \emptyset$. A code \mathcal{C} is *t -intersecting* if $\forall c \neq 0, c' \neq 0 \in \mathcal{C}, |supp(c) \cap supp(c')| \geq t$

For a complete treatment of coding theory, we refer to the book of MacWilliams and Sloane [14].

2.2 Generic Bounds

Two non-constructive bounds on the rates of minimal codes are exhibited in [4]. We recall them with their proofs. Notice that these constructions are more demanding as q grows.

Theorem 1 (Maximal Bound). [4] *Let \mathcal{C} a minimal linear $[n, k, d]$ q -ary code, then, asymptotically, $R \leq \log_q(2)$.*

Proof. This bound is even true for non-linear minimal codes. Let us consider the family F of supports of the vectors of \mathcal{C} . By definition of minimal codes, this is a Sperner family. It is known that $|F| \leq \binom{n}{n/2}$. Thus, $|\mathcal{C}| = q^k \leq 1 + (q-1)\binom{n}{n/2}$ and $R = k/n \leq \log_q(2) + o(1)$.

Theorem 2 (Minimal Bound). [4]

For any R , $0 \leq R = k/n \leq \frac{1}{2} \log_q\left(\frac{q^2}{q^2-q+1}\right)$, there exists an infinite sequence of $[n, k]$ minimal linear codes.

Proof. The proof is similar to the one of [7] in the binary case. Let us fix n and k . For $a \in \mathbb{F}_q^n$, such that $|supp(a)| = i$, there are $q^i - q$ linearly independent vectors b such that $supp(b) \subset supp(a)$. The pair (a, b) belongs to $\begin{bmatrix} n-2 \\ k-2 \end{bmatrix}$ linear $[n, k]$ codes, where $\begin{bmatrix} x \\ k \end{bmatrix}$ denotes the q -ary Gaussian binomial coefficient.

There are less than $\sum_{i=0}^n \binom{n}{i} (q-1)^i (q^i - q) = (1 + (q-1)q)^n - q^{n+1} \leq (q^2 - q + 1)^n$ such ordered “bad” (a, b) pairs. At least $\begin{bmatrix} n \\ k \end{bmatrix} - \begin{bmatrix} n-2 \\ k-2 \end{bmatrix} (q^2 - q + 1)^n$ linear $[n, k]$ codes thus contain no “bad” pairs, *i. e.* are minimal. For $k/n \leq \frac{1}{2} \log_q\left(\frac{q^2}{q^2-q+1}\right)$, this quantity is positive.

2.3 Minimal Codes and Intersecting Codes

Proposition 1. *A minimal linear code \mathcal{C} is intersecting.*

Proof. Let c, c' be two codewords such that $supp(c) \cap supp(c') = \emptyset$. We have $supp(c) \subset supp(c + c')$ and $supp(c') \subset supp(c + c')$. Thus, c and $c + c'$ are linearly dependent, c' and $c + c'$ are linearly dependent; hence c and c' are linearly dependent. Since $supp(c) \cap supp(c') = \emptyset$, at least one of c, c' is equal to zero, thus \mathcal{C} is intersecting. □

The converse is true in the binary case (only).

Proposition 2. *A binary intersecting linear code \mathcal{C} is minimal.*

Proof. Let \mathcal{C} be a binary linear code. Let us assume that there exist two nonzero codewords $c \neq c'$ with $supp(c) \subset supp(c')$. The inclusion is strict since two different binary codewords cannot share the same support. the support of $c + c'$ does not intersect with the support of c . Hence, a non-minimal code is not intersecting. □

The condition of minimality is more demanding than that of intersection, and the more so when q increases. This fact is captured by the next result (which also proves that the only case where the converse of Proposition 1 is true is the binary case).

Proposition 3. *A minimal $[n, k, d]_q$ code is $(q - 1)$ -intersecting, if $k \geq 2$.*

Proof. Let c, c' be two linearly independent codewords. One can write by blocks, w.l.o.g., $c = 0||X||0||Z$ and $c' = 0||0||Y||Z'$, where all X, Y, Z, Z' do not contain any zeros, $|X| \geq 1$, $|Y| \geq 1$ and $|Z| = |Z'| \geq 1$ (minimality). Let $\lambda \in \mathbb{F}_q^*$, $c + \lambda c' = 0||X||\lambda Y||Z + \lambda Z'$ is independent of c and of c' , consequently it should not cover either c or c' . Thus, there exists i_λ such that $z_{i_\lambda} + \lambda z'_{i_\lambda} = 0$. This must be true for any $\lambda \in \mathbb{F}_q^*$. Since all coordinates of Z and Z' are non-zero, one cannot have $i_\lambda = i_\mu$, for $\lambda \neq \mu$. Consequently $|supp(c) \cap supp(c')| \geq |\mathbb{F}_q| = q - 1$. In particular, the minimum weight d of a nonzero codeword is at least $(q - 1)$ and two linearly dependent nonzero codewords also intersect in at least $q - 1$ positions. Thus \mathcal{C} is $(q - 1)$ -intersecting. \square

Example 1 (Simplex Code). The shortest minimal codes of dimension 2 have length $q + 1$.

For instance, consider the simplex code $\mathcal{S}_{q,k}[(q^k - 1)/(q - 1), k, q^{k-1}]_q$, where the generator matrix's columns are a complete set of pairwise linearly independent vectors.

For $k = 2$, it is a $\mathcal{S}_{q,2}[q + 1, 2, q]$ code with generator matrix $\begin{pmatrix} 1 & 0 & 1 & \dots & 1 \\ 0 & 1 & \alpha_1 & \dots & \alpha_{q-1} \end{pmatrix}$, where $\alpha_1, \dots, \alpha_{q-1}$ are all the nonzero elements of \mathbb{F}_q .

Corollary 1. *Let \mathcal{C} be a minimal $[n, k, d]_q$ code, then*

$$d \geq k + q - 2$$

Proof. The projection on a codeword with minimal weight gives a $[d, k, d' \geq (q - 1)]$ code (see the proof of Proposition 3). The Singleton bound now implies $d \geq k + d' - 1$, thus $d \geq k + q - 2$. \square

Proposition 4. *Let \mathcal{C} be a minimal $[n, k, d]_q$ code with maximal distance d_{max} . Then*

$$d_{max} \leq n - k + 1$$

Proof. Consider a codeword c_{max} of weight d_{max} and the projection of \mathcal{C} on its zero coordinates, i. e. on $\{1, \dots, n\} \setminus supp(c_{max})$. It is a linear operation, whose kernel has dimension 1 (since c_{max} is minimal), so its rank is $k - 1$ and $k - 1 \leq n - d_{max}$. \square

Notice that the bounds given by the three previous results are all tight: to see this, consider the code $\mathcal{S}_{q,2}[q + 1, 2, q]$ of Example 1.

2.4 Constructions

We now give a construction based on the Kronecker product of codes. which yields infinite families of minimal codes with relatively slowly decreasing rates.

Proposition 5. *The product $\mathcal{C}_1 \otimes \mathcal{C}_2$ of a minimal $[n_1, k_1, d_1]_q$ code \mathcal{C}_1 and of a minimal $[n_2, k_2, d_2]_q$ code \mathcal{C}_2 is a minimal $[n_1 \times n_2, k_1 \times k_2, d_1 \times d_2]_q$ code.*

Proof. Let $c \neq 0, c'$ be two codewords of $\mathcal{C}_1 \otimes \mathcal{C}_2$. They can both be written as $n_1 \times n_2$ matrices where rows are codewords of \mathcal{C}_1 and columns are codewords of \mathcal{C}_2 . Let us assume that c covers c' . For $i = 1, \dots, n_1, j = 1, \dots, n_2$ let c_i^1 (resp. $c_i'^1$) be the i^{th} row of c (resp. c') and c_j^2 (resp. $c_j'^2$) be the j^{th} column of c (resp. c'). For every i, c_i^1 covers $c_i'^1$, so $\exists \lambda_i$ such that $c_i'^1 = \lambda_i c_i^1$. With the same reasoning on the columns, for every j , there exists λ_j such that $c_j'^2 = \lambda_j c_j^2$. Then, all the λ_i 's and λ_j 's are equal and there exists λ such that $c' = \lambda c$, so c and c' are linearly dependent. Thus, $\mathcal{C}_1 \otimes \mathcal{C}_2$ is minimal. □

Example 2. For $q = 3, k = 2$, the associated simplex $\mathcal{S}_{3,2}$ is the celebrated $[4, 2, 3]_3$ tetracode T. T is self-dual, both a simplex and a Hamming code. Its (Kronecker) square is T^2 , a $[16, 4, 9]_3$ minimal code. More generally, the square of the $[q+1, 2, q]_q$ simplex code is a $[(q+1)^2, 4, q^2]_q$ minimal code. Repeating the process, we obtain $[(q+1)^\ell, 2^\ell, q^\ell]_q$ minimal codes, with rate $R := k/n = (2/(q+1))^\ell$.

There exists a sufficient condition on weights for a given linear code to be minimal. More precisely, if the weights of a linear code are close enough to each other, then each nonzero codeword of the code is a minimal vector as described by the following statement.

Proposition 6. *[10] Let \mathcal{C} be an $[n, k, d]$ code. Let d and d_{max} be the minimum and maximum nonzero weights respectively. If $\frac{d}{d_{max}} > \frac{q-1}{q}$ then \mathcal{C} is minimal.*

Remark 1. Note that the previous condition is only necessary. Indeed, the square of the tetracode is $T^2[16, 4, d = 9, d_{max} = 12]$. To see this, take as a basis for T $c^1 = 1011, c^2 = 0112$, giving

$$G = H = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{pmatrix}$$

Then $c^1, c^2, c^3 = c^1 + c^2, c^4 = c^1 + 2c^2$ is a codeword A of T^2 of weight 12:

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 2 & 0 \\ 1 & 2 & 0 & 2 \end{pmatrix}$$

Consider now $T^4[256, 16, 81]$, the square of T^2 . It is easy to check that $A \otimes A \in T^2 \otimes T^2 = T^4$ has weight 144:

$$\begin{pmatrix} A & 0 & A & A \\ 0 & A & A & 2A \\ A & A & 2A & 0 \\ A & 2A & 0 & 2A \end{pmatrix}$$

Thus $d_{max}(T^4) \geq 144$ and for this minimal code, $d/d_{max} \leq 81/144 < (q - 1)/q = 2/3$.

Remark 2. Note that the easier to check sufficient condition $\frac{d}{n} > \frac{q-1}{q}$ is too strong to get asymptotically good codes; indeed, by the Plotkin bound ([14], for any code, not necessarily linear, of length n , size M and distance d , if $d > (q - 1)n/q$, then $M \leq d/(d - (1 - q^{-1}))$).

Plotkin bound is tight, achieved with equality by simplex codes $\mathcal{S}_{q,k}[(q^k - 1)/(q - 1), k, q^{k-1}]$.

On the other hand, for $\delta < 1 - q^{-1}$, the classical Varshamov-Gilbert bound [12] guarantees the existence of asymptotic families of codes with non zero rate $R(\delta, q)$. We shall come back to that later.

Example 3. The sufficient condition exposed in Proposition 6 enables to prove the minimality of several known codes. Many examples come from the codes with a limited number of weights. For instance, in [22], one can find 3-weight codes with parameters $[26, 6, 15]_3$ or $[124, 6, 90]_5$ that satisfy the sufficient condition and that have better rates than simplex codes (respectively $\mathcal{S}_{3,4}$ and $\mathcal{S}_{5,4}$).

Similarly, the $[39, 4, 28]_5$ 4-weight code exposed in [9] also meets the condition and beats the $\mathcal{S}_{5,4}$ simplex code.

3 Constructions of Minimal Linear Codes via Trace Functions

Let p be a prime, m be a positive integer and h be a positive integer, divisor of m . Set $m = hr$. and $q = p^h$.

Definition 3 (Trace function over \mathbb{F}_{q^r}).

The trace function $Tr_{q^r/q} : \mathbb{F}_{q^r} \rightarrow \mathbb{F}_q$ is defined as:

$$Tr_{q^r/q}(x) := \sum_{i=0}^{r-1} x^{q^i} = x + x^q + x^{q^2} + \dots + x^{q^{r-1}}$$

The trace function from \mathbb{F}_{q^r} to its prime subfield is called the absolute trace function.

Recall that the trace function $Tr_{q^r/q}$ is \mathbb{F}_q -linear and satisfies the transitivity property in a chain of extension fields ($m = hr$): $Tr_{p^m/p}(x) = Tr_{p^h/p}(Tr_{p^m/p^h}(x))$ for all $x \in \mathbb{F}_{q^r}$.

Given a Boolean function f defined on \mathbb{F}_{2^n} (that is, a mapping from \mathbb{F}_{2^n} to \mathbb{F}_2), the Walsh transform of f is the discrete Fourier transform of the sign function of f that is, $\chi(f) := (-1)^f$ where χ is the canonical additive character. The Walsh transform of f denoted by $\widehat{\chi}_f$ is defined as:

$$\widehat{\chi}_f(a) = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{f(x) + x \cdot a}, \forall a \in \mathbb{F}_{2^n}$$

where " \cdot " denotes a scalar product in \mathbb{F}_{2^n} . The mapping $(x, y) \mapsto Tr_{2^n/2}(xy)$ defines an inner (scalar) product on \mathbb{F}_{2^n} . Finally, a Boolean function f on \mathbb{F}_{2^n} (n even) is bent if and only if its Walsh transform satisfies $\widehat{\chi}_f(a) = \pm 2^{\frac{n}{2}}$ for all $a \in \mathbb{F}_{2^n}$. The dual \tilde{f} of a bent function f is defined by the relation $\widehat{\chi}_f(\omega) = 2^{\frac{n}{2}}(-1)^{\tilde{f}(\omega)}, \forall \omega \in \mathbb{F}_{2^n}$.

3.1 A Construction of a Class of q -ary Linear Minimal Codes

For any $\alpha, \beta \in \mathbb{F}_{p^m}$, define a q -ary function $f_{\alpha, \beta}$ as:

$$\begin{aligned} f_{\alpha, \beta} : \mathbb{F}_{q^r} &\longrightarrow \mathbb{F}_q \\ x &\longmapsto f_{\alpha, \beta}(x) := Tr_{q^r/q}(\alpha\Psi(x) + \beta x) \end{aligned}$$

where Ψ is a mapping from \mathbb{F}_{q^r} to \mathbb{F}_{q^r} such that $\Psi(0) = 0$. We now define a linear code \mathcal{C}_Ψ over \mathbb{F}_q as :

$$\mathcal{C}_\Psi := \{ \bar{c}_{\alpha, \beta} = (f_{\alpha, \beta}(\zeta_1), f_{\alpha, \beta}(\zeta_2), \dots, f_{\alpha, \beta}(\zeta_{q^r-1})), \alpha, \beta \in \mathbb{F}_{q^r} \}$$

where $\zeta_1, \dots, \zeta_{q^r-1}$ denote the nonzero elements of \mathbb{F}_{q^r} .

Proposition 7. *The linear code \mathcal{C}_Ψ is of length $q^r - 1$ and dimension k with $k = \frac{2m}{h} = 2r$ if the mapping Ψ has no linear components, and $k < 2r$ otherwise.*

Proof. It is clear that \mathcal{C}_Ψ is of length $q^r - 1$. Now, compute the cardinality of \mathcal{C}_Ψ . Let $\bar{c}_{\alpha, \beta}$ be a codeword of \mathcal{C}_Ψ . We have

$$\begin{aligned} \bar{c}_{\alpha, \beta} = 0 &\iff Tr_{q^r/q}(\alpha\Psi(\zeta_i) - \beta\zeta_i) = 0, \forall i \in \{1, \dots, q^r - 1\} \\ &\iff Tr_{q^r/q}(\alpha\Psi(x) - \beta x) = 0, \forall x \in \mathbb{F}_{q^r}^* \\ &\Rightarrow Tr_{q^r/p}(\alpha\Psi(x) - \beta x) = 0, \forall x \in \mathbb{F}_{q^r}^* \\ &\Rightarrow Tr_{q^r/p}(\alpha\Psi(x) - \beta x) = 0, \forall x \in \mathbb{F}_{q^r} \\ &\Rightarrow Tr_{q^r/p}(\alpha\Psi(x)) = Tr_{q^r/p}(\beta x), \forall x \in \mathbb{F}_{q^r} \end{aligned}$$

Hence, $\bar{c}_{\alpha, \beta} = 0$ implies that the mapping from \mathbb{F}_{q^r} to \mathbb{F}_q , that is, a component of Ψ associated to $\alpha \neq 0$, is linear (or null) and coincides with $x \mapsto Tr_{q^r/p}(\beta x)$. Therefore, it suffices that no component function of Ψ is identically equal to 0 or linear to ensure that the only null codeword appears only one time at $\alpha = \beta = 0$. Furthermore, this implies that all the codewords $\bar{c}_{\alpha, \beta}$ are pairwise distinct. In this case, the size of the code is q^{2r} and the dimension of the code is thus $2r$. \square

Assume p is an odd prime. Choose Ψ a perfect nonlinear mapping, that is, Ψ is such that $\max_{a \in \mathbb{F}_{q^r}^*} \min_{b \in \mathbb{F}_q} |D_a \Psi^{-1}(b)| = \frac{q^r - 1}{q^r}$ where $D_a \Psi(x)$ denotes the derivatives of Ψ defined by $D_a \Psi(x) := \Psi(x + a) + \Psi(x)$. According to [3], if $q < \frac{q^{r/2} + 1}{2}$, then (using the sufficient condition given in Proposition 6) \mathcal{C}_Ψ is a minimal $[q^r - 1, 2r, d > \frac{q-1}{q}(q^r - q^{r/2})]$ -code.

3.2 A Construction of a Class of Linear Minimal 2^h -ary Codes

The previous construction of minimal codes is valid when p is an odd prime. In this subsection, we provide a construction of minimal codes in the case where

$p = 2$. To this end, let m be a positive integer and h a divisor of m . Set $r := \frac{m}{h}$. We define two sets E and R of $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ as follows:

$$E := \{(x, 0), x \in \mathbb{F}_{2^m}\},$$

and

$$R := \{(0, y), y \in \mathbb{F}_{2^m}\}.$$

Set

$$\Gamma := \mathbb{F}_{2^m} \times \mathbb{F}_{2^m} \setminus (E \cup R) = \{(\delta_i, \zeta_i), 1 \leq i \leq (2^m - 1)^2\}.$$

For any $a \in \mathbb{F}_{2^m}$, we define the function Φ_a as

$$\begin{aligned} \Phi_a : \Gamma &\longrightarrow \mathbb{F}_{2^h} \\ (x, y) &\longmapsto \Phi_a(x, y) := Tr_{2^m/2^h}(ax^{2^{m+1}-3}y^2) \end{aligned}$$

We now define a linear code \mathcal{C} over \mathbb{F}_{2^h} as :

$$\mathcal{C} := \{\bar{c}_a = (\Phi_a(\delta_1, \zeta_1), \dots, \Phi_a(\delta_{(2^m-1)^2}, \zeta_{(2^m-1)^2})), a \in \mathbb{F}_{2^m}\}$$

It is clear that the code \mathcal{C} is of length $(2^m - 1)^2$. The following statement provides the weight distribution of \mathcal{C} .

Proposition 8. *The linear code \mathcal{C} is a one-weight minimal code. More precisely, every non-zero codeword has Hamming weight $2^{m-h}(2^h - 1)(2^m - 1)$.*

Proof. For $\omega \in \mathbb{F}_{2^m}^*$, denote by $\psi_{a\omega}$ the Boolean function defined as follows:

$$\begin{aligned} \psi_{a\omega} : \mathbb{F}_{2^m} \times \mathbb{F}_{2^m} &\longrightarrow \mathbb{F}_2 \\ (x, y) &\longmapsto \psi_{a\omega}(x, y) := Tr_{2^m/2}(a\omega x^{2^{m+1}-3}y^2) \end{aligned}$$

Thanks to [17], the Walsh transform of $\psi_{a\omega}$ can be computed as well as its dual function. For every $a \neq 0$, we have:

$$\widehat{\chi\psi_{a\omega}}(z, t) = 2^m(-1)^{Tr_{2^m/2}(a\omega zt^{-2})}, \forall (z, t) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$$

This result implies that the function $\psi_{a\omega}$ is bent (since its Walsh transform takes only the values $\pm 2^m$) and that its dual equals $\widetilde{\psi_{a\omega}}$ defined by $\widetilde{\psi_{a\omega}}(z, t) = Tr_{2^m/2}(a\omega zt^{-2}), \forall (z, t) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$. In particular, for $a \in \mathbb{F}_{2^m}^*$ and $\omega \in \mathbb{F}_{2^m}^*$, $\widehat{\chi\psi_{a\omega}}(0, 0) = 2^m$. Now, let us compute the value of the sum $\sum_{\omega \in \mathbb{F}_{2^h}^*} \widehat{\chi\psi_{a\omega}}(0, 0)$ over the subfield \mathbb{F}_{2^h} of \mathbb{F}_{2^m} in two ways. On the one hand, thanks to the the above expression of the Walsh transform, we get:

$$\begin{aligned} \sum_{\omega \in \mathbb{F}_{2^h}^*} \widehat{\chi\psi_{a\omega}}(0, 0) &= \widehat{\chi\psi_0}(0, 0) + \sum_{\omega \in \mathbb{F}_{2^h}^*} \widehat{\chi\psi_{a\omega}}(0, 0) \\ &= 2^{2m} + 2^m(2^h - 1). \end{aligned}$$

On the other hand, using the transitivity rule of the trace function and the \mathbb{F}_{2^h} -linearity of the trace function $Tr_{2^m/2^h}$, we have:

$$\begin{aligned}
 \sum_{\omega \in \mathbb{F}_{2^h}} \widehat{\chi}_{a\omega}(0,0) &= \sum_{\omega \in \mathbb{F}_{2^h}} \sum_{x \in \mathbb{F}_{2^m}} \sum_{y \in \mathbb{F}_{2^m}} (-1)^{\psi_{a\omega}(x,y)} \\
 &= \sum_{x \in \mathbb{F}_{2^m}} \sum_{y \in \mathbb{F}_{2^m}} \sum_{\omega \in \mathbb{F}_{2^h}} (-1)^{Tr_{2^m/2^h}(a\omega x^{2^{m+1}-3}y^2)} \\
 &= \sum_{x \in \mathbb{F}_{2^m}} \sum_{y \in \mathbb{F}_{2^m}} \sum_{\omega \in \mathbb{F}_{2^h}} (-1)^{Tr_{2^h/2}(Tr_{2^m/2^h}(a\omega x^{2^{m+1}-3}y^2))} \\
 &= \sum_{x \in \mathbb{F}_{2^m}} \sum_{y \in \mathbb{F}_{2^m}} \sum_{\omega \in \mathbb{F}_{2^h}} (-1)^{Tr_{2^h/2}(Tr_{2^m/2^h}(ax^{2^{m+1}-3}y^2)\omega)} \\
 &= \sum_{(x,y) \in \mathbb{F}_{2^m}^2 | Tr_{2^m/2^h}(ax^{2^{m+1}-3}y^2)=0} 2^h \\
 &= 2^h \#\{(x,y) \in \mathbb{F}_{2^m}^2 \mid Tr_{2^m/2^h}(ax^{2^{m+1}-3}y^2) = 0\} \\
 &= 2^h \left(2^{2m} - \#\{(x,y) \in \mathbb{F}_{2^m}^2 \mid Tr_{2^m/2^h}(ax^{2^{m+1}-3}y^2) \neq 0\} \right) \\
 &= 2^h \left(2^{2m} - \#\{(x,y) \in \mathbb{F}_{2^m}^2 \mid \Phi_a(x,y) \neq 0\} \right) \\
 &= 2^h \left(2^{2m} - \#\{(x,y) \in \Gamma \mid \Phi_a(x,y) \neq 0\} \right) \\
 &= 2^{2m+h} - 2^h wt(\bar{c}_a).
 \end{aligned}$$

Hence, we have the following equality:

$$2^{2m+h} - 2^h wt(\bar{c}_a) = 2^{2m} + 2^m(2^h - 1)$$

from which we deduce the Hamming weight of any non-zero codeword of \mathcal{C} : $wt(\bar{c}_a) = 2^{2m} - 2^{2m-h} - 2^m + 2^{m-h} = 2^{m-h}(2^h - 1)(2^m - 1)$.

According to the previous result, the code \mathcal{C} is of constant weight. The structure of linear codes of constant weight is well-known. In fact, it has been proved that such codes are equivalent to simplex codes.

The next theorem ([5], page 363) characterizes all the q -ary linear codes with constant weight in terms of simplex codes and therefore defines the structure of the code \mathcal{C} .

Theorem 3. ([5]) *If all the nonzero codewords of a q -ary $[n, k]$ -code have the same weight and no coordinate identically zero, then the code has a generator matrix of the form (G_1, G_2, \dots, G_t) , where each G_i is a generator matrix of the k -dimensional simplex code $\mathcal{S}_{q,k}$ over \mathbb{F}_q .*

Therefore, we deduce that the code \mathcal{C} defined explicitly above is a minimal code equivalent to a $(2^m - 1)(2^h - 1)$ -multiple of the 2^h -ary simplex code $\mathcal{S}_{2^h, \frac{m}{h}}$.

The code \mathcal{C} has a generator matrix of the form $(G_1, G_2, \dots, G_{(2^m-1)(2^h-1)})$, where each G_i is a generator matrix of the $\frac{m}{h}$ -dimensional simplex code over \mathbb{F}_2^h , (that is, a $\frac{m}{h} \times (\frac{2^m-1}{2^h-1})$ matrix whose columns are pairwise linearly independent).

Note that one can generalize the previous construction and prove the following result.

Proposition 9. *Let i be a positive integer co-prime with m . For any $a \in \mathbb{F}_{2^m}$, we define the function Φ_a^i as*

$$\begin{aligned} \Phi_a^i : \Gamma &\longrightarrow \mathbb{F}_{2^h} \\ (x, y) &\longmapsto \Phi_a^i(x, y) := Tr_{2^m/2^h}(ax^{2^{m+i}-2^{i+1}+1}y^{2^i}) \end{aligned}$$

Define $\mathcal{C}^i := \{\bar{c}_a = (\Phi_a^i(\delta_1, \zeta_1), \dots, \Phi_a^i(\delta_{(2^m-1)^2}, \zeta_{(2^m-1)^2})), a \in \mathbb{F}_{2^m}\}$. Then, the linear code \mathcal{C}^i over \mathbb{F}_{2^h} is a minimal code with parameters $[(2^m-1)^2, \frac{m}{h}, 2^{m-h}(2^h-1)(2^m-1)]$.

4 Quasi-minimal Codes

As we have seen in the previous sections, we still have no construction of minimal codes with asymptotic nonzero rate. To obtain such constructions, we slightly relax the notion of minimal codes to the new notion of *quasi-minimal* codes. Minimal codes prevent a codeword to have its support included in the support of a linearly independent codeword, whereas quasi-minimal codes prevent a codeword to have the same support as a linearly independent codeword.

We will see that this new setting, although it also brings intersection properties, allows constructions with nonzero asymptotic rates.

4.1 Definitions and Properties

Definition 4 (Quasi-minimal codeword). *A codeword c is quasi-minimal if $\forall c' \in \mathcal{C}, (\text{supp}(c') = \text{supp}(c)) \implies (c, c')$ linearly dependent.*

Definition 5 (Quasi-minimal linear code). *A linear code \mathcal{C} is quasi-minimal if every non-zero codeword $c \in \mathcal{C}$ is quasi-minimal.*

Quasi-minimality is clearly a weaker requirement than minimality. For instance, every binary code is obviously quasi-minimal. Still, these codes do enjoy intersection properties.

Theorem 4. *If \mathcal{C} is quasi-minimal with $n \geq q-2, k \geq 2, q \geq 3$, then it is $(q-2)$ -intersecting.*

Proof. Suppose \mathcal{C} minimal and $c, c' \in \mathcal{C}$ with support intersection of size $s \leq q-3$. W.l.o.g., one can write by blocks $c = 0||X||0||Z$ and $c' = 0||0||Y||Z'$ with $|Z| = |Z'| = s$, and where Z and Z' do not contain any zeros.

Then at most s elements $\lambda_i \in F_q^*$ can make $|supp(Z) \cap supp(Z + \lambda_i Z')| < s$. If $s \leq q - 3$, there are at least two nonzero field elements left, say α and β , such that $Z + \alpha Z'$ and $Z + \beta Z'$ are independent and have the same support. Moreover $c + \alpha c'$ and $c + \beta c'$ will also share the same support and be linearly independent, which contradicts the minimality of C . Hence, $s > q - 3$. \square

We now prove a sufficient condition for quasi-minimality, weaker than the one for minimality. This relaxation will then allow us to construct infinite classes of asymptotically good quasi-minimal codes by concatenation.

Theorem 5 (Sufficient condition for quasi-minimality). *Let C be a linear $[n, k, d]_q$ code; if $d/n > (q - 2)/(q - 1)$, then C is quasi-minimal.*

Proof. Let C be a linear $[n, k, d]_q$ code and let c, c' be two linearly independent codewords of C such that $supp(c) = supp(c')$. Let α be a primitive element of F_q . Then, w.l.o.g., one can write c and c' by blocks, in the following way: $c = \beta_0 || \dots || \beta_{q-2} || 0$ and $c' = \alpha^0 \beta_0 || \dots || \alpha^{q-2} \beta_{q-2} || 0$. Let A_i be the size of the (possibly empty) block β_i . Then $wt(c) = wt(c') = \sum_{i=0}^{q-2} A_i \geq d$. We also have, for $j = 0, \dots, q - 2$, $d(\alpha^j c, c') = \sum_{i \neq j} A_i \geq d$. If we sum all these inequalities, we get $(q - 2) \sum_{i=0}^{q-2} A_i \geq (q - 1)d$, hence $wt(c) \geq \frac{q-1}{q-2}d$. Thus, if $n < \frac{q-1}{q-2}d$, $wt(c) > n$, which is impossible, so c and c' cannot exist and C is quasi-minimal. \square

Now, the celebrated non-constructive Varshamov-Gilbert bound implies the existence of infinite families of semi-constructive codes with rate $R = 1 - h_q(\frac{q-2}{q-1}) > 0$. Estimations of this rate are given in Table 1. This is still far from the upper bound, derived analogously to the minimal case:

Theorem 6 (Maximal Bound). *Let C be a quasi-minimal linear $[n, k, d]_q$ code, then, asymptotically, $R \leq \log_q(2)$.*

Proof. This bound is even true for non-linear quasi-minimal codes. Consider the family F of the supports of the vectors of C . Clearly, $|F| \leq 2^n$. Thus, $|C| = q^k \leq 1 + (q - 1)2^n$ and $R = k/n \leq \log_q(2) + o(1)$.

Table 1. Estimations of the rates of semi-constructive codes

q	2	3	4	5	7
Rate of the semi-constructive code	1	0.053	0.013	0.0046	0.0011
Upper bound	1	0.63	0.5	0.43	0.36

4.2 Infinite Constructions

The general idea is to concatenate a q -ary “seed” or inner code (e.g. a simplex) with an infinite family of algebraic-geometric (AG) codes (the outer codes) [21], in such a way as to obtain a high enough minimum distance and conclude by Theorem 5.

In practice, we can take the seed to be $\mathcal{S}_{q,r}[n = (q^r - 1)/(q - 1), k = r, d = q^{r-1}]_q$ (with $\delta > (q - 1)/q$), set $r = 2m$ and concatenate with $AG[N, K = NR, D = N\Delta]_{q^{2m}}$. These codes exist lying almost on the Singleton bound, namely satisfying $R + \Delta = 1 - (q^m - 1)^{-1}$.

This concatenation results in the family $C[nN, kK, dD]_q$. If $dD/nN = \delta\Delta > (q - 2)/(q - 1)$, this family is quasi-minimal by Theorem 5.

It is not hard to check that, for example, choosing q large enough, $m \geq 2$, $\Delta = (q^m - q)/(q^m - 1)$, $R = (q - 2)/(q^m - 1)$, this is the case.

Example 4 (Small examples).

- Take $q = 4$, $\mathcal{S}_{4,4}[85, 4, 64]_4$, $\Delta = 9/10$, $R = 1/30$, resulting in an infinite construction of $[n, 2n/1275]$ quaternary codes.
- Take $q = 3$, $C[15, 4, 9]_3$ [13] as inner code and $AG[N, NR, N\Delta]_{3^4}$ with $R + \Delta = 7/8$. Choose $\Delta = 41/48$, $R = 1/48$; then $\Delta\delta = 41/80$ and by Theorem 5 the concatenation is an infinite construction of quasi-minimal $[n, n/180, 41n/80]$ ternary codes.

Concluding Remarks. We can prove, non-constructively, the existence of infinite families of codes with $\delta_{max} := d_{max}/n < 1 - \omega$, for some fixed $0 < \omega$.

To do so, observe that any nonzero n -tuple belongs to $\begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$ linear $[n, k]$ codes, i.e. a fraction $\approx q^{n-k}$ of their total number.

Fix ω , $0 < \omega < 1 - q^{-1}$, $w := \omega n$.

The number of q -ary n -tuples of weight at least $n - w$ is $\sum_{i=0}^w \binom{n}{i} (q - 1)^{n-i} \approx \binom{n}{w} (q - 1)^{n-w} \approx 2^{nh(\omega)} (q - 1)^{n(1-\omega)}$, where $h(\cdot)$ is the binary entropy function. As in the proof of the previous theorem, if $R := R(q, \omega = \epsilon(q)) \leq 1 - h(\omega) \log_q 2 - (1 - \omega) \log_q (q - 1)$, then the number of “bad” vectors is negligible and there exist codes with (in fact almost all codes have) rate R and no high-weight vector (of weight larger than $n(1 - \omega)$).

Now, take a code on the Varshamov-Gilbert bound (again, almost all codes are), with $\delta = 1 - q^{-1} - \alpha$ and rate $R(q, \alpha) > 0$, with $\alpha = \alpha(\omega)$ small enough so that $\delta/\delta_{max} > (1 - q^{-1} - \alpha)/(1 - \omega) > 1 - q^{-1}$; this code will necessarily be minimal.

To summarize, for a small enough rate $R = R(q)$, there exist infinite families of codes satisfying $\delta/\delta_{max} > (q - 1)/q$, thus minimal. Note that, by the Plotkin bound, they necessarily satisfy $\delta < (q - 1)/q$, so the fact that $\delta_{max} < 1$ is crucial.

Open Problems. We saw that obtaining explicit constructions of minimal binary linear codes with asymptotically non zero rates can be done using known

techniques (e.g. [7,8]). We can however not use the same techniques in the q -ary case, where obtaining minimal linear codes, with asymptotically non zero rates, remains an open issue. Finding such codes might be done using quasi-minimal linear codes, it would thus be interesting to find a condition for minimality specific to quasi-minimal codes.

Acknowledgements. This work was partially done during the French FUI-12 RESILIENCE project that is funded by DGCIS.

References

1. Ashikhmin, A.E., Barg, A.: Minimal vectors in linear codes. *IEEE Transactions on Information Theory* 44(5), 2010–2017 (1998)
2. Brassard, G., Crépeau, C., Santha, M.: Oblivious transfers and intersecting codes. *IEEE Transactions on Information Theory* 42(6), 1769–1780 (1996)
3. Carlet, C., Ding, C., Yuan, J.: Linear codes from perfect nonlinear mappings and their secret sharing schemes. *IEEE Transactions on Information Theory* 51(6), 2089–2102 (2005)
4. Chabanne, H., Cohen, G., Patey, A.: Towards Secure Two-Party Computation from the Wire-Tap Channel. *ArXiv e-prints* (June 2013)
5. Cohen, G., Honkala, I., Litsyn, S., Lobstein, A.: *Covering codes*. North Holland (1997)
6. Cohen, G.D., Encheva, S.B., Litsyn, S., Schaathun, H.G.: Intersecting codes and separating codes. *Discrete Applied Mathematics* 128(1), 75–83 (2003)
7. Cohen, G.D., Lempel, A.: Linear intersecting codes. *Discrete Mathematics* 56(1), 35–43 (1985)
8. Cohen, G.D., Zémor, G.: Intersecting codes and independent families. *IEEE Transactions on Information Theory* 40(6), 1872–1881 (1994)
9. Ding, C.: A class of three-weight and four-weight codes. In: Chee, Y.M., Li, C., Ling, S., Wang, H., Xing, C. (eds.) *IWCC 2009*. LNCS, vol. 5557, pp. 34–42. Springer, Heidelberg (2009)
10. Ding, C., Yuan, J.: Covering and secret sharing with linear codes. In: Calude, C.S., Dinneen, M.J., Vajnovszki, V. (eds.) *DMTCS 2003*. LNCS, vol. 2731, pp. 11–25. Springer, Heidelberg (2003)
11. Encheva, S.B., Cohen, G.D.: Constructions of intersecting codes. *IEEE Transactions on Information Theory* 45(4), 1234–1237 (1999)
12. Gilbert, E.N.: A comparison of signalling alphabets. *Bell System Technical Journal* 31(3), 504–522 (1952)
13. van Lint, J.H., Schrijver, A.: Construction of strongly regular graphs, two-weight codes and partial geometries by finite fields. *Combinatorica* 1(1), 63–73 (1981)
14. MacWilliams, F.J., Sloane, N.J.: *The theory of error-correcting codes*. North-Holland, Amsterdam (1977)
15. Massey, J.L.: Minimal codewords and secret sharing. In: *Proc. 6th Joint Swedish-Russian Int. Workshop on Info. Theory*, pp. 276–279 (1993)
16. Massey, J.L.: Some applications of coding theory in cryptography. In: Farrell, P.G. (ed.) *Codes and Cyphers: Cryptography and Coding IV*, pp. 33–47. Formara Ltd. (1995)
17. Mesnager, S.: Bent functions from spreads. *Fq11 proceedings* (preprint 2013)

18. Schaathun, H.G.: The Boneh-Shaw fingerprinting scheme is better than we thought. *IEEE Transactions on Information Forensics and Security* 1(2), 248–255 (2006)
19. Sloane, N.: Covering arrays and intersecting codes. *Journal of Combinatorics Designs* 1, 51–63 (1993)
20. Song, Y., Li, Z.: Secret sharing with a class of minimal linear codes. *CoRR* abs/1202.4058 (2012)
21. Tsfasman, M.A., Vladut, S.G.: *Algebraic Geometric Codes*. Kluwer (1991)
22. Zhou, Z., Ding, C.: A class of three-weight cyclic codes. *CoRR* abs/1302.0569 (2013)

A Code-Based Undeniable Signature Scheme

Carlos Aguilar-Melchor, Slim Bettaieb, Philippe Gaborit, and Julien Schrek

XLIM-DMI, Université de Limoges,

123, av. Albert Thomas

87060 Limoges Cedex, France

{carlos.aguilar,slim.bettaieb,philippe.gaborit,julien.schrek}@xlim.fr

Abstract. In this work we propose the first code-based undeniable signature scheme (and more generally the first post-quantum undeniable signature scheme). The verification protocols for our scheme are 3-pass zero-knowledge protocols derived from the Stern authentication protocol. There are two main ideas in our protocol, first we remark that it is possible to obtain a full-time undeniable signature from a one-time undeniable signature simply by signing the one-time public key by a standard signature. Second, we introduce a zero-knowledge variation on the Stern authentication scheme which permits to prove that one or two different syndromes are associated (or not) to the *same* low weight word. We give a polynomial reduction of the security of our scheme to the security of the syndrome decoding problem.

Keywords: code based cryptography, undeniable signature schemes, syndrome decoding.

1 Introduction

Digital signatures are an important cryptographic primitive. Digital signature are known to satisfy the property of universal verifiability; the ability to verify the signature by anyone using the signer's public key. Undeniable signature schemes are similar to digital signatures but they do not satisfy the property of universal verifiability. In fact, verification of validity or invalidity of the signature can only be done by interaction with the signer.

The concept of undeniable signatures scheme was introduced by Chaum and Antwerpen [6]. The motivation of the authors was the limitation of verifiability of signed confidential contract or documents that contains private informations. Since their introduction in 1989, undeniable signatures were used in various applications such as software licensing [4], e-voting [19] as well as e-cash [5,18]. An undeniable signature scheme is composed of four algorithms. The key generation algorithm, the signing algorithm, and two verification protocols (confirmation/disavowal) used to prove the validity or invalidity of a given signature.

An undeniable signature scheme is secure if it satisfies two basic security properties: the existential unforgeability, which means that no one except the signer can generate valid signatures; and invisibility, which means that an adversary cannot be able to decide whether a given signature is valid or not. Zero-knowledge and non-transferability are two additional security properties related

to the verification protocols. It is also important to have protocols which satisfy non-transferability because undeniable signatures with this property have many useful applications like for instance software licensing.

There are many constructions that have been proposed since 1989, works based on discrete logarithm such as [3,4,7,9] or RSA in [13] and pairings based constructions such as [15,16,17]. Concerning code-based cryptography besides classical signature schemes like CFS [8] or zero-knowledge based signature schemes [21,12] there exist very few special signature schemes, essentially the ring signature of [1,2].

Our Contribution. All previous undeniable signature schemes are based on number theory hard problems (like factoring big integers and discrete logarithm) or pairings hard problems. We give the first undeniable signature with security based on problems from codes. With the resistance against quantum computers, code-based cryptography is an attractive alternative to construct cryptographic primitive. Our scheme has the properties of existential unforgeability, invisibility, completeness and soundness. It can also be made non-transferable and zero knowledge. In this conference version of the paper we only present a version of the protocol without the non-transferability property. Meanwhile we explain in the last section how it is possible to obtain this property but by lack of space we do not give the detail of this more complex version of the protocol.

Organization. Section 2 is composed of definitions about difficult problems and some properties. In section 3 we give a general overview of the scheme. In section 4, a description of the protocol is proposed. Section 5 is concerned with security proofs. In section 6, some parameters are given to use the signature with different level of security. At last section 7 deals with non-transferability.

2 Preliminaries

2.1 Notation

We use the following syntax notation: \parallel for the concatenation. We note $\mathcal{M}_{i,j}(\mathbb{F}_q)$ the set of matrix of size $i \times j$ over \mathbb{F}_q .

Let h and h' two hash functions modeled as random oracles. Images of h are in $\mathcal{M}_{n-k,n}(\mathbb{F}_2)$ and images of h' are in $\mathcal{M}_{n,t}(\mathbb{F}_2)$. For protocols, we denote by \mathcal{P} the prover and by \mathcal{V} the verifier. The prover secret key is $K_s^{\mathcal{P}}$ and his public key is $K_p^{\mathcal{P}}$. In this paper we use a generic digital signature Sign with (K_s^S, K_p^S) the couple of private-public key associated. To sign x with Sign we use the notation $\text{Sign}(x, K_s^S)$. To verify the signature σ of m , we use $\text{Verif}(m, \sigma, K_p^S)$. Among different possibilities ([8,10]) an example of a signature algorithm which could be used in practice in our case is a code-based signature obtained through a Fiat-Shamir paradigm applied to a zero-knowledge code-based authentication scheme like the Stern authentication scheme or its derivatives ([21]).

We also use usual coding theory notation, with \mathcal{C} a code of parameter $[n, k]$, H the parity check matrix of \mathcal{C} ($H \in \mathcal{M}_{n-k, n}(\mathbb{F}_2)$). w is integer and denote the Hamming weight of a word.

We denote by S_w^n the set of words in \mathbb{F}_2^n of Hamming weight w and by w_t the function which gives the Hamming weight of a word. The Gilbert-Varshamov bound of a random code $\mathcal{C}[n, k]$ is the value of a distance d such that the number of codewords of weight less than d is close or equal to the number of possible syndromes, it means that *on the average* any syndrome as one preimage, but this does not imply that there is a unique preimage for any syndrome, we will see in the following sufficient conditions to obtain a unique preimage with a good probability.

In the paper, all elements referring to a matrix or a vector is named by a bold character.

2.2 Syndrome Decoding Problem

The syndrome decoding problem is a NP-Complete problem based on coding theory. In our scheme, we use the particular finite field \mathbb{F}_2 . It consists in finding a low weight word of a given syndrome.

Definition 1 (Syndrome Decoding problem).

Let \mathbf{H} a random $(n - k, n)$ matrix in a finite fields \mathbb{F}_q , w an integer and $\mathbf{s} \in \mathbb{F}_q^{n-k}$. Finding \mathbf{e} a word of weight w or less and such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}$.

We will have to use a decisional form of that problem in the proof of invisibility.

Definition 2 (Decisional Syndrome Decoding problem).

Let \mathbf{H} be a $(n - k, n)$ matrix in a finite fields \mathbb{F}_q , w an integer and $\mathbf{s} \in \mathbb{F}_q^{n-k}$. Finding if there exists a word \mathbf{e} of weight w such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}$ with good probability.

In cryptography, the syndrome is usually used as the public key and the low weight word as the private key. This problem is used in cryptography because of its useful properties. Indeed it uses fast operations like matrix product or coordinates permutation. This problem also resists *a priori* to quantum computers at the difference of hard problems based on number theory.

Remark 1. It is a folk theorem that both previous problem can be reduced to one another. In one way it is obvious, for the other way, suppose we have an algorithm which solves the decisional problem, then starting from a syndrome $\mathbf{s} = \mathbf{H}\mathbf{x}^T$ associated to a small weight w unknown word \mathbf{x} (below the Gilbert-Varshamov bound and with unicity conditions- the case we are interested in), then it is possible to recover \mathbf{x} . Indeed, consider the syndrome $\mathbf{s}_i = \mathbf{s} +$ the i^{th} column of \mathbf{H} , if the i^{th} coordinate is in the support of x then the unique small weight word associated to \mathbf{s}_i is $\mathbf{x} + '1'$ in the i^{th} coordinate, therefore knowing by the decisional problem whether there is a solution of weight $w + 1$ or not permits to deduce whether the i^{th} coordinate belongs to the support of \mathbf{x} or not. Eventually it is possible to recover \mathbf{x} completely.

In the following we prove a simple lemma on sufficient conditions on the preimage of a syndrome for a random code that we will need later for the parameters of our scheme.

Lemma 1 (Unicity of syndrome preimage). *Let \mathbf{H} be a random dual matrix of a random $C[n, k]$ binary code, and let x be a codeword of weight w with syndrome $\mathbf{s} = \mathbf{H}\cdot\mathbf{x}^t$. The probability over the random choices of \mathbf{H} and \mathbf{x} that there exists a different codeword $\mathbf{y} \neq \mathbf{x}$ of weight w such that $\mathbf{H}\cdot\mathbf{y}^t = \mathbf{s}$ is bounded above by $\frac{\sum_{i=0}^{2w} \binom{n}{i}}{2^n}$.*

Proof. Suppose there exists y of weight w such that $H\cdot y^t = s = H\cdot x^t$ then $H\cdot(x-y)^t = 0$, and hence $x-y$ belongs to C . Now obviously the weight of $x-y$ is bounded above by $2w$. For any random code C the number of codewords of weight $2w$ is bounded above by $\sum_{i=0}^{2w} \binom{n}{i}$, since x is random of fixed weight there are 2^{n-k} possible syndromes, the result follows.

In practice since $\sum_{i=0}^{2w} \binom{n}{i} \sim \binom{n}{2w}$, taking $\frac{\binom{n}{2w}}{2^n} < 2^{-80}$ assures with very good probability the uniqueness of a preimage of any syndrome s . Asymptotically it means that unicity is assured for a very good probability whenever $2w$ is a little below the Gilbert-Varshamov bound of the random code, and hence w is a little less than half the Gilbert-Varshamov bound.

2.3 Definitions

Definition 3. *Formally, an undeniable signature scheme consists of a quadruplet of polynomial-time algorithms defined as follows.*

- $\text{KG}_{\mathcal{P}}(n, k)$: *The key generation algorithm consists of a polynomial randomized algorithm. The signer runs $\text{KG}_{\mathcal{P}}$ with input the security parameters (n, k) to get valid key pair $(K_s^{\mathcal{P}}, K_p^{\mathcal{P}})$.*
- USign : *The signing algorithm is a randomized algorithm that takes the signer's secret key $K_s^{\mathcal{P}}$ and a message m , and outputs a signature σ of m .*
- $\text{Confirmation}_{\mathcal{P}, \mathcal{V}}$: *The confirmation protocol is an interactive proof between a prover \mathcal{P} and a verifier \mathcal{V} in possess of a potential message-signature pair (m, σ) . The protocol takes as input the message m , its supposed signature σ , the prover's public key $K_p^{\mathcal{P}}$. At the end of the protocol outputs “Success” if σ is a valid signature of m , otherwise it outputs \perp .*
- $\text{Disavowal}_{\mathcal{P}, \mathcal{V}}$: *The disavowal protocol is an interactive proof between a prover \mathcal{P} and a verifier \mathcal{V} in possess of a potential invalid message-signature pair (m, σ) . The protocol takes as input the message m , σ , the prover's public key $K_p^{\mathcal{P}}$. At the end of the protocol outputs “Success” if σ is an invalid signature of m , otherwise it outputs \perp .*

A one-time undeniable signature allows to sign a single message. In order to obtain a secure signature the signer has to generate new key-pairs for each message he wants to sign.

Definition 4. *More formally such scheme consists of a quadruplet of polynomial-time algorithms defined as follows.*

- $\text{KG}_{\mathcal{P}}(n, k)$: *The key generation algorithm consists of a polynomial randomized algorithm. The signer runs $\text{KG}_{\mathcal{P}}$ with input the security parameters (n, k) to get valid key pair $(\mathcal{K}_s, \mathcal{K}_p)$.*
- OTUSign : *The signing algorithm is a randomized algorithm that takes the signer’s secret key \mathcal{K}_s and a message m , and outputs a one-time signature σ of m .*
- $\text{Confirmation}_{\mathcal{P}, \mathcal{V}}$: *The confirmation protocol is similar to the one given in Definition 3.*
- $\text{Disavowal}_{\mathcal{P}, \mathcal{V}}$: *The disavowal protocol is similar to the one given in Definition 3.*

2.4 Security Model

In this subsection we give the definitions of the security notions related to the signing algorithm (Definition 5, Definition 6) and ones related to the verification protocols (Definition 7, Definition 8). The following two definitions are similar to the one in [14].

Definition 5 (Unforgeability). *The unforgeability of an undeniable signature is defined by the following game between a challenger \mathcal{CH} and a forger \mathcal{F} . \mathcal{CH} starts the game by running $\text{KG}_{\mathcal{P}}(n, k)$ and obtain $(K_s^{\mathcal{P}}, K_p^{\mathcal{P}})$, then he gives $K_p^{\mathcal{P}}$ to \mathcal{F} . The forger is allowed to make signing queries to a signing oracle and to the verification oracles. The signing oracle responds with $\sigma \leftarrow \text{USign}(K_s^{\mathcal{P}}, m)$ when he receives m as a signing query from \mathcal{F} . Queries to the confirmation (respectively, disavowal) oracles are of the form (m, σ) . After receiving a query (m, σ) , the confirmation (respectively, disavowal) oracle execute the confirmation (respectively, disavowal) protocol with \mathcal{F} . Finally, \mathcal{F} outputs a pair (m^*, σ^*) . \mathcal{F} wins the game if (m^*, σ^*) is a valid message-signature pair and m^* was not queried to the signing oracle.*

Definition 6 (Invisibility). *Let us consider \mathcal{D} as a probabilistic polynomial time distinguisher. We define the invisibility of an undeniable signature scheme using the following game between a challenger \mathcal{CH}' and a distinguisher \mathcal{D} . First \mathcal{CH}' runs $\text{KG}_{\mathcal{P}}(n, k)$ and obtain $(K_s^{\mathcal{P}}, K_p^{\mathcal{P}})$, then he gives $K_p^{\mathcal{P}}$ to \mathcal{D} . The distinguisher \mathcal{D} allowed to make queries to oracles that execute the signing and the verification (confirmation and disavowal) protocols. At some time, \mathcal{D} will have no more access to the verification oracles. Then, \mathcal{D} requests a challenge on message m of his choice. \mathcal{CH}' chooses $b \xleftarrow{\$} \{0, 1\}$ and sets $\sigma \leftarrow \text{USign}(K_s^{\mathcal{P}}, m)$ if $b = 0$, otherwise σ is chosen randomly at uniform from the signature space and sends σ to \mathcal{D} . After that, \mathcal{D} is allowed to make queries to the signing and verification protocols on condition that m is not a query of signing oracle and (m, σ) is not a query to the verification protocols. At the end \mathcal{D} outputs a guess b' , we say that \mathcal{D} wins the game if it outputs the right guess.*

Definition 7 (Completeness). *If the prover \mathcal{P} and the verifier \mathcal{V} honestly execute the confirmation and the disavowal protocols. Then for any message m , we have that*

$$\Pr[\text{Confirmation}_{\mathcal{P},\mathcal{V}}(m, \text{USign}(K_s^{\mathcal{P}}, m)) \text{ outputs Accept}] = 1.$$

And for any invalid message-signature pair (m, σ) , we have that

$$\Pr[\text{Disavowal}_{\mathcal{P},\mathcal{V}}(m, \sigma) \text{ outputs Accept}] = 1.$$

Definition 8 (Soundness). *For any cheating prover \mathcal{P} and any invalid message-signature pair (m, σ) , we have that :*

$$\Pr[\text{Confirmation}_{\mathcal{P},\mathcal{V}}(m, \sigma) \text{ outputs Accept}] \text{ is negligible.}$$

And for any message-signature pair (m, σ) , we have that :

$$\Pr[\text{Disavowal}_{\mathcal{P},\mathcal{V}}(m, \sigma) \text{ outputs Accept}] \text{ is negligible.}$$

Definition 9 (Impersonation). *Let us consider \mathcal{I} as a probabilistic polynomial time impersonator. We define the impersonation of an undeniable signature scheme using the following game between a him and a verifier \mathcal{V} . First \mathcal{I} gets on input a public key $K_p^{\mathcal{P}}$. Then the impersonator \mathcal{I} can make some signing queries or confirmation/disavowal queries. At the end, the impersonator \mathcal{I} decides to run a confirmation or disavowal verification with a verifier \mathcal{V} and a couple message-signature (m, σ) . We say that \mathcal{I} wins the game if the verifier \mathcal{V} is convinced of the verification protocol. An undeniable signature is said secure against impersonation under adaptive chosen message attack if no PPT impersonator \mathcal{I} has a non-negligible advantage in the above game.*

3 General Overview

In this section we describe the main idea used to obtain our undeniable code-based signature scheme. As recalled in the introduction we only detail here a basic undeniable scheme with unforgeability and invisibility properties, we will consider non-transferability in the last section of the paper.

The main idea is that we consider a one-time undeniable signature scheme, and we show that it is possible to extend it to a full-time undeniable signature scheme simply by the use of a standard full signature algorithm. We first explain the idea behind the one-time signature scheme (with confirmation and disavowal protocols), we then explain how to make a full time signature with the one time protocol.

3.1 Undeniable One-Time Signature: Basic Idea

The idea behind undeniable signature schemes is that the undeniable signature must be linked to the secret key but 'not too much' in the sense that if the link is too strong we get closer to a classical signature scheme.

The idea of the one time signature is that the private key of the one time signature is a low Hamming weight word \mathbf{x} . The public key is its syndrome \mathbf{s} , computed with a fixed random $(n - k) \times n$ matrix \mathbf{H} . The difficulty to recover \mathbf{x} from \mathbf{s} and \mathbf{H} is the syndrome decoding problem described in section 2. The idea of this one time signature is to take as a signature a new syndrome \mathbf{z} associated to \mathbf{x} but with another $k' \times n$ matrix \mathbf{M} related to the message m . We can assume that \mathbf{z} cannot be distinguished from a random word, which is the principal property of an undeniable signature. Now, two problems appear. The first one is that a new syndrome gives new informations about the secret \mathbf{x} . An adapted parametrization can solve this problem, nevertheless such a signature can only be a few-time signature, because any new signature will give new linear equations on the secret \mathbf{x} . The second problem is to involve the message m into the signature. This problem is solved by constructing a new random matrix \mathbf{M} from a hash value of m (also related to the public key), which serves to compute the new syndrome. If one proceeds this way the obtained signature $\mathbf{z} = \mathbf{M}\mathbf{x}^t$ is well linked, both to the secret \mathbf{x} and to the message m since the matrix \mathbf{M} is built from the message m . The following scheme sums up the basic idea of our protocol.

$$\begin{array}{l}
 H \rightarrow \\
 M \rightarrow
 \end{array}
 \left(\begin{array}{cccc}
 H_{1,1} & H_{1,2} & \dots & H_{1,n} \\
 \dots & \dots & \dots & \dots \\
 H_{n-k,1} & H_{n-k,2} & \dots & H_{n-k,n} \\
 \hline
 M_{1,1} & M_{1,2} & \dots & M_{1,n} \\
 \dots & \dots & \dots & \dots \\
 M_{k',1} & M_{k',2} & \dots & M_{k',n}
 \end{array} \right)
 \begin{pmatrix}
 x_1 \\
 x_2 \\
 \vdots \\
 x_{n-1} \\
 x_n
 \end{pmatrix}
 =
 \begin{pmatrix}
 s_1 \\
 \vdots \\
 s_{n-k} \\
 \hline
 z_1 \\
 \vdots \\
 z_{k'}
 \end{pmatrix}
 \begin{array}{l}
 \leftarrow s \\
 \\
 \leftarrow z
 \end{array}$$

Proceeding this way permits to obtain many possible different signatures depending on the message, since different messages give different matrices \mathbf{M} which give different signature \mathbf{z} . In terms of parameters a natural choice is to take $n - k = k' = n/3$, moreover for unicity reasons the weight w of \mathbf{x} is chosen with conditions of Lemma 1 for a random matrix H .

Now the main property of an undeniable signature scheme are the two disavowal and confirmation protocols, we explain now how our approach makes them easy to obtain.

3.2 One-Time Undeniable Signature: Confirmation and Disavowal Protocols

The aim of the confirmation protocol is to make possible for the signer to prove the validity of the signature. In our case, we want to prove the relation between the syndrome \mathbf{s} and the syndrome \mathbf{z} , ie the fact that both syndromes are obtained from the same low weight vector \mathbf{x} . A natural tool to use in that case is the Stern authentication protocol [21], this protocol is a zero-knowledge protocol in which

the prover proves that he knows a low weight vector \mathbf{x} associated to a given syndrome $\mathbf{H}\mathbf{x}^T$.

We introduce a variation on the Stern authentication protocol in which one considers not one syndrome $\mathbf{s} = \mathbf{H}\mathbf{x}^T$ but two syndromes $\mathbf{s} = \mathbf{H}\mathbf{x}^T$ and $\mathbf{z} = \mathbf{M}\mathbf{x}^T$ at the same time. In the confirmation protocol the prover proves that he knows a small weight vector \mathbf{x} associated to both syndromes \mathbf{s} and \mathbf{z} . Now this approach also permits to show that the prover is unable to repudiate a valid signature \mathbf{z} , indeed if the signature $\mathbf{z} = \mathbf{M}\mathbf{x}^T$ is valid, the fact that the prover has to use *at the same time* the same small weight vector with both syndromes \mathbf{s} and \mathbf{z} , implies that he has to use \mathbf{x} (since \mathbf{x} is the unique small weight vector associated to \mathbf{s}). Now since the verifications of both syndromes are linked in the variation of the protocol, the same \mathbf{x} will also work for the valid signature $\mathbf{z} = \mathbf{M}\mathbf{x}^T$, so that the prover will not be able to repudiate the valid signature \mathbf{z} .

Now for the disavowal protocol, the case where a signature \mathbf{z} is not valid (ie $\mathbf{z} \neq \mathbf{M}\mathbf{x}^T$ with a very strong probability), the prover wants to prove that he knows the secret associated to the public key syndrome \mathbf{s} but that he did not compute the false signature \mathbf{z} . Recall that for the variation on the protocol the same small weight vector \mathbf{x} is used with the two syndromes \mathbf{s} and \mathbf{z} *at the same time* so that it permits the prover to prove that he knows the secret \mathbf{x} and that this secret is compatible with the public syndrome $\mathbf{s} = \mathbf{H}\mathbf{x}^T$ but that \mathbf{x} is not compatible with \mathbf{z} , so that since \mathbf{z} is not compatible with \mathbf{x} , the signature \mathbf{z} is not valid since $\mathbf{M}\mathbf{x}^T \neq \mathbf{z}$.

In this way we obtain the desired properties, a prover can confirm a valid signature but cannot repudiate it, moreover he can disavow a false signature without giving any information since the protocol is zero-knowledge.

3.3 One-Time Signature to Full-Time Signature

The idea is that a full-time undeniable signature can be obtained by using a standard signature scheme together with the one-time undeniable signature scheme. Each time the signer wants to sign, he generates *extra keys* corresponding to one-time undeniable keys: a private extra key which permits him to use the undeniable one-time signature and a public extra key, which is signed with the standard signature. The new full-time signature is therefore composed of the public extra key signed with the standard signature, together with a one-time signature of the message obtained from the private extra key.

The invisibility property remains unchanged because the digital signature is not used to sign the message but to sign an extra public key, which can be anything. The unforgeability or other properties are not affected neither by this transformation. On the other hand, a problem appears when the signer has to stock each extra secret key in anticipation for future confirmation or disavowal protocols. We can solve that by adding to the undeniable signature the seed who was used to build the extra secret key. Naturally, this seed will be masked by a random word, which will be part of the full time undeniable private key.

To conclude, we have a private key, composed of a digital signature private key and a random word, and a public key equal to the digital signature public key.

The full time undeniable signature consists in constructing for each signature: an extra couple of one-time key, computing a one-time signature for the message and signing with the digital signature the one-time extra public key and the masked seed together. The only difference for the confirmation and disavowal protocols is the rebuilding of the private extra key for each signature.

4 Undeniable Signature Scheme

The undeniable signature defined in section 2 is composed of 4 algorithms. We describe here those components: a key generator, a signature algorithm, a confirmation protocol and a disavowal protocol.

4.1 Key Generation

In this subsection we present the key generator algorithm. It depends on a random public matrix \mathbf{H} .

To guarantee that the signer cannot hide a trapdoor in the public dual matrix H , we will use the following approach. The signer is given access to a hash function h that outputs random values in $\mathbb{F}_2^{(n-k) \times n}$, therefore to generate his public matrix he chooses a random seed \mathbf{sd} and sets \mathbf{H} to the hash value of the seed. Obviously, to share the matrix with the verifier he will just send the seed used as input to the hash function h .

It only generates a random word and includes the key pair of **Sign** (the generic signature), (K_s^S, K_p^S) . Recall that **Gen** is a key generation algorithm for the generic signature algorithm. In the following for the protocol and the proofs we consider the case $k = \frac{2n}{3}$.

$\text{KG}_{\mathcal{P}}(n, k)$:
Generates a key pair

1. Choose a random seed $\mathbf{sd} \in \mathbb{F}_2^n$.
2. Set $\mathbf{H} = h(\mathbf{sd})$, where h is a hash function such that $h(\mu) \in \mathbb{F}_2^{(n-k) \times n}$.
3. Run **Gen** to get a valid key pair (K_s^S, K_p^S) .
4. Set $\mathbf{r} \xleftarrow{\$} \mathbb{F}_2^t$.
5. Output $(K_s^{\mathcal{P}}, K_p^{\mathcal{P}}) = ((K_s^S, \mathbf{r}), (K_p^S, \mathbf{sd}))$.

Fig. 1. Signer key generation

The digital signature scheme can be for instance the CFS signature scheme or the Stern zero-knowledge signature scheme.

4.2 The Signature

The signer wants to sign the message m . First of all he needs to compute an extra key $(\mathcal{K}_s, \mathcal{K}_p)$ using the EKG (Extra Key Generator) generator and random seeds α and sd .

EKG(α, sd):

Generates a key pair $(\mathcal{K}_s, \mathcal{K}_p)$, for w a given weight chosen as in Lemma 1 for a random code \mathcal{C} , relatively to the security parameter t

1. Set $\mathbf{H} = h(\text{sd})$.
2. Set $\mathbf{e} \xleftarrow{\$} S_w^n$.
3. Set $\mathbf{s} = \mathbf{H}\mathbf{e}^T$.
4. Output $(\mathcal{K}_s, \mathcal{K}_p) = (\mathbf{e}, \mathbf{s})$.

Fig. 2. Extra Key Generation

The extra key pair consists of a word of Hamming weight w and its syndrome. To build the signature, the signer needs $\mathbf{M} = h'(m)$ and then computes :

$$((\mathbf{s}, \alpha \oplus \mathbf{r}, \text{sd}), \text{Sign}((\mathbf{s}, \alpha \oplus \mathbf{r}, \text{sd}), K_s^S), \mathbf{M}\mathbf{e}^T).$$

USign(K_s^P, m):

For a given message m , he generates a signature as follows:

1. Set $\alpha \xleftarrow{\$} \mathbb{F}_2^t$
2. The signer runs EKG(α, sd) and obtain a key pair $(\mathcal{K}_s, \mathcal{K}_p) = (\mathbf{e}, \mathbf{s})$.
3. Set $\mathbf{M} = h'(m)$ where h' is a hash function such that $h'(m) \in \mathbb{F}_2^{n \times t}$.
4. Set $\mathbf{z} = \mathbf{M}\mathbf{e}^T$.
5. Set $x = (\mathbf{s}, \alpha \oplus \mathbf{r}, \text{sd})$.
6. Set $y = \text{Sign}(x, K_s^S)$.
7. Output $\sigma = (x, y, \mathbf{z})$.

Fig. 3. Signing protocol

The signature space and the invisibility property

For the invisibility property, we have to introduce the signature space. Even if it was obvious for the one time signature, it is not the case for the full time signature. The signature space associated to the key pair (K_s^P, K_p^P) is :

$$(x, y, \mathbf{z}) \text{ where } : x \in (\mathbb{F}_2^n)^3, y = \text{Sign}(x, K_s^S) \text{ and } \mathbf{z} \in \mathbb{F}_2^{n-k}$$

It is clear that a signature obtained with the signature algorithm in Figure 3 belongs to this space. On the other hand, for the aim of the undeniable signature, with the parameters in the parameter section , this space is much larger than the possible result of Figure 3.

Remark 2. This signature space is composed with valid and invalid signatures. All the signature are composed with a valid message-signature couple, for the standard signature Sign . That could be disturbing but there is no contradiction with the definition. In fact, anyone can build an invalid signature, from a valid or invalid one, just by changing the third value \mathbf{z} . The validity of the standard signature only means that his owner participated to a given signature process, but that does say what he did: it could be a previous signature, for which the last part z of the triple $\sigma = (x, y, z)$ could be invalid or valid.

4.3 The Verification

According to the definition, the undeniable signature needs two verification protocols. A confirmation and a disavowal protocol. Both involve a prover \mathcal{P} and a verifier \mathcal{V} . Those protocols are interactive and aim to prove the validity or the invalidity of the signature. We describe the confirmation and the disavowal protocol in the same scheme because there are only few differences between those protocols. We denote by 'verification' those two protocols, the difference being in the verification step $c = 1$, which changes for the disavowal and the confirmation protocols .

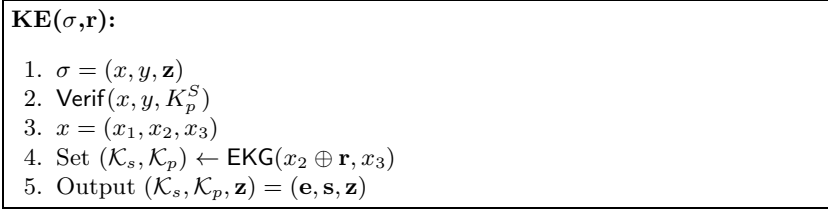


Fig. 4. Key Extractor algorithm

To make the verification protocol the signer needs to use the extra key used during the signing protocol. To do that he uses his secret key \mathbf{r} to find α and rebuild the extra key $(\mathcal{K}_s, \mathcal{K}_p)$.

The verification protocol is a variation of the Stern protocol , in [21], with the extra key $(\mathcal{K}_s, \mathcal{K}_p)$. Here the protocol uses four commitments, the last one is to verify the validity or invalidity of the undeniable signature.

The protocol of verification needs to be repeated several time to decrease the probability close to 0.

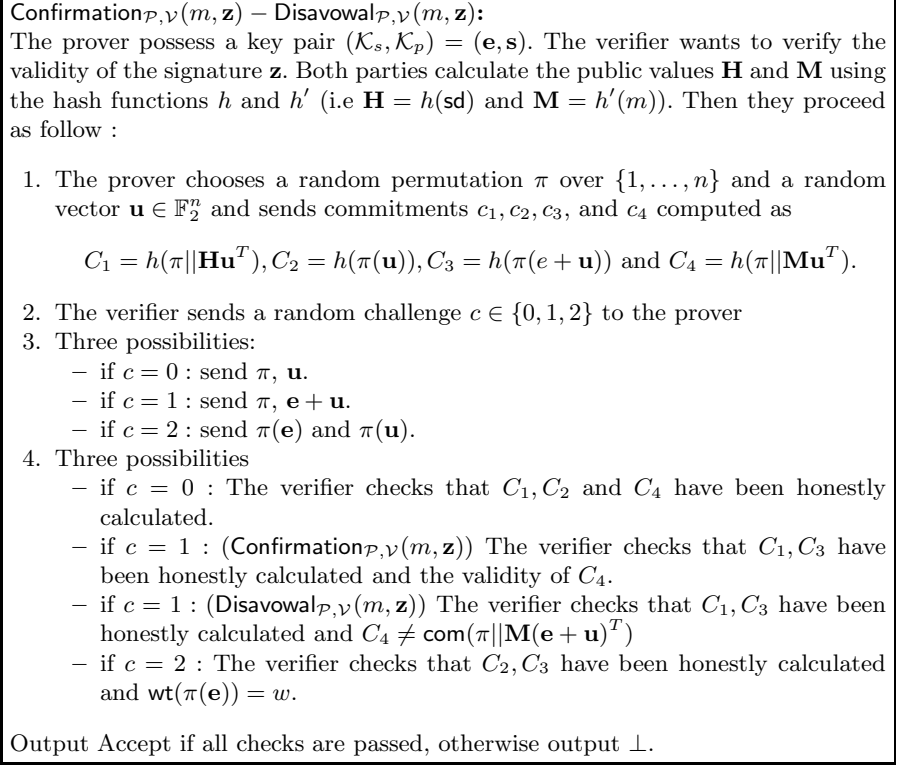


Fig. 5. Verification protocol

5 Security

5.1 Completeness

Theoreme 1. *If the prover \mathcal{P} and the verifier \mathcal{V} honestly execute the confirmation and the disavowal protocols. Then for any message m , we have that*

$$Pr[\text{Confirmation}_{\mathcal{P},\mathcal{V}}(m, \text{USign}(K_s^{\mathcal{P}}, m)) \text{ outputs Accept}] = 1.$$

And for any invalid message-signature pair (m, σ) , we have that

$$Pr[\text{Disavowal}_{\mathcal{P},\mathcal{V}}(m, \sigma) \text{ outputs Accept}] = 1.$$

Proof. For the first part, let's take (m, σ) a valid couple, message-signature. We have that $\sigma = (\mathbf{x}, \mathbf{y}, \mathbf{z})$, and we can extract the extra key (\mathbf{e}, \mathbf{s}) such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}$ and $\mathbf{M}\mathbf{e}^T = \mathbf{z}$ where $h'(m) = \mathbf{M}$. With (\mathbf{e}, \mathbf{s}) , the prover can make the confirmation protocol. The last part we have to verify is that the verification of the hash value in the protocol is possible. This verification is trivial for the major part of the case and the other part use the properties : $\mathbf{H}(\mathbf{e} + \mathbf{u})^T - \mathbf{s} = \mathbf{H}\mathbf{u}^T$,

$\mathbf{M}(\mathbf{e} + \mathbf{u})^T - \mathbf{z} = \mathbf{M}\mathbf{u}^T$ or $\pi(\mathbf{e}) + \pi(\mathbf{u}) = \pi(\mathbf{e} + \mathbf{u})$. The confirmation protocol is always accepted in that condition.

For the second part of the theorem, let's take (m, σ) an invalid signature. We saw that even the invalid signatures $\sigma = (\mathbf{x}, \mathbf{y}, \mathbf{z})$ are made with a valid couple $(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \text{Sign}(\mathbf{x}, K_s^S))$. That means that the key extractor can be apply to this signature to obtain the extra key (\mathbf{e}, \mathbf{s}) . If σ is invalid, that means that $\mathbf{z} \neq \mathbf{M}\mathbf{e}^T$, with $\mathbf{M} = h'(m)$. That imply that $\mathbf{M}(\mathbf{e} + \mathbf{u})^T - \mathbf{z} \neq \mathbf{M}\mathbf{u}^T$ and the hash value C_4 cannot be verify. Meanwhile, the others hash values are verified as well as in the first part of the proof. The disavowal protocol is so always accepted in that condition.

5.2 Soundness

Theorem 2. *For any cheating prover \mathcal{P} and any invalid message-signature pair (m, σ) , we have that :*

$$\Pr[\text{Confirmation}_{\mathcal{P}, \mathcal{V}}(m, \sigma) \text{ outputs Accept}] \text{ is negligible.}$$

And for any message-signature pair (m, σ) , we have that :

$$\Pr[\text{Disavowal}_{\mathcal{P}, \mathcal{V}}(m, \sigma) \text{ outputs Accept}] \text{ is negligible.}$$

Proof. Let's prove the first point of the theorem with (m, σ) an invalid couple message-signature. The idea of the proof is as follows: we will prove that the cheating prover \mathcal{P} has a maximum probability of $2/3$ for each round to cheat (meaning proving that an invalid couple message-0 is valid), the result is obtained by proving that if he can cheat (being able to answer correctly to all possible challenges) then necessarily its couple message-signature is valid, which is a contradiction, since by hypothesis it is invalid (the proof for the Disavowal works the same but in the other way).

The signature σ is equal to $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. As we saw, even if σ is an invalid signature, the couple (\mathbf{x}, \mathbf{y}) stay valid for the signature Sign . We can make an extraction of the key (\mathbf{e}, \mathbf{s}) where $\mathbf{H}\mathbf{e}^T = \mathbf{s}$ and $w(\mathbf{e}) = w$. The invalidity of σ and the validity of (\mathbf{x}, \mathbf{y}) imply that $\mathbf{z} \neq \mathbf{M}\mathbf{e}^T$, with $\mathbf{M} = h'(m)$.

To accept the confirmation protocol, the verifier needs to verify all the hash values at the end of the interaction. We consider the situation where the prover can answer correctly to the 3 possible challenges. More precisely, suppose that for $c = 0$ the prover has sent (say) a permutation ϕ (with $\phi(x)$ the action of the permutation ϕ on a vector x) and α , the permutation (say) ψ and the word β for $c = 1$ or the words (say) $(\gamma$ and δ for $c = 3$. Those values verify the construction of the hash values, it implies that :

$$C_1 = h(\phi \parallel \mathbf{H}\alpha^T) = h(\psi \parallel \mathbf{H}\beta^T - \mathbf{s})$$

$$C_2 = h(\phi(\alpha)) = h(\delta)$$

$$C_3 = h(\psi(\beta)) = h(\gamma + \delta)$$

$$C_4 = h(\phi || \mathbf{M}\alpha^T) = h(\psi || \mathbf{M}\beta^T - \mathbf{z})$$

$$wt(\gamma) = w$$

Either the prover can make a collision on h or we have the equalities :

$$\phi = \psi$$

$$\mathbf{H}\alpha^T = \mathbf{H}\beta^T - \mathbf{s}$$

$$\phi(\alpha) = \delta$$

$$\psi(\beta) = \gamma + \delta$$

$$\mathbf{M}\alpha^T = \mathbf{M}\beta^T - \mathbf{z}$$

Those equations imply that $\mathbf{s} = \mathbf{H}\phi^{-1}(\gamma)^T$ and $\mathbf{z} = \mathbf{M}\phi^{-1}(\gamma)^T$ and $wt(\gamma) = w$. From Lemma 1 we have that $\phi^{-1}(\gamma) = \mathbf{e}$. The next equation gives $\mathbf{z} = \mathbf{M}\mathbf{e}^T$ and let appear a contradiction since because of the invalidity $\mathbf{z} \neq \mathbf{M}\mathbf{e}^T$. That means that the prover is not able to answer correctly the 3 different challenges in one round. The best he could do is answer with a probability $2/3$. With a t -round protocol, it makes the probability for the confirmation to outcome *accept* equal to $(2/3)^t$, which becomes negligible when t increases.

The proof of the second point is close to the first one. Let us consider (m, σ) as a valid couple message-signature. We can use the extractor key to obtain (\mathbf{e}, \mathbf{s}) such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}$, $wt(\mathbf{e}) = w$ and $\mathbf{M}\mathbf{e}^T = \mathbf{z}$ where $\sigma = (\mathbf{x}, \mathbf{y}, \mathbf{z})$. The last equation is due to the validity of σ . For the confirmation protocol, the fact that the prover could anticipate the 3 different challenges implied that $\mathbf{z} = \mathbf{M}\mathbf{e}^T$. For the disavowal protocol, it is easy to remark that the equation become $\mathbf{z} \neq \mathbf{M}\mathbf{e}^T$. It is also a contradiction in this case. That means that the prover cannot anticipate the 3 challenges during the protocol and that the probability that the algorithm outputs *accept* is $(2/3)^t$, which is negligible in t .

5.3 Zero-Knowledge

(Sketch of proof) The zero knowledge proof consists in constructing a simulator which can create an interaction between a prover and a verifier in polynomial time. The interaction must be indistinguishable from an interaction with a true prover and a true verifier. The interaction for the verification part is only about the verification protocol, because this is the only part during the verification where the prover and the verifier interact. We can construct a simulator for this protocol using the one constructed in the Stern authentication protocol's proof of zero knowledge, with an 0 hash value. The rest of the interaction is exactly the same. We refer to the proof of zero knowledge of the Stern protocol to prove the property, in [21].

5.4 Impersonation

Theorem 3. *The signature is secure against impersonation under adaptive chosen message attack or we can find a polynomial attack on the syndrome decoding problem.*

Proof. We consider an instance of the syndrome decoding problem (H, s) with H a $(n - k) \times n$ matrix and s its syndrome in \mathbb{F}_2 . Let consider a PPT \mathcal{I} that wins the game of impersonation with non-negligible probability and input H . We note (m, σ) the couple message-signature used to convince the verifier \mathcal{V} and (e, He^T) the couple of one-time secret and public keys associated to the one-time signature in σ . Since it is possible to choose the signing queries, we can assume that $He^T = s$. The one-time keys in the signatures are independent from the full-time keys. That means that the impersonator \mathcal{I} took no information about (e, He^T) from the signing queries. Moreover, the confirmation/disavowal protocols are zero-knowledge. That means that \mathcal{I} took no information about (e, He^T) from those protocols as well. If we remove the commitment C_4 and its verification in the confirmation/disavowal protocols, we obtain the same protocol, call *new verification* which doesn't refer to the message m . The impersonator \mathcal{I} is able to pass this protocol as well as the previous one. To summarize, we can create from \mathcal{I} a PPT \mathcal{KE} with input (H, s) and no access to signing queries or verification queries which can pass the *new verification* protocol with non-negligible probability. This *new verification* corresponds exactly to the protocol of Stern [21] and it is proved on the proof of soundness of this paper that, with the previous assumptions, \mathcal{I} and \mathcal{KE} know e with non-negligible probability (we used the same arguments in the soundness proof in our paper). We proved that if \mathcal{I} can wins this game with non-negligible probability then \mathcal{KE} can solve the syndrome decoding instance (H, s) with good probability.

5.5 Unforgeability

In this section we consider parameters for the syndrome decoding associated to a code $\mathcal{C} [n, k = \frac{2n}{3}]$ with a small weight words of weight w , chosen as in Lemma 1. The first part of the section proves the security of the one-time signature and the second part proves the unforgeability of the full-time signature.

Theorem 4. *Consider the previous parameters for one-time signature associated to a $[n, k = \frac{2n}{3}]$ code \mathcal{C} with security parameters associated to a weight w chosen for unicity of syndrome preimage as in Lemma 1 for a random $[n, k = \frac{2n}{3}]$ code. Then if a forger can forge a one-time signature in O operations with probability λ , then he can solve the syndrome decoding problem for a random $[n, \frac{n}{3}]$ code with small weight w in $n^3 + O$ operations with probability $\frac{1}{2^{q_{RO}}} \lambda$ in the random oracle model, where q_{RO} correspond to the number of queries to the random oracle.*

Proof. We consider an instance $\mathbf{H}\mathbf{x}^T = y$ of the syndrome decoding problem such that \mathbf{H} is a $2/3n \times n$ matrix and \mathbf{x} is a small weight word of weight w , as

in Lemma 1 for a random $[n, k = \frac{2n}{3}]$ code. We will prove that a forger \mathcal{F} who can forge a one-time signature (with previously described parameters) can solve this instance of the syndrome decoding problem with a polynomial factor in some probability. We decompose the instance (\mathbf{H}, \mathbf{y}) into $(\mathbf{H}_1, \mathbf{y}_1)$ and $(\mathbf{H}_2, \mathbf{y}_2)$ with $\mathbf{H} = \begin{pmatrix} H_1 \\ H_2 \end{pmatrix}$ and $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$. The problem is now to search \mathbf{x} such that $\mathbf{H}_1 \mathbf{x}^T = \mathbf{y}_1$ and $\mathbf{H}_2 \mathbf{x}^T = \mathbf{y}_2$ with \mathbf{H}_1 and \mathbf{H}_2 of size $\frac{n}{3} \times n$. In those problems \mathbf{x} still needs to be of weight w . The public key of the one time signature correspond to $(\mathbf{H}_1, \mathbf{y}_1)$. The forger receives the public key $(\mathbf{H}_1, \mathbf{y}_1)$. Then the forger needs to interact with a confirmation/disavowal oracle and a signing oracle to produce a result. He also have access to a random oracle \mathcal{RO} .

When \mathcal{F} asks for a signature, he is given (m, \mathbf{y}_2) (for m a random message). In that case since the signature is one time, the forger has only access to a maximum of one signature. The forger could have asked for the hash value of message m before. That is why, among the $q_{\mathcal{RO}}$ queries to the random oracle, one is fixed to \mathbf{H}_2 . The probability that the hash value \mathbf{H}_2 corresponds to the message m is $\frac{1}{q_{\mathcal{RO}}}$, in the case that this value doesn't correspond, the algorithm stops.

It is also possible for \mathcal{F} to ask for an interaction with the confirmation/disavowal protocol. There is a difficulty here since only someone who knows the secret key can simulate those protocol 0. To overcome this problem we modify the forger \mathcal{F} : when the forger runs the confirmation protocol, he receives at the end a result value equal to true or false. We construct two new PPT Turing machine \mathcal{F}_1 and \mathcal{F}_2 such that both machines respond true when \mathcal{F} asks for the couple (m, \mathbf{y}_2) , but such that for all the other queries \mathcal{F}_1 responds always true and \mathcal{F}_2 responds always false. Notice that in our scheme, the confirmation and the disavowal protocols are almost the same. If the confirmation succeeds that means that the disavowal fails and vice versa. That is not true in general but the fact that the construction of those two confirmation/disavowal protocols in our scheme is almost identical implies this property. That means that if the confirmation protocol succeeds with a negligible probability, the disavowal protocol succeeds with a non negligible probability. Therefore, either \mathcal{F}_1 or \mathcal{F}_2 returns the same result as \mathcal{F} with good probability. We now rename \mathcal{F} the PPT which is \mathcal{F}_1 with probability 1/2 and \mathcal{F}_2 else. This new forger is a polynomial Turing machine which succeeds with non negligible probability and does not ask anything to the confirmation/disavowal protocols.

With those considerations, the machine \mathcal{F} returns a valid couple message-signature (m, σ) . If we denote by \mathbf{H}_3 the hash value of m , we have a new instance of the syndrome decoding problem. This instance is $(\mathbf{H}_1|\mathbf{H}_2|\mathbf{H}_3, \mathbf{y}_1|\mathbf{y}_2|\sigma)$, this instance has a non negligible probability to be solved in polynomial time because the matrix $(\mathbf{H}_1|\mathbf{H}_2|\mathbf{H}_3)$ has a non negligible probability to be invertible. The solution of this problem is obviously the solution to the first difficult problem (\mathbf{H}, \mathbf{y}) , the cost of the matrix inversion is n^3 . That concludes the proof.

Theoreme 5. *Consider a full-time signature built from a one-time signature with parameters chosen as in the previous theorem, and from a generic signature Sign. Then if a forger can forge a full-time signature in O operations and N*

access to the signing oracle with probability λ , either he can forge the signature **Sign** or he can solve an instance of the syndrome decoding problem (as in the previous theorem: for a $[n, n/3]$ code with w the value of the GV bound for a random $[n, 2n/3]$ code) in $N(n^3 + O)$ operations with probability $\frac{1}{2q_{\mathcal{RO}}} \lambda$ in the random oracle model, where $q_{\mathcal{RO}}$ correspond to the number of queries to the random oracle.

Proof. This proof is very similar to the one for the one-time signature. Here we use only one instance of the syndrome decoding problem but we consider N instances of this problem but with different matrices \mathbf{H}_i . These instances correspond to different couples $(\mathbf{H}_1, \mathbf{y}_1), \dots, (\mathbf{H}_N, \mathbf{y}_N)$. Each matrix \mathbf{H}_i is split in two - equal size - matrices $\mathbf{H}_{i,1}$ and $\mathbf{H}_{i,2}$, as well as each \mathbf{y}_i is split into $\mathbf{y}_{i,1}$ and $\mathbf{y}_{i,2}$.

We will prove that a forger \mathcal{F} which can forge a full-time signature in polynomial time and with non negligible success probability, can solve one instance of the syndrome decoding problem in polynomial time and with non negligible probability.

We start the proof as in the previous theorem. We compute a key pair (sk, pk) and give pk to the forger. The forger needs access to a signing oracle, a confirmation oracle, a disavowal oracle and a random oracle. This time, the forger can ask for a polynomial number of signatures.

The signing oracle is simulated with the signing protocol described in figure 3 and using as extra key an instance $\mathbf{H}_{i,1}, \mathbf{y}_{i,1}$ of the difficult problem. The forger could have asked for the hash value of message needed before. That is why, among the $q_{\mathcal{RO}}$ queries to the random oracle, one is fixed to $\mathbf{H}_{j,2}$.

The problem with the confirmation and the disavowal protocol is solved with the same approach as in the proof for the one-time signature.

The forger can return a valid couple message-signature (m, σ) in polynomial time and with non negligible probability. The signature σ is a triple $(x, \text{Sign}(x, K_s^S), z)$. Hence, either the forger can forge the signature **Sign** or he uses a signed couple (x, K_s^S) which was already used. If he uses a signed couple (x, K_s^S) , which has already been used, it means that the forger is in the case of forging a one-time signature. Then we can reduce the proof to the proof of the one time signature. The difference being that the probability that the hash value $\mathbf{H}_{i,2}$ corresponds to the message m is $\frac{1}{q_{\mathcal{RO}}}$ in the case.

Therefore, we constructed a PPT machine which can solve one instance among N of the syndrome decoding problem. The fact that those instances are independent means that the maximum gain we can obtain in comparison to the research of only one instance is almost N (see also [20]). With N polynomial in n , this problem is polynomially equivalent to solve one instance of the syndrome decoding problem with parameters values as the one-time signature proof.

5.6 Invisibility

The proof follows the same approach as in the unforgeability proof to prove that the distinguisher \mathcal{D} can solve the decisional syndrome decoding problem.

The problem is given by a matrix \mathbf{H} and a syndrome \mathbf{y} . We want to know if there exists a word of weight w with associated syndrome \mathbf{y} by the matrix \mathbf{H} . We split \mathbf{H} into $\begin{pmatrix} H_1 \\ H_2 \end{pmatrix}$ and \mathbf{y} into $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$. We make the same construction as the previous theorems in order to interact with the distinguisher \mathcal{D} , taking $(\mathbf{H}_2, \mathbf{y}_2)$ to compute the new signature. The distinguisher returns true or false, corresponding to the fact that (\mathbf{H}, \mathbf{y}) has a solution or not.

6 Parameters

In this section we give some parameters to use with the signature for different levels of security. For simplicity matter we consider the case where the matrix H is a $\frac{n}{3} \times n$ matrix (hence $k = \frac{2n}{3}$) and where length of the signature is also $k' = n/3 = n - k$. The parameter w - the low weight of x - must be chosen according to Lemma 1 with appropriate probability so that one can suppose the unicity of the syndrome preimage - in practice a little below half the Gilbert-Varshamov bound of a random $[n, \frac{2n}{3}]$ code-, and the length n of the code must be chosen so that it resists to syndrome decoding attacks for finding a low weight vector of weight w in a $[n, \frac{n}{3}]$ code. For the best practical attacks we follow the lower bounds from [11]. Notice that the matrices \mathbf{H} and \mathbf{M} can also be chosen quasi-cyclic, which reduces the size of the public key. Remember that from the reduction theorems the security of the scheme for a $[n, 2n/3]$ code with a given w corresponds to attacking with the same value of w , not a $[n, 2n/3]$ code but a $[n, n/3]$ code.

n	k	w	bits security
4500	3000	130	80
5100	3400	157	90
6000	4000	175	100
7500	5000	220	128

For digital signature one can use the zero-knowledge based Stern signature scheme (and its improvements) so that the digital signature has length roughly 40k Bytes.

Remark 3. In the approach of the proofs of the different theorem, we 3 for simplicity to rely on Lemma 1 and unicity of the syndrome preimage. Another approach would have consisted in basing our proofs on the fact that a cheating prover, given a syndrome s with preimage x of weight w is not able to find another word y of weight w with the same syndrome s . This can be done by supposing that finding a word of weight $2w$ in a random code $[n, n/3]$ is difficult. Indeed if such a word of weight $2w$ was reachable, it would possible by cutting this word in two equal weight parts, to find two words of same weight with the same syndrome. Such an approach would indeed improve a little bit the parameters but would make proofs less direct.

7 Non-transferability

The non-transferability property corresponds to the fact that only one chosen person at a time can verify the validity of the undeniable signature, and that this person is not able to convince a third party of the validity or invalidity of a signature. To make this possible, any potential verifier must have a private and a public key.

Definition 10 (Non-transferability).

Let (m, σ) be a message-signature couple. If someone gets information of the validity or invalidity of the signature with good probability, then he gets information of almost one of the private key used during a verification. The private key can be the one of the prover or the one of one of the verifiers.

Remark 4. The non-transferability is not induced by the zero-knowledge property. In fact, the zero-knowledge proof only guarantees that the process doesn't give any information about the secret key. For the non-transferability, we want that the verifier gives no information about the proof of knowledge of the secret. In the particular case of the undeniable signature describes in the paper, the verifier can generate his challenges from a PRNG, make an approval (or disavowal) protocol, and reveals the transcript of the interaction with the seed of the PRNG to prove, with no interaction, the validity (or invalidity) of the signature. This strategy correspond to the Fiat-Shamir paradigm and is usually used to make a signature scheme from a zero-knowledge authentication scheme.

The scheme described in this paper does not respect this property. For a reason of lack of space, we did not present the whole version of the scheme which permit to obtain a non-transferable undeniable signature. We now explain here, how it is possible construct a new version of our protocol with this property. First of all, the verifier needs a couple of private-public keys. We use the extra key generator to generate the verifier keys. Recall that a ring signature scheme for n persons permits to convince a verifier that one persons of the ring has signed the message but without knowing which one. A particularity of a ring signature scheme compared to group signature scheme is that : first the anonymity cannot be eliminated and second that the public keys used to compose the ring are not related to one another, so that it is possible to use public keys of persons which do not know their public is used (we refer to [1,2] for more details). The idea to obtain the non-transferability property is then to use a ring authentication protocol with 2 persons: the prover and the chosen verifier. If one proceeds this way, the chosen verifier knows that the prover is confirming a valid signature - simply by the fact that the chosen verifier knows there are only two persons in the ring: himself and the prover, and since it is not him confirming the signature, it has to be the prover. Now the verifier cannot convince a third person that the prover did the confirmation, without revealing his private key, indeed since a third person is not be able to break the anonymity of the 2-ring authentication scheme, a third person (at the difference of any two member of the ring) is not

be able to know who did the confirmation protocol between the two members of the ring: the prover or the chosen verifier.

Overall the protocol we described in previous sections can be adapted to obtain the non-transferability property by mixing it with the code-based ring signature of [2], but we obtain a more complex version of our protocol that we do not describe in this extended abstract version.

8 Conclusion

In this paper we present the first undeniable signature based on coding theory (and more generally the first undeniable signature scheme for post-quantum cryptography). Our scheme relies on a variation of the Stern authentication algorithm. The main idea of the scheme is to consider a very simple one-time undeniable signature which is turned into a full-time signature scheme. The scheme uses only fast operations with attractive parameters, as small size keys. We described how the main properties of undeniable signatures could be obtained. We also give a general view on how it was also possible to obtain the non-transferability properties from our scheme.

References

1. Aguilar Melchor, C., Cayrel, P.-L., Gaborit, P.: A new efficient threshold ring signature scheme based on coding theory. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 1–16. Springer, Heidelberg (2008)
2. Aguilar-Melchor, C., Cayrel, P.-L., Gaborit, P., Laguillaumie, F.: A new efficient threshold ring signature scheme based on coding theory. *IEEE Transactions on Information Theory* 57(7), 4833–4842 (2011)
3. Boyar, J., Chaum, D., Damgård, I., Pedersen, T.: Convertible undeniable signatures. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 189–205. Springer, Heidelberg (1991)
4. Chaum, D.: Zero-knowledge undeniable signatures. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 458–464. Springer, Heidelberg (1991)
5. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
6. Chaum, D., van Antwerpen, H.: Undeniable signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
7. Chaum, D., van Heijst, E., Pfitzmann, B.: Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 470–484. Springer, Heidelberg (1992)
8. Courtois, N.T., Finiasz, M., Sendrier, N.: How to achieve a mcEliece-based digital signature scheme. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 157–174. Springer, Heidelberg (2001)
9. Damgård, I., Pedersen, T.: New convertible undeniable signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 372–386. Springer, Heidelberg (1996)
10. Finiasz, M.: Parallel-CFS strengthening the CFS mceliece-based signature scheme. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 159–170. Springer, Heidelberg (2011)

11. Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 88–105. Springer, Heidelberg (2009)
12. Gaborit, P., Schrek, J.: Efficient code-based one-time signature from automorphism groups with syndrome compatibility. In: International Symposium on Information Theory (ISIT 2012), pp. 1982–1986. MIT, Boston (2012)
13. Gennaro, R., Krawczyk, H., Rabin, T.: RSA-based undeniable signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 132–149. Springer, Heidelberg (1997)
14. Kurosawa, K., Heng, S.-H.: 3-move undeniable signature scheme. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 181–197. Springer, Heidelberg (2005)
15. Laguillaumie, F., Vergnaud, D.: Short undeniable signatures without random oracles: The missing link. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 283–296. Springer, Heidelberg (2005)
16. Laguillaumie, F., Vergnaud, D.: Time-selective convertible undeniable signatures. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 154–171. Springer, Heidelberg (2005)
17. Libert, B., Quisquater, J.-J.: Identity based undeniable signatures. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 112–125. Springer, Heidelberg (2004)
18. Pointcheval, D.: Self-scrambling anonymizers. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 259–275. Springer, Heidelberg (2001)
19. Sakurai, K., Miyazaki, S.: An anonymous electronic bidding protocol based on a new convertible group signature scheme. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 385–399. Springer, Heidelberg (2000)
20. Sendrier, N.: Decoding one out of many. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 51–67. Springer, Heidelberg (2011)
21. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (1994)

Filtered Nonlinear Cryptanalysis of Reduced-Round Serpent, and the Wrong-Key Randomization Hypothesis*

James McLaughlin¹ and John A. Clark²

¹ Solarflare, Block 2, Westbrook Centre,
Milton Road, Cambridge, UK
james.d.mclaughlin@gmail.com

² Department of Computer Science, University of York,
Deramore Lane, Heslington, York, UK
john.clark@york.ac.uk

Abstract. We present a deterministic algorithm to find nonlinear S-box approximations, and a new nonlinear cryptanalytic technique; the “filtered” nonlinear attack, which achieves the lowest data complexity of any known-plaintext attack on reduced-round Serpent so far. We demonstrate that the Wrong-Key Randomization Hypothesis is not entirely valid for attacks on reduced-round Serpent which rely on linear cryptanalysis or a variant thereof, and survey the effects of this on existing attacks (including existing nonlinear attacks) on 11 and 12-round Serpent.

Keywords: Nonlinear cryptanalysis, generalized linear cryptanalysis, multidimensional linear cryptanalysis, WKRH, Wrong-Key Randomization Hypothesis, Serpent.

1 Introduction

Linear cryptanalysis [1][2] has had several extensions and variations proposed since its discovery in 1993. One such generalisation was the use of *nonlinear* approximations. That is, instead of being restricted to equations of the form $x_{a_1} \oplus x_{a_2} \oplus \dots \oplus x_{a_i} \oplus y_{b_1} \oplus y_{b_2} \oplus \dots \oplus y_{b_j}$ in the input bits x_i and output bits y_i of cipher components, the cryptanalyst could utilise higher-degree terms such as $x_{a_1}x_{a_3}$.

This was first proposed by Harpes, Kramer and Massey [3], and investigated in more depth by Knudsen and Robshaw [4]. It was concluded that nonlinear approximations could replace linear approximations only in the first and last rounds of the distinguisher - and even then, there were problems that would not apply in the case of a purely linear approximation. One of these was the difficulty of finding the nonlinear S-box approximations; for a DES-sized 6×4

* This work was carried out while the first author was a graduate student at the University of York.

S-box, the search space for possible approximations was 2^{64} in size, increasing to 2^{256} for an AES-sized 8×8 S-box.

Courtois [5][6] demonstrated that the use of nonlinear approximations was in fact possible in other rounds of a Feistel cipher, as long as each round's approximation was a particular form of quadratic expression. This approach, however, could not be generalised to non-Feistel ciphers.

The first attempt to obtain arbitrary-degree nonlinear approximations without restrictions on cipher type was the use of simulated annealing (SA) by Clark et al. [7] to evolve nonlinear approximations to the MARS S-box [8] for use in the first round of nonlinear distinguishers. They were able to obtain nonlinear approximations holding with a much higher absolute bias (151/512) than the best linear approximations for the MARS S-box. However, no attack on reduced-round MARS that could exploit these was known. Subsequent research [9] built on this, refining the SA algorithm to obtain nonlinear approximations for the Serpent S-boxes.

In this paper, we build on the above research in the following directions:

- We present a fast, deterministic algorithm for obtaining the full set of nonlinear approximations for a given S-box with the highest possible bias.
- The cryptanalyst does not know the values of the key bits xored with the bits involved in the nonlinear approximation. Where n_0 denotes the nonlinear function involved, computing n_0 on the bits exposed through partial encryption/decryption means that the cryptanalyst is in fact computing $n_{\alpha_1\alpha_2\dots\alpha_h} = n_0(x_1 \oplus k_{\alpha_1}, x_2 \oplus k_{\alpha_2}, \dots, x_h \oplus k_{\alpha_h})$. There exist 2^h candidates for the correct function, n_i , to compute on these bits, and the cryptanalyst does not know which is correct. We present an adaptation of Matsui's Algorithm 2 which can make use of nonlinear approximations, as well as two separate means of addressing this issue. One of these is a straightforward generalisation of linear cryptanalysis to incorporate the new approximations, the other utilises a technique known as "filtering" to achieve even lower data complexity at the cost of higher time and memory complexity.
- In [9], nonlinear approximations for some of the Serpent S-boxes, with higher bias than the best linear approximations for the same, were derived. We incorporate these into both filtered and unfiltered nonlinear attacks, which we compare to the previous attacks on 11-round Serpent.
- For linear cryptanalysis and its variants, the "Wrong-Key Randomization Hypothesis" (WKRH) states that, for any wrong key value used to partially encrypt/decrypt a cipher during cryptanalysis, the expectation for the bias is 0; and it should certainly be much lower than the bias for the correct key. We demonstrate that in the case of the Serpent cipher, this does not always apply, and quantify its effects on the various attacks on 11- and 12-round Serpent.

This paper is structured as follows: The remainder of this section describes the notation used, and provides a brief description of certain key aspects of linear cryptanalysis. Section 2 describes the new search algorithm for S-box approximations, and discusses the ways in which the new approximations affect the

attack. It also contains an explanation of how we handle nonlinear approximations differently in the filtered attacks. Section 3 describes the new attacks, in particular the adaptation of Collard et al.’s improved algorithm for the analysis phase [10] to the nonlinear and filtered nonlinear domains. It also contains a detailed discussion of the complexities of this algorithm and the nonlinear attacks. Finally, Section 4 surveys the existing attacks on reduced-round Serpent, recalculates their complexities in light of the issues surrounding the WKRH, and describes the nonlinear and filtered nonlinear attacks on 11-round Serpent.

1.1 Linear Cryptanalysis – The Algorithm 2 Attack

We use the following notation:

- N is the number of known plaintext/ciphertext pairs.
- K denotes the cipher’s key length.
- P and C denote, respectively, the plaintext and ciphertext.
- l is the number of “active” text bits which are relevant to the attack. In a 1R attack, this includes plaintext bits which are xored together but not partially encrypted.
- The subset of key bits we seek to recover is known as the *target partial subkey* (TPS).
- k is the number of key bits in the TPS. For 2R linear attacks on SPN-based ciphers such as Serpent, $k = l$. For 1R attacks on SPNs, k is equal to the number of active ciphertext bits.
- k_0 is the correct k -bit value for the TPS.
- In nonlinear attacks, k_1 denotes the subset of TPS bits that are used in the round keys for the outer rounds of the cipher. (All attacked key bits are of this type in a linear attack.)
- In nonlinear attacks, k_2 is the set of TPS bits active in the outer rounds of the *approximation*.
- r is the number of rounds of the cipher.
- P_s is the success probability of the attack.
- If, in a cryptanalytic attack, we aim for the correct key to be one of the 2^{n-a} highest ranked keys, the value a is referred to as the “advantage”.

In a 1R attack, the cryptanalyst knows of a linear approximation to rounds $1, 2, \dots, (r-1)$ of the cipher, and uses candidate key values to partially decipher some of the bits in the known ciphertexts. In a 2R attack, the cryptanalyst only has an approximation to rounds $2, \dots, (r-1)$, and as well as the aforementioned partial decryption, must partially encrypt certain plaintext bits to obtain the bits on which the probabilistic linear relation is expected to hold.

The theoretical bias for this linear approximation is calculated using the *Piling-Up Lemma* [1]:

Definition 1. For $1 \leq i \leq n$, let X_i be independent Bernoulli random variables such that $p_i = Pr(X_i = 0)$, and $(1 - p_i) = Pr(X_i = 1)$.

(In the case of linear cryptanalysis, $X_i = 0$ iff the linear approximation to the i th approximated S-box holds.)

Then $Pr(X_1 \oplus X_2 \oplus \dots \oplus X_n = 0)$ is:

$$(1/2) + 2^{n-1} \prod_{i=1}^n (p_i - 1/2).$$

with probability bias:

$$\epsilon = 2^{n-1} \prod_{i=1}^n (p_i - 1/2)$$

In reality, the probabilities of the linear approximations to the S-boxes in one round holding are not independent of the probabilities of the linear approximations to other rounds holding, so the Piling-Up Lemma only estimates the true bias. This is usually accurate enough for the purposes of cryptanalysis, although there are situations where it is not [11][12][13].

Definition 2. Where a linear approximation holds with bias ϵ , i.e. with probability $1/2 + \epsilon$, the capacity C of the approximation is equal to $4 \times \epsilon^2$. More generally, in an attack using multiple approximations A_i ($1 \leq i \leq M$), each with bias ϵ_i , the set of approximations has capacity $4 \sum_{i=1}^M \epsilon_i^2$.

2 Finding and Utilising Nonlinear Approximations

2.1 Finding the Approximations

The approximations used are of the following form: (linear function of either the input or the output bits) = (nonlinear function of some subset of the other) with bias ϵ . The linear function is defined by some bitmask with 1s in the positions corresponding to the bits involved.

We use the term “projection” to refer to the subset of either the input bits x_i or output bits y_i involved in the nonlinear function. For example, $y_0 \oplus y_1 \oplus y_0y_3$ has the projection $\{y_0, y_1, y_3\}$.

Let us use Serpent S3 to illustrate the new algorithm. We will search for approximations involving a nonlinear function on the set of input bits, with projection $\{x_0, x_1, x_3\}$ and bitmask 10 (1010).

First of all, we reorder the truth table of the linear function.

Value $x_0x_1x_3$ of bits in projection	000	001	010	011	100	101	110	111
TT entry for $x_0x_10x_3$	0	0	1	1	1	0	0	1
TT entry for $x_0x_11x_3$	0	1	1	1	1	0	0	0

We define a template for the approximations as follows: For any value $x_0x_1x_3$ of the bits in the projection, if the truth table of the linear function takes the

value 0 more often than the value 1, let template entry $x_0x_1x_3$ equal 0. If the opposite is true, set entry $x_0x_1x_3$ to 1. If the two entries occur equally often, let entry $x_0x_1x_3$ be the character *.

This gives us:

Value $x_0x_1x_3$ of bits in projection	000	001	010	011	100	101	110	111
Template entry for $x_0x_1x_3$	0	*	1	1	1	0	0	*

We can now obtain four approximations with bias 6 by replacing the *s in the template with 0s and 1s. These are:

- 00111000 ($x_1 \oplus x_0 \oplus x_0x_3 \oplus x_0x_1x_3$),
- 00111001 ($x_1 \oplus x_0 \oplus x_0x_3$),
- 01111000 ($x_3 \oplus x_1 \oplus x_0 \oplus x_1x_3$), and
- 01111001 ($x_3 \oplus x_1 \oplus x_0 \oplus x_1x_3 \oplus x_0x_1x_3$).

2.2 The “Related” Approximations

We have already mentioned the difficulty faced by the cryptanalyst in working out which of $2^{|k_2|}$ functions is the correct nonlinear function. One possible approach would be to compute all of the functions, and for each guess at the key bits involved, accept the function with the highest bias as correct.

If we wish to include the k_2 bits in our attack, several of the related approximations may also possess biases with high magnitude. In some cases, one or more of the relateds may have a bias with the same magnitude as the original, and even when this is not the case, we may still need to distinguish, say, the correct function and a bias 24 approximation from an incorrect function defining a bias -22 approximation.

Let x_i denote the i th input bit to whichever S-box we are dealing with, and y_j the j th output bit. Consider the nonlinear approximation to Serpent S3 in Table 1:

Table 1. Nonlinear approximation to Serpent S3. The “filtered” biases are explained in subsection 2.3.

Related approximation	Nonlinear function	Bias	Bias (filtered)
0	$x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_1y_3$	+6	6
1	$x_3 \oplus x_4 = y_4 \oplus y_3 \oplus (y_1 \oplus 1)y_3$	0	-4
2	$x_3 \oplus x_4 = y_4 \oplus (y_3 \oplus 1) \oplus y_1(y_3 \oplus 1)$	0	0
3	$x_3 \oplus x_4 = y_4 \oplus (y_3 \oplus 1) \oplus (y_1 \oplus 1)(y_3 \oplus 1)$	+2	0
4	$x_3 \oplus x_4 = (y_4 \oplus 1) \oplus y_3 \oplus y_1y_3$	-6	0
5	$x_3 \oplus x_4 = (y_4 \oplus 1) \oplus y_3 \oplus (y_1 \oplus 1)y_3$	0	0
6	$x_3 \oplus x_4 = (y_4 \oplus 1) \oplus (y_3 \oplus 1) \oplus y_1(y_3 \oplus 1)$	0	2
7	$x_3 \oplus x_4 = (y_4 \oplus 1) \oplus (y_3 \oplus 1) \oplus (y_1 \oplus 1)(y_3 \oplus 1)$	-2	-4

For this nonlinear approximation, if $y_1y_3 = 1$, any wrong guess for corresponding key bits (k_a, k_c) will result in y_1y_3 being wrongly calculated as 0. If $y_1y_3 = 0$, by contrast, only one of the three possible wrong guesses for (k_a, k_c) will result in its being incorrectly calculated. In general, an incorrect key guess will not consistently result in the wrong value being assigned to the nonlinear terms affected by it, and so will not simply leave the magnitude of the bias invariant.

It is therefore necessary to guess the key bits involved in the first and last rounds of the approximation, as well as those involved in the first and last rounds of the cipher, to obtain the latter set of key bits.

2.3 Increasing the Signal/Noise Ratio – “Filtering” Nonlinear Cryptanalysis

Each of the approximations in Section 2.1 has bias $6/16 = 0.375$. However, for two of the possible inputs to the function (001 and 111), the approximation has no bias. In the analysis phase, no information is obtained by adding to counter values when these inputs occur, and we therefore have no reason to do so.

In fact, we have very good reason not to do so. Consider that, for each (k_1, k_2) pair, by ignoring (P, C) -pairs such that function inputs 001 or 111 would occur, we effectively increase the bias from $6/16$ to $6/12$. Since the data complexity is proportional to the square of the bias, we appear to reduce the KP requirements to $(3/4)^2 = 9/16$ of their original value. In actual fact, since the improved bias is obtained by discarding a quarter of the available data, the value of N is only reduced to $3/4$ of its original value.

This improvement may come at a price. In a basic nonlinear attack using one of the four original approximations, we could for two of these approximations ignore certain values of k_2 which simply resulted in the truth table of the nonlinear approximations being flipped. Since we now need the full value of $PARTIAL_ENCRYPT(P \oplus k_1) \oplus k_2$ to know whether to filter it out, and since different k_2 result in different sets of values being filtered out, we cannot now easily omit these k_2 from the attack. As this previously allowed us to compute the nonlinear function for only half the values of k_2 , the time complexity of the attack is doubled.

For example, in a basic nonlinear attack using the approximation from Table 1, we would not have needed to compute truth table values for half of the relateds, since the related for $k_2 \oplus 100$ would have the same absolute bias (but opposite sign) to that for k_2 . In an attack using filtering, this is clearly no longer the case.

2.4 How Unbalanced Nonlinear Components in the Approximation Affect the Attack

Let us assume that one end of the overall approximation is balanced. Without loss of generality, we may assume that this is the input end. Let $P(\text{function at output end} = 0)$ be denoted α .

Then, for an incorrect key, $\Pr(\text{approximation} = 0) =$

$$\begin{aligned} & Pr((x_{a_1} \oplus \dots \oplus x_{a_s} = 0) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 0)) \\ & + Pr((x_{a_1} \oplus \dots \oplus x_{a_s} = 1) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 1)) \\ & = (0.5 \times \alpha) + (0.5 \times (1 - \alpha)) \\ & = 0.5 \end{aligned}$$

We see that, as long as either the first or the last round of the overall approximation is a balanced function, it does not matter whether the function at the other end is balanced.

Unfortunately, in general we cannot use unbalanced approximations at both ends. Let β denote the probability that the nonlinear function at the input end equates to zero, and let γ be the corresponding probability for the function at the output end. Then, for an incorrect key, $\Pr(\text{approximation} = 0) =$

$$\begin{aligned} & Pr((x_{a_1} \oplus \dots \oplus x_{a_s} = 0) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 0)) \\ & + Pr((x_{a_1} \oplus \dots \oplus x_{a_s} = 1) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 1)) \\ & = (\beta \times \gamma) + ((1 - \beta) \times (1 - \gamma)) \end{aligned}$$

which is not always equal to 0.5.

3 The New Cryptanalysis Algorithm

3.1 Adapting the New Analysis Phase to Nonlinear Cryptanalysis of SPNs

Where the cipher being attacked is an SPN, we present an adaptation of Colard et al.'s algorithm for the analysis phase [10][14] to nonlinear (and filtered nonlinear) cryptanalysis.

- Let $f(i, j)$, where i is the value of the active text bits, and j the value of the k_1 bits with which they are xored, be a $2^{|k_2|}$ -long string of values. We compute it as follows:
 1. Partially encrypt/decrypt i using j . This yields a string δ of text bits entering/leaving the outer rounds of the approximation, $|k_2|$ of which are involved in the nonlinear component.
 2. For each possible value μ of k_2 , compute the nonlinear function on $(\delta \oplus \mu)$.
 3. If the attack does not use filtering, set the μ th entry in the string of values to -1 if the nonlinear approximation does not hold, 1 if it does hold.
 4. For a filtered attack, set the μ th entry to 0 if $(\delta \oplus \mu)$ is one of the inputs being "filtered out". Otherwise, assign either -1 or 1 as a value in the same way as before.

- Since δ is obtained by applying a sequence of functions to a set of bits determined entirely by the value of $(i \oplus j)$, the matrix C such that $C_{ij} = f(i \oplus j) =$ the value $\in \{-1, 1\}$ or $\{-1, 0, 1\}$ which we have just computed can be defined as it was in [10], except that C_{ij} is now a string of values instead of just one. We only need to compute one column of C .
- Where x is the vector containing the frequency with which each value for the l active text bits occurred, since C is a circulant matrix, Cx can be calculated using the Fast Fourier Transform. Each entry in Cx is a $2^{|k_2|}$ -string of integers.
- The memory complexity, and time complexities in terms of arithmetic operations (AOs) and memory accesses (MAs), of the corresponding stages of the linear version of this method can be multiplied by $2^{|k_2|}$ to obtain the complexity of the new method up to this point. Since we do not need this many copies of the “interim” arrays y and z , the memory complexity is in fact slightly lower.

However, if we do not employ key ranking, we can optimise much further in terms of memory. Instead of calculating and storing the entire $2^{|k_1|} \times 2^{|k_2|}$ matrix Cx (the final column of which can use the space originally occupied by x), we could compute one column at a time and search it for its maximal absolute entry. In an array indexed by k_2 value, we store this entry and its corresponding value of k_1 . The highest value will correspond to the most likely (k_1, k_2) and we therefore need only enough memory for two columns of Cx (half of which will in fact be used to store x). For a 2R attack, this reduces memory requirements to $2^{|k_1|+|k_2|} + 2^{|k_1|+4+1} + 2^{\max(l_1, l_2)+5} = 2^{|k_1|+5} + 2^{|k_1|+|k_2|} + 2^{\max(l_1, l_2)+5}$ bytes (The $2^{\max(l_1, l_2)+5}$ bytes are explained in the discussion of the original method’s complexity below) instead of $2^{|k_1|+|k_2|+4.087} + 2^{\max(l_1, l_2)+5}$.

The time complexity of each partial encryption/decryption may be higher than in the case of linear cryptanalysis, due to the complexity of computing the nonlinear function.

- We assign to each Cx_i a score equal to the maximum absolute value therein. The highest-scoring Cx_i corresponds to the most likely key. This requires $(2^{|k_1|} + 2^{|k_1|+|k_2|})$ MAs, to access all values in all strings and write the scores to an array.

The array of scores should need at most (block size of cipher) bits per entry. For block size 128, this adds $16 \times 2^{|k_1|}$ bytes to the memory complexity.

- This allows us to deduce k_1 . We can then proceed to obtain information on k_2 by analysing the biases of the relateds for the correct k_1 candidate.

We can use the maximum absolute bias of all the related approximations to calculate the data complexity in the same way that the bias of one approximation is used in linear cryptanalysis.

3.2 The Complexity of the Method

We discuss the complexity of this analysis method for linear attacks in more detail, having already explained how the complexity of the nonlinear version is related to it.

The column of C has 2^k entries, all -1 or 1. We need 2^k bytes to store it in signed char variables. Variable types using fewer bits are unlikely to be efficiently implemented on any platform.

The vector x has 2^k entries, each of which must be at least $\log_2(N)$ bits in size. On a 64-bit processor, a cipher with 128-bit block size will require 2^{k+1} words here, or 2^{k+4} bytes.

During the calculation of Cx , two “interim” arrays, y and z , are used [10]. Based on Carlet’s description [15] of a version of the FFT which is equivalent to both the Fast Walsh-Hadamard Transform and the k -dimensional FFT of size 2^k [16], we note that the same data type can be used for these as for x , and hence these arrays will require 2^{k+5} bytes.

This gives us a memory complexity of $2^k + 2^{k+4} + 2^{k+5} \approx 2^{k+5.615}$ bytes. The space used by one of the previous arrays, such as x , can be reused to store Cx .

We now consider the time complexity. The algorithm requires 2^k partial encryption/decryptions (PEDs) to calculate a column of C , followed by $O(3 \cdot k \cdot 2^k)$ MAs and AOs to calculate Cx .

Based on the aforementioned version of the FFT [15], we estimate $\approx (2k+3) \cdot 2^k$ MAs per transform. Where y and z denote the output arrays from the first two transforms, calculating the dot product $y \cdot z$ requires 3×2^k MAs. Multiplying the per-transform complexity by three, and adding the complexity of the dot product and the 2^k MAs when the first column of C was calculated and written to memory, gives us $\approx (6k + 13) \cdot 2^k$ MAs in total. As for AOs, the calculation of the dot product requires 2^k AOs, and we estimate $\approx (2k + 1) \cdot 2^k$ AOs per transform, giving us a total of $\approx (6k + 4) \cdot 2^k$.

This is a significant improvement on the $O(2^{2k})$ MAs of the original analysis phase; although in most cases that phase was able to access contiguously stored array elements in sequence and it may be that the extent of the improvement is reduced if this factor aided the CPU’s cache management/location-seeking in main memory.

Equating complexity in terms of memory accesses to complexity in terms of partial cipher encryptions is a difficult matter [17], depending on several factors such as; whether the CPU’s memory controller is on-die or off-die, whether the memory access is to L1 cache, L2 cache, higher-level cache or main memory, the instruction set of the CPU, the efficiency of physical address extension... Previous work on the cryptanalysis of reduced-round Serpent [18][19][20] was not always consistent in converting between the two, and assumed 3 processor cycles per memory access - which would seem to require all memory accesses to be to L1 processor cache. Estimates for the time required to access data in main memory in the event of a cache miss vary from 75 to 300 cycles, and it is not clear if this figure is likely to increase or decrease over time, as processor performance improvements increasingly rely on multiple cores and parallel execution rather

than increased clock speed. In 2003, the NESSIE project [21] gave a figure of 50 cycles per encrypted byte on either the PowerPC G3 or G4 processor as the best performance for full Serpent; if we extrapolate from this to 800 cycles per block we have a worst-case estimate of 1 MA = 3/8 of a full Serpent encryption, and we do not have up-to-date figures for more recent processors to compare this to. It is becoming accepted that there is no easy means to compare complexity in terms of memory accesses to complexity in terms of cipher operations [17], and this is a problem we ourselves will encounter when discussing the performance of our attacks in a later section.

For 2R attacks, later research [14] allows us to trade very slight increases in MA and AO complexity for reduced memory and PED complexities. Let l_1, l_2 be such that $(l_1 + l_2) = k$, where l_1 denotes the number of TPS bits acting on the plaintext, and l_2 the number of TPS bits acting on the ciphertext. Then instead of 2^k PEDs, the method need only execute 2^{l_1} partial encryptions (PEs) and 2^{l_2} partial decryptions (PDs), in addition to est. $(2^{l_2} \cdot (6l_1 + 4) \cdot 2^{l_1} + 2^{l_1} \cdot (6l_2 + 4) \cdot 2^{l_2}) = (6k + 8) \cdot 2^k$ AOs and est. $(2^{l_2} \cdot (6l_1 + 13) \cdot 2^{l_1} + 2^{l_1} \cdot (6l_2 + 13) \cdot 2^{l_2}) = (6k + 26) \cdot 2^k$ MAs. Memory complexity is also improved, since the arrays y and z need only have $2^{\max(l_1, l_2)}$ entries each, reducing the total to $2^k + 2^{k+4} + 2^{\max(l_1, l_2)+5} \approx 2^{k+4.087} + 2^{\max(l_1, l_2)+5}$ bytes.

This algorithm was also generalised for multidimensional linear attacks [14]. Where m is the number of dimensions, the generalised algorithm requires $2^m \times$ the number of MAs and AOs for the one-dimensional case, plus the complexity of computing $2^{l_1+l_2}$ more transforms on a data set of size 2^m to convert correlations to empirical probability distributions, plus the complexity of applying the convolution method [22] to these distributions.

3.3 Other Issues Affecting the Complexity of the New Attack

The time complexity is affected by the cost of computing a nonlinear function compared to the cost of a linear function (usually considered negligible), and by the differing numbers of active S-boxes. For example, this is the nonlinear component of an approximation to DES S5:

$$1 \oplus x_5 \oplus x_5x_6 \oplus x_2x_6 \oplus x_1x_5 \oplus x_1x_2 \oplus x_1x_5x_6 \oplus x_1x_2x_6$$

It is not clear how to compare the complexity of this to the complexity of the full S-box, as it is unlikely that an S-box implementation would rely solely on XOR, AND and NOT (to add the constant term) gates. Moreover, the difficulty of finding, for a given basis and function, the circuit for that function with the smallest number of gates is a difficult and still open problem [23]. It is to be assumed that the cryptanalyst would be using S-box implementations chosen to maximise speed, without regard to such factors as resistance to side-channel attacks which most cipher implementations would have to address.

Since this may be represented by a lookup table with as many elements as the S-box:

1101110111011101100010001000100011111111111111111000000000000000,

and since its algebraic normal form has a much smaller weight than any coordinate function of the S-box, we will assume that the complexity of calculating this function is \leq that of computing the full S-box. Since it must be calculated $2^{|k_2|}$ times for each PED, where S_c denotes the total number of S-boxes in all the rounds of the cipher, we estimate the time required for each PED to be \leq (number of active outer round boxes)/ $S_c + 2^{|k_2|}/S_c$ of the time required for a full encryption.

In a filtered attack, prior to computing the nonlinear function we must check whether $(\delta \oplus \mu)$ is filtered. This requires either another lookup table or the computation of a second function, and so we upper-bound the PED complexity with (no. of active outer round boxes)/ $S_c + 2^{|k_2|+1}/S_c$.

4 Cryptanalysing Reduced-Round Serpent

4.1 Survey of Existing Attacks

The various linear, differential-linear and multidimensional linear attacks on reduced-round Serpent fall into two categories; those based on Collard et al.'s approximations [24][25][10][14] and those based on the approximation of Dunkelman, Keller et al. [18][19][26]. In Appendix A, we point out a few errors in the existing descriptions of Dunkelman et al.'s approximation.

However, the data complexities of some of these attacks have been underestimated.

Let C denote capacity, and p the probability that the linear approximation holds (so $(p - 1/2)$ is the bias). Let P_s denote the success probability of the attack. In [14], N is equal to $4C^{-1}$. This figure is intended to match the values for N used by Collard et al. in multiple linear attacks. However, Collard et al. also used $N = 4 \cdot |p - 1/2|^{-2}$ in conventional linear attacks (apparently to obtain $P_s = 0.785$ as predicted by Matsui in Table 3 of [1]), and this is $16C^{-1}$, not $4C^{-1}$. Moreover, Table 3 of [1] assumes that $l = 6$ - which is not the case in any of the attacks on Serpent - and the values therein are calculated using a double integral which does not match that obtained by Selçuk [27].

The below equation is Selçuk's double integral. It allows the success probabilities for various x such that $N = x \cdot |p - 1/2|^{-2}$ to be calculated for arbitrary k , assuming that the Wrong-Key Randomization Hypothesis holds:

$$P_s = \int_{-2\sqrt{N}|p-1/2|}^{\infty} \left(\int_{-u-2\sqrt{N}|p-1/2|}^{u+2\sqrt{N}|p-1/2|} \phi(v)dv \right)^{2^k-1} \phi(u)du \quad (4.1)$$

However, in the case of Collard et al.'s approximation, the WKRH does not always apply, and so we cannot use Equation 4.1 directly. If we look at Figure 1, we see that an input difference of 0010 or 1000 to Serpent S2 will always cause the value of y_4 to flip - and input difference 1010 will always leave it invariant. Likewise, input differences 0010, 0100 and 0110 will cause the value of $y_0 \oplus y_1 \oplus y_2$ to flip with probability bias $\pm 1/2$.

Absolute input and XOR-truncated output difference distribution table for Serpent S2.

Table entries should be divided by 16.

XORed BITS OF OUTPUT DIFFERENCE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I 0	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8
N 1	-8	0	+4	0	+4	0	0	0	+4	+4	-4	0	0	-4	0	0
P 2	-8	+8	0	0	0	0	0	0	0	0	0	0	0	0	+8	-8
U 3	-8	0	0	+4	0	+4	-4	+4	0	0	0	+4	-4	0	0	0
T 4	-8	0	+4	+4	0	0	+4	-4	+4	-4	0	0	+4	+4	-8	0
5	-8	0	-4	0	0	+4	0	0	-4	+4	0	+4	0	+4	0	0
D 6	-8	0	0	0	0	0	0	0	0	0	0	0	0	0	+8	0
I 7	-8	0	0	-4	+4	0	+4	+4	0	0	-4	0	+4	0	0	0
F 8	-8	+8	0	0	+4	-4	0	0	0	0	+4	-4	0	0	0	0
F 9	-8	0	0	-4	-4	0	0	0	+4	+4	+4	0	+4	0	0	0
E 10	-8	-8	+4	+4	0	0	0	0	0	0	0	0	+4	+4	0	0
R 11	-8	0	-4	0	0	+4	-4	+4	0	0	0	+4	0	+4	0	0
E 12	-8	0	0	0	0	0	+4	-4	+4	-4	0	0	0	0	0	+8
N 13	-8	0	0	+4	0	+4	0	0	-4	+4	0	+4	-4	0	0	0
C 14	-8	0	0	0	+4	-4	0	0	0	0	+4	-4	0	0	0	+8
E 15	-8	0	+4	0	-4	0	+4	+4	0	0	+4	0	0	-4	0	0

Fig. 1. Table showing probability biases for truncated differentials for Serpent S2, in which input difference ΔX causes linear combination λY of the output bits to flip with bias ϵ

Figure 2 shows that four of the input-end S-boxes for which we guess key bits in the Collard/Standart/Quisquater attack are affected by this; two with output bitmask 1 and two with bitmask 14. This means that we can only recover eight of the sixteen key bits for these S-boxes.

Likewise, an input difference to S4's inverse (i.e. an output difference to S4) of 0010, 1100 or 1110 causes the value of $x_0 \oplus x_2 \oplus x_3$ to flip with bias 0.5. Since one of the active ciphertext S-boxes contributes the parity of these bits to the approximation, the number of bits that can be recovered is reduced by 2 again. Instead of recovering 108 key bits, we can only recover 98.

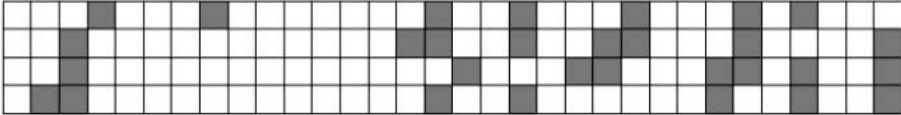


Fig. 2. Input end S-boxes in Collard et al.'s attack. Dark cells signify active output bits

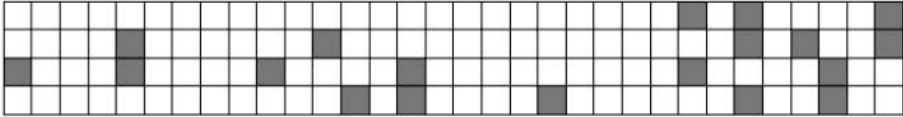


Fig. 3. Output end S-boxes in Collard et al.’s attack

This leaves 22 active S-boxes. For each of these, there are six incorrect keys such that the approximation is expected to hold with an absolute bias equal to half the absolute bias of the correct key. Let ε denote the value $|p - 1/2|$. Instead of using Equation 4.1, we computed P_s in a way that took this into account, by integrating $Z(u) =$

$$\left(\prod_{i=1}^{22} \left(\int_{-u-(2\sqrt{N}\varepsilon(1+1/2^i))}^{u+2\sqrt{N}\varepsilon(1-1/2^i)} \phi(v)dv \right)^{6^i \binom{22}{i}} \right) \left(\int_{-u-2\sqrt{N}\varepsilon}^{u+2\sqrt{N}\varepsilon} \phi(v)dv \right)^{2^l - \sum_{i=1}^{22} 6^i \binom{22}{i}} \phi(u)$$

from $-2\sqrt{N}\varepsilon$ to ∞ :

$$\int_{-2\sqrt{N}\varepsilon}^{\infty} Z(u)du$$

for $k = 98$. However, the difference between this and the value of P_s obtained by using Equation 4.1 was negligible. We deduced that $N = 37.63|p - 1/2|^{-2} \approx 2^{121.234}$ was necessary to achieve $P_s = 0.785$.

In Biham et al.’s linear attack [18], five active plaintext boxes have incorrect key values which cause the parity of their active output bits to flip with bias 0.5. This reduces the number of key bits which can be recovered from 140 to 130. The other active plaintext/ciphertext S-boxes all have six input/output differences which flip the parities of their active bits with bias 0.25, but these have a negligible effect on the value of P_s .

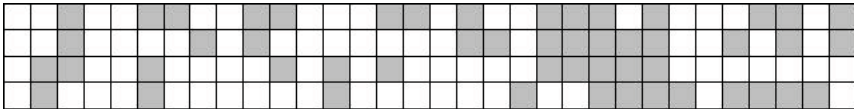


Fig. 4. Active plaintext S-boxes in Biham et al.’s attack

Nguyen et al.’s multidimensional “Method 2” attack on 12-round Serpent [14] modifies Collard et al.’s 9-round approximation by adding a 56-dimensional approximation to the preceding round, resulting in a multidimensional 10-round approximation. The attack aims for maximum advantage $a = k = 172$ with $M = (2^{56} - 1)$.

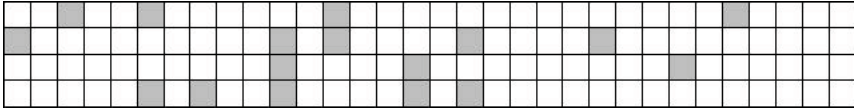


Fig. 5. Active ciphertext S-boxes in Biham et al.’s attack

There are two statistical frameworks for multidimensional linear cryptanalysis; one based on the χ^2 statistic and the other on the log-likelihood ratio (LLR) [28]. None of the attacks in [14] are feasible with the χ^2 statistic, so we assume that the LLR is used and generalise Equation 4.1 to the multidimensional case:

$$\begin{aligned}
 P_s &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^x f_W(y) dy \right)^{2^k - 1} f_R(x) dx \\
 &= \int_{-\infty}^{\infty} F_W(x)^{2^k - 1} \frac{1}{\sigma_R} \phi \left(\frac{x - \mu_R}{\sigma_R} \right) dx \\
 &= \int_{-\infty}^{\infty} \left(\Phi_{\mu_W, \sigma_W^2}(x)^{M+1} \right)^{2^k - 1} \frac{1}{\sigma_R} \phi \left(\frac{x - \mu_R}{\sigma_R} \right) dx
 \end{aligned}$$

For large M and k , this, and its generalisations when the WKRH does not hold, are not easy to calculate numerically. With $M = 2^{56} - 1$, Wolfram Mathematica fails to complete the calculation. As a result, we are currently forced to rely on the approximate statistical framework for the case in which key-ranking is used [28], based on Normally-approximated order statistics [29][27][28]. We begin by addressing an error in this framework.

Let b denote the value $\Phi^{-1}(\sqrt{M+1} - 2^{-a})$. The following equation is derived in [28] using the incorrect approximation $a \approx (b^2/2) - \log_2(M + 1)$:

$$a \approx (\sqrt{NC} - \Phi^{-1}(P_s))^2/2 - \log_2(M + 1) \tag{4.2}$$

Using the approximation $b \approx \Phi^{-1}(1 - 2^{-a - \log_2(M+1)})$ instead, we obtain a very different equation:

$$a \approx 0.72(\sqrt{NC} - \Phi^{-1}(P_s))^2 + \log_2(\sqrt{NC} - \Phi^{-1}(P_s)) + 1.325 - \log_2(M + 1) \tag{4.3}$$

There is also the “linear hull” effect to consider. Approximations with the same input and output bitmasks, but following different paths through the cipher, may cause the actual distribution to differ from that predicted theoretically. Figure 4 of [13] shows the results of experiments on a cipher similar to Serpent [30]. In these, as the number of rounds increases, the magnitude of the bias calculated with the Piling-Up Lemma increasingly underestimates that of the actual bias, to an extent which varies significantly depending on the key value. The LLR statistic in multidimensional linear cryptanalysis rewards high Kullback-Leibler distance from the uniform distribution, and low distance from the theoretical distribution, equally [31]. Clearly, the linear hull effect interferes with the second part of this.

The capacity claimed by Nguyen et al. for this attack is 2^{-116} . However, this is incorrect:

- The various 2^{-4m} terms in their Equation 2 correspond to ± 2 s in the columns of Serpent S2's linear approximation table, and should therefore be 2^{-3m} .
- The equation multiplies ($4 \times$ the square of the bias of the rest of the approximation) by (the sum of some individual S-box biases). These S-box biases should be multiplied by 2 when calculating the overall bias using the Piling-Up Lemma, and Equation 2 should have multiplied by the sum of the squares of these doubled biases.
- The term 8^m assumes that all ± 2 s in the relevant LAT columns for the relevant active S-boxes contribute towards the attack's capacity. However, any approximation which is the sum of an even number of the 56 base approximations will have output bitmask 0, and hence zero bias. After writing a script to quantify the effect of this, we discovered that at least four nonzero entries in the LAT columns for output bitmasks other than 0001, 1110 and 1111 must fail to contribute to the attack's capacity. The highest value this term can take is therefore 4^m .

We therefore recalculate the capacity as follows:

$$C \leq (2^{-58})^2 \sum_{m=0}^{11} \binom{11}{m} 4^m 2^{11-m} 4^4 [(2^{15} (2^{-3m} 2^{-2(11-m)} 2^{-2 \times 4}))^2] \\ = 2^{-120.565}$$

Solving Equation 4.3 with this capacity, we obtain $N \approx 2^{128.956}$, in excess of the size of the codebook and thus invalidating the 12-round Method 2 attack.

A wrong key is far more likely for this sort of attack to have a randomising effect, since each active S-box may contribute more than one bit or sum of bits to the 56 “base” approximations in the multidimensional attack, and a wrong key value for one S-box is less likely to flip all of these with high or indeed any bias than just one. We believe that the WKRH is sufficiently valid for the active plaintext boxes to make little or no difference to P_s in the multidimensional attacks of [14].

The “Method 1” attack from the same paper consists of 2^{128} separate 1R attacks with key guessing on 48 bits in the final round (only 46 of which we can recover). The data complexity for one such 1R attack must lower-bound the value of N . For capacity $2^{-118.565}$, we obtain $N \geq \approx 2^{125.813}$, but note that this may be adversely affected by the linear hull effect.

We also consider [14]'s attacks on 11-round Serpent. In the case of the attack with twelve active S-boxes in the final round, only 46 of the 48 attacked bits can be recovered. We solve Equation 4.3 for capacity $2^{-118.565}$ and $P_s = 0.785$, and obtain $N \approx 2^{125.813}$. In the case of the attack with eleven active final-round S-boxes, we obtain $N \approx 2^{127.784}$.

(These figures do not take into account the linear hull effect, as there is no way to quantify it.)

If the LLR statistic is used, the convolution method [22] for converting empirical probability distributions into scores for key candidates requires $2^k((6m + 13) \cdot 2^m)$ MAs + $2^k((6m + 4) \cdot 2^m)$ AOs.

Finally, we consider the differential-linear attacks. In Indesteege et al.'s chosen-plaintext attack on 11-round Serpent [26], two active plaintext boxes (both Serpent S4) contribute bit y_4 to the attack. Since three input differences cause this bit to flip with bias ± 0.5 , the attack recovers 56 key bits instead of 60. For the other differential-linear attacks [26], all key bits are obtainable and the effect of the bias ± 0.25 parity flips on P_s is negligible.

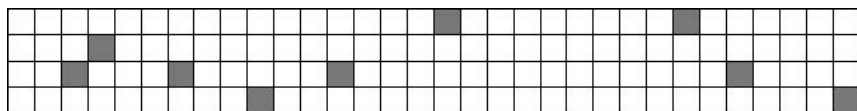


Fig. 6. Active plaintext S-boxes in Biham et al.'s reverse-direction CC differential-linear attack

4.2 Nonlinear Attacks on 11-round Serpent-192 and Serpent-256.

In this section, we describe various modifications to Collard et al.'s Approximation D2 [25], which will allow us to attack 11-round Serpent using nonlinear approximations.

The simplest change is to replace the (input bitmask 12, output bitmask 10) approximation in the first round (affecting bits 16, 17, 18, 19) with $x_2 \oplus x_1 \oplus x_4 = y_1 \oplus y_3$. Doing this gives us 20 active plaintext S-boxes, increasing $|k_1|$ to 128. However, four such boxes contribute a bit (or parity) that can flip with bias ± 0.5 for various input differences, as does one of the active ciphertext boxes. We can therefore recover only 118 bits of k_1 . One of the bits affected by this is involved in the quadratic term of the nonlinear approximation, so we cannot recover any k_2 bits. The memory requirements are increased to $2^{(128+2)} + 2^{(128+5)} = 2^{133.17}$ bytes. The time complexity of the analysis phase also increases, and is dominated by $4 \cdot (6 \times 128 + 8) \cdot 2^{128} = 2^{139.6}$ AOs and $4 \cdot (6 \times 128 + 26) \cdot 2^{128} = 2^{139.63}$ MAs.

We estimate the number of AOs per reduced-round encryption by counting the number of AOs in Serpent's bitslice implementation [32]. We assume that Osvik's implementation of S-box 6 [33] is used. This gives us $2^{12.65}$ AOs per 11-round encryption, and $2^{12.78}$ per 12-round encryption. We therefore obtain time complexity of $2^{126.95}$ encryptions + $2^{139.63}$ MAs.

The capacity C is multiplied by $(6/4)^2 = 2.25$. The increased number of k_1 bits, and the need to deal with 2^2 relateds, minus the ten k_1 bits which cannot be deduced, effectively raise k to 120 for the purposes of calculating N . We obtain $P_s = 0.8$ with $N = 2^{120.357}$.

Table 2. Attack complexities. Memory is measured in bytes. The memory required for the attack of [18] when the analysis method of [10] is not used is unclear, and the relevant sources [18][10] disagree on this. In most cases $P_s = 0.785$ (or slightly higher.) The chosen plaintext attacks of Biham et al. have $P_s = 0.84$, and the chosen-ciphertext attack has $P_s = 0.93$. The time complexity for Biham et al.'s linear cryptanalysis varies depending on whether the new analysis method of Collard et al. is used, or whether an earlier method [18] is. Table entries in bold signify that the method may not work as claimed depending on the linear hull effect. E = full encryptions of the reduced round cipher. PE = partial encryptions. PD = partial decryptions.

Rounds	Type of attack	Data	Time (analysis)	Mem	Bits recovered
11	Linear [18]	$2^{121.728}$ KP	$2^{188.1}$ E	*	130
11	Linear [18]	$2^{121.728}$ KP	2^{96} PE + 2^{44} PD + $2^{149.73}$ AO + $2^{149.76}$ MA	$2^{144.087}$	130
11	Linear [10]	$2^{121.234}$ KP	2^{60} PE + 2^{48} PD + $2^{117.36}$ AO + $2^{117.4}$ MA	$2^{112.087}$	98
11	Multidim. lin. [14]	$2^{125.813}$ KP	2^{48} PD + $2^{114.087}$ AO + $2^{114.134}$ MA	2^{108}	46
11	Multidim. lin. [14]	$2^{127.784}$ KP	2^{44} PD + $2^{110.055}$ AO + $2^{110.103}$ MA	2^{104}	44
11	Differential-linear [26]	$2^{121.8}$ CP	$2^{135.7}$ MA	2^{76}	48
11	Nonlinear (this paper)	$2^{120.357}$ KP	2^{80} PE + 2^{48} PD + $2^{139.6}$ AO + $2^{139.63}$ MA	$2^{133.17}$	118 k_1
11	Nonlinear (this paper)	$2^{117.317}$ KP	2^{60} PE + 2^{76} PD + $2^{149.69}$ AO + $2^{149.72}$ MA	$2^{141.585}$	128 k_1 , 4 k_2
11	Filtered NL (this paper)	$2^{116.508}$ KP	2^{60} PE + 2^{76} PD + $2^{151.69}$ AO + $2^{151.72}$ MA	$2^{142.585}$	128 k_1 , 6 k_2
11	Nonlinear (this paper)	$2^{115.44}$ KP	2^{60} PE + 2^{80} PD + $2^{153.73}$ AO + $2^{153.76}$ MA	$2^{145.585}$	130 k_1 , 4 k_2
11	Filtered NL (this paper)	$2^{114.55}$ KP	2^{60} PE + 2^{80} PD + $2^{155.73}$ AO + $2^{155.76}$ MA	$2^{146.585}$	132 k_1 , 6 k_2
11	Differential-linear [26]	$2^{113.7}$ CC	$2^{137.7}$ MA	2^{99}	56
12	Differential-linear [26]	$2^{123.5}$ CP	$2^{249.4}$ E	$2^{128.5}$	160
12	Multidim. lin. [14]	$\geq 2^{125.813}$ KP	2^{128} PE + 2^{48} PD + $2^{242.087}$ AO + $2^{242.134}$ MA	2^{108}	174

Table 3. Attack complexities cont. $2^{12.65}$ AOs are needed for an 11-round Serpent encryption, and $2^{12.78}$ AOs for twelve rounds

Rounds	Type of attack	Time (analysis) summary
11	Linear [18]	$2^{188.1}$ E
11	Linear [18]	$2^{137.08}$ E + $2^{149.76}$ MA
11	Linear [10]	$2^{104.71}$ E + $2^{117.4}$ MA
11	Multidim. linear [14]	$2^{101.437}$ E + $2^{114.134}$ MA
11	Multidim. linear [14]	$2^{97.405}$ E + $2^{110.103}$ MA
11	Differential-linear [26]	$2^{135.7}$ MA
11	Nonlinear (this paper)	$2^{126.95}$ E + $2^{139.63}$ MA
11	Nonlinear (this paper)	$2^{137.04}$ E + $2^{149.72}$ MA
11	Filtered NL (this paper)	$2^{139.04}$ E + $2^{151.72}$ MA
11	Nonlinear (this paper)	$2^{141.08}$ E + $2^{153.76}$ MA
11	Filtered NL (this paper)	$2^{143.08}$ E + $2^{155.76}$ MA
11	Differential-linear [26]	$2^{137.7}$ MA
12	Differential-linear [26]	$2^{249.4}$ E
12	Multidim. linear [14]	$2^{229.437}$ E + $2^{242.134}$ MA

We now consider a situation in which the entire first round approximation remains linear. We replace the final-round $x_1 \oplus x_3 \oplus x_4 = y_2$ approximation on state bits 96-99 with $x_1 \oplus x_3 \oplus x_4 = y_2 \oplus y_1 \oplus y_2 y_4$, and also replace $x_3 \oplus x_4 = y_4$ (bits 76-79) with $x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_1 y_3$. The total number of active S-boxes increases to 34. We have replaced a bias 4 approximation and a bias 2 approximation with two bias 6 approximations, multiplying C by $\left(\frac{6 \times 6}{4 \times 2}\right)^2 = 20.25$. For the purposes of calculating N , the value of k is effectively increased to $(140-8) = 132$, and $N = 2^{117.317}$ gives $P_s \approx 0.8$. The memory requirements are increased to $2^{141.585}$. The time complexity of the analysis phase is dominated by $16 \cdot (6 \cdot 136 + 26) \cdot 2^{136} = 2^{149.72}$ MAs and $16 \cdot (6 \cdot 136 + 8) \cdot 2^{136} = 2^{149.69}$ AOs.

If we utilise filtering here, the number of active S-boxes does not change. The biases of the eight relateds for each of the S-box approximations become $(6, -4, -4, 2, 0, 0, 0, 0)$, allowing us to attack all six k_2 bits. Memory requirements increase to $2^{142.585}$, and the time complexity of the analysis phase increases to $64 \cdot (6 \cdot 136 + 26) \cdot 2^{136} = 2^{151.72}$ MAs and $64 \cdot (6 \cdot 136 + 8) \cdot 2^{136} = 2^{151.69}$ AOs. For the purposes of calculating N , k is effectively equal to 134, and C is multiplied by $(16/9)^2$. However, the need to effectively discard 9/16 of our KP pairs means we must calculate N as if it were only multiplied by $(16/9)$, and we obtain $N = 2^{116.508}$.

To reduce the data complexity further, we could replace the $x_1 \oplus x_3 \oplus x_4 = y_2$ approximation on state bits 116-119 with a nonlinear approximation, instead of replacing $x_3 \oplus x_4 = y_4$. C is then multiplied by 81 instead of 20.25, and we activate 35 S-boxes. We obtain time complexity $16 \cdot (6 \cdot 140 + 26) \cdot 2^{140} = 2^{153.76}$ MAs and $16 \cdot (6 \cdot 140 + 8) \cdot 2^{140} = 2^{153.73}$ AOs with memory complexity $2^{145.585}$. For the purposes of calculating N , k is effectively increased to $(144-10)=134$, and $N = 2^{115.338}$ yields $P_s = 0.8$.

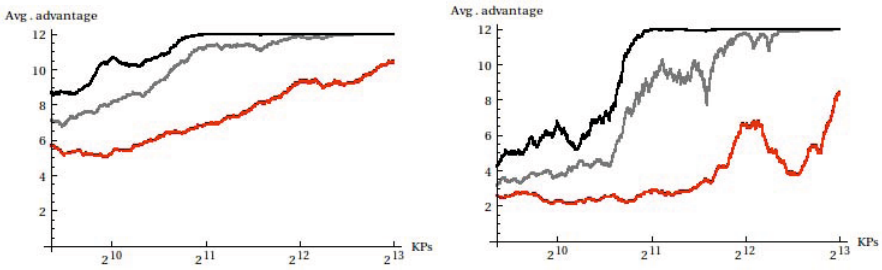


Fig. 7. Left-hand graph shows mean advantages for attack on four round SPN with 4×4 S-boxes using: linear approximation (red), nonlinear approximation (grey), nonlinear approximation with filtering (black). Right-hand graph shows results of alternate calculation for average advantage in which the mean rank obtained is input to the formula for advantage.

Again, we can employ filtering here. The number of active S-boxes is still 35, and all six k_2 bits can now be attacked. The memory complexity increases to $2^{146.585}$, and the time complexity to $2^{155.76}$ MAs and $2^{155.73}$ AOs. To calculate N , since one less S-box is affected by the “WKRH max-bias” issue than before, k is effectively $(146-8)=138$ and we obtain $N = 2^{114.55}$.

5 Conclusion

We have obtained nonlinear approximations for block cipher S-boxes with higher absolute bias than their best linear approximations. We have also derived algorithms which can use the new approximations in attacks, and calculated the complexities for these new attacks. Having done this, we have presented nonlinear attacks on 11-round Serpent with better data complexity than any other known-plaintext attack, as well as the best time complexity of any attack so far on 11-round Serpent-256.

References

1. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseht, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
2. Matsui, M.: The first experimental cryptanalysis of the Data Encryption Standard. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 1–11. Springer, Heidelberg (1994)
3. Harpes, C., Kramer, G.G., Massey, J.L.: 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)
4. Knudsen, L.R., Robshaw, M.: Non-linear approximations in linear cryptanalysis. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 224–236. Springer, Heidelberg (1996)

5. Courtois, N.T.: Feistel schemes and bi-linear cryptanalysis (extended abstract). In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 23–40. Springer, Heidelberg (2004)
6. Courtois, N.: Feistel schemes and bi-linear cryptanalysis. Cryptology ePrint Archive, Report 2005/251 (August 2005), <http://eprint.iacr.org/2005/251>
7. Tapiador, J.M.E., Clark, J.A., Hernandez-Castro, J.C.: Non-linear cryptanalysis revisited: Heuristic search for approximations to S-boxes. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 99–117. Springer, Heidelberg (2007)
8. Burwick, C., Coppersmith, D., D’Avignon, E., Gennaro, R., Halevi, S., Jutla, C.: Jr, S.M., O’Connor, L., Peyravian, M., Safford, D., Zunic, N.: MARS - a candidate cipher for AES. Technical report, IBM (September 1999), <http://www.research.ibm.com/security/mars.pdf>
9. Clark, J., McLaughlin, J.: Nonlinear cryptanalysis of reduced-round Serpent and metaheuristic search for s-box approximations. Cryptology ePrint Archive, Report 2013/ (January 2013), <http://eprint.iacr.org/2013/>
10. 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)
11. Murphy, S.: The effectiveness of the linear hull effect. Technical Report RHUL-MA-2009-19, Royal Holloway, University of London (October 2009), http://www.isg.rhul.ac.uk/~sean/Linear_Hull_JMC-Rev2-11nsc.pdf
12. Leander, G.: On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 303–322. Springer, Heidelberg (2011)
13. Collard, B., Standaert, F.X.: Experimenting linear cryptanalysis (2011), <http://perso.uclouvain.be/fstandae/PUBLIS/90.pdf>
14. Nguyen, P.H., Wu, H., Wang, H.: Improving the algorithm 2 in multidimensional linear cryptanalysis. In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 61–74. Springer, Heidelberg (2011)
15. Carlet, C.: Boolean functions for cryptography and error-correcting codes. In: Crama, Y., Hammer, P. (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering, Cambridge University Press, Cambridge (2010), The chapter is downloadable from <http://www.math.univ-paris13.fr/~carlet/chap-fcts-Bool-corr.pdf>.
16. Kunz, H.: On the equivalence between one-dimensional discrete Walsh-Hadamard and multidimensional discrete Fourier transforms. IEEE Transactions on Computers C-28(3), 267–268 (1979)
17. Dunkelman, O.: Private communication
18. Biham, E., Dunkelman, O., Keller, N.: Linear cryptanalysis of reduced round Serpent. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 16–27. Springer, Heidelberg (2002)
19. Biham, E., Dunkelman, O., Keller, N.: Differential-linear cryptanalysis of Serpent. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 9–21. Springer, Heidelberg (2003)
20. Biham, E., Dunkelman, O., Keller, N.: New results on boomerang and rectangle attacks. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 1–16. Springer, Heidelberg (2002)
21. Preneel, B., Rompay, B.V., Ors, S.B., Biryukov, A., Granboulan, L., Dottax, E., Dichtl, M., Schafheutle, M., Serf, P., Pyka, S., Biham, E., Barkan, E., Dunkelman,

- O., Stolin, J., Ciet, M., Quisquater, J.J., Sica, F., Raddum, H., Parker, M.: Performance of optimized implementations of the NESSIE primitives (version 2.0) (February 2003), <http://www.cosic.esat.kuleuven.be/nessie/deliverables/D21-v2.pdf>
22. Hermelin, M., Nyberg, K.: Dependent linear approximations: The algorithm of Biryukov and others revisited. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 318–333. Springer, Heidelberg (2010)
 23. Courtois, N., Hulme, D., Mourouzis, T.: Solving circuit optimisation problems in cryptography and cryptanalysis. Cryptology ePrint Archive, Report 2011/475 (September 2011), <http://eprint.iacr.org/2011/475>
 24. Collard, B., Standaert, F.-X., Quisquater, J.-J.: Improved and multiple linear cryptanalysis of reduced round Serpent. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 51–65. Springer, Heidelberg (2008)
 25. Collard, B., Standaert, F.X., Quisquater, J.J.: Improved and multiple linear cryptanalysis of reduced round Serpent - description of the linear approximations (2007), <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.522&rep=rep1&type=pdf>
 26. Dunkelman, O., Indestege, S., Keller, N.: A differential-linear attack on 12-round Serpent. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 308–321. Springer, Heidelberg (2008)
 27. Selçuk, A.: On probability of success in linear and differential cryptanalysis. *Journal of Cryptology* 21, 131–147 (2008)
 28. 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)
 29. David, H.: *Order Statistics*. 2nd edn. Wiley (1981)
 30. Leander, G.: Small scale variants of the block cipher PRESENT. Cryptology ePrint Archive, Report 2010/143 (March 2010), <http://eprint.iacr.org/2010/143>
 31. Cover, T., Thomas, J.: *Elements of Information Theory*. 2nd edn. Wiley-Interscience (2006)
 32. Anderson, R., Biham, E., Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard, <http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf>
 33. Osvik, D.: Speeding up Serpent. In: Proceedings of the 3rd Advanced Encryption Standard Candidate Conference, AES 2000 (April 2000)

A Errors in the Description of the Dunkelman/Keller Approximation

In the original description of Biham et al.’s linear approximation [18], on page 20, after S_6 is applied the only active bit in the state is bit 30. In later papers [19][26], after the application of S_6 , bit 28 is shown as active instead of bit 30. One of the authors has informed us by email that bit 28 was correct.

The Serpent diffusion layer is then applied, after which the active bits according to the diagram are 80, 101 and 103. However, the xor of diffusion layer output bits $\{80, 101, 103\}$ is the xor of input bits $\{4, 22, 35, 44, 46, 57, 62, 75, 86, 96, 97\}$ - and is therefore unaffected by either bit 28 or bit 30. In the same correspondence mentioned above, this was revealed to be a typographical error - the active bits shown at this point should have been 81, 83 and 100.

Differential Cryptanalysis of Keccak Variants

Stefan Kölbl, Florian Mendel, Tomislav Nad, and Martin Schläffer

IAIK, Graz University of Technology, Austria

Abstract. In October 2012, NIST has announced Keccak as the winner of the SHA-3 cryptographic hash function competition. Recently, at CT-RSA 2013, NIST brought up the idea to standardize Keccak variants with different parameters than those submitted to the SHA-3 competition. In particular, NIST considers to reduce the capacity to the output size of the SHA-3 standard and additionally, standardize a Keccak variant with a permutation size of 800 instead of 1600 bits. However, these variants have not been analyzed very well during the SHA-3 competition. Especially for the variant using an 800-bit permutation no analysis on the hash function has been published so far.

In this work, we analyze these newly proposed Keccak variants and provide practical collisions for up to 4 rounds for all output sizes by constructing internal collisions. Our attacks are based on standard differential cryptanalysis contrary to the recent attacks by Dinur et al. from FSE 2013. We use a non-linear low probability path for the first two rounds and use methods from coding theory to find a high-probability path for the last two rounds. The low probability path as well as the conforming message pair is found using an automatic differential path search tool. Our results indicate that reducing the capacity slightly improves attacks, while reducing the permutation size degrades attacks on Keccak.

Keywords: hash functions, SHA-3, collision attack, differential cryptanalysis.

1 Introduction

In October 2012, NIST has announced Keccak [2] as the winner of the SHA-3 cryptographic hash function competition [16], which was held between 2008 and 2012 [15]. Traditionally, cryptographic hash functions take as input a string of arbitrary finite length and produce a fixed sized output of n bits. As a consequence, the following main security requirements are defined for cryptographic hash functions:

- **Preimage Resistance:** For a given output y it should be computationally infeasible to find any input x' such that $y = f(x')$.
- **Second Preimage Resistance:** For given $x, y = f(x)$ it should be computationally infeasible to find any $x' \neq x$ such that $y = f(x')$.
- **Collision Resistance:** It should be computationally infeasible to find two distinct inputs x, x' such that $f(x) = f(x')$.

Table 1. Summary of collision attacks on Keccak

Variants	Hash Size	Capacity	Permutation	Rounds	Complexity	Reference
Keccak ($c = 2n$)	224	448	1600	4	practical	[8]
	256	512	1600	4	practical	[8]
	256	512	1600	5	2^{115}	[9]
	384	768	1600	3	practical	[9]
	384	768	1600	4	2^{147}	[9]
	512	1024	1600	3	practical	[9]
Keccak ($c = n$)	224	224	1600	4	practical	this work
	256	256	1600	4	practical	this work
	384	384	1600	4	practical	this work
	512	512	1600	4	practical	this work
	224	224	800	4	practical	this work
	256	256	800	4	practical	this work
	384	384	800	4	2^{102}	this work
other variants	all	≤ 640	1600	4	practical	this work
	all	≤ 352	800	4	practical	this work

For any ideal hash function with n -bit output size, we can find preimages or second preimages with a complexity of 2^n , and collisions with a complexity of $2^{n/2}$ using generic attacks. However, in practice the security of a hash function is not necessarily linked to the hash output size. This is addressed by the sponge construction [1] which is the underlying design principle of Keccak. The sponge construction provides an internal capacity of c bits and allows an arbitrary hash value output size of n bits. Therefore, the security is given by $s = \min(c/2, n)$ bits against (second-) preimage attacks and by $s = \min(c/2, n/2)$ bits against collision attacks. As a consequence, the SHA-3 candidates of Keccak submitted to the competition were defined with a capacity of $c = 2n$ bits.

At CT-RSA 2013, NIST proposed the idea to standardize Keccak variants with different parameters than those submitted to the SHA-3 competition. More specifically, NIST proposes to reduce the capacity c to n instead of $2n$ bits. Due to the reduced capacity, NIST may also consider to standardize a smaller 800-bit permutation for small capacities. Unfortunately, these variants have not been analyzed very well during the SHA-3 competition.

Our Contribution. In this work, we analyze these new and the original variants of Keccak using the same attack strategy. We use a standard differential attack, which allows us to better compare the security of all variants. Our results show that reducing the capacity does not lead to much better differential attacks on Keccak. On the other hand, reducing the permutation size b from 1600 to 800 bits even increases the security against differential attacks. To summarize, we are able to provide practical results for up to 4 rounds of all Keccak variants proposed for SHA-3 with a permutation size of 1600 bits. This includes the Keccak variants supporting arbitrary output sizes. A summary of our results and related work can be found in Table 1.

Related Work. The collision resistance of Keccak with permutation size of $b = 1600$ has already been investigated by a number of researchers. The first practical attack on Keccak with $c = 512$ and $n = 256$ has been published by Naya-Plasencia et al. in [17]. They have presented a 2-round collision attack which uses an efficient method to find high probability differential characteristics using the column parity (or kernel) property. By using these characteristics and connecting them with the input using an algebraic method, Dinur et al. have presented a 4-round collision attack for Keccak with $c = 448$ and $n = 224$, and for Keccak with $c = 512$ and $n = 256$ in [8]. Furthermore, in [9] Dinur et al. have presented the first attacks on reduced Keccak with $c = 768$ and $n = 384$, and Keccak with $c = 1024$ and $n = 512$. For both variants practical collision attacks on 3 rounds were shown. Moreover, they have shown theoretical attacks on Keccak with $c = 512$ and $n = 256$ reduced to 5 rounds and Keccak with $c = 768$ and $n = 384$ reduced to 4 rounds with a complexity of 2^{115} and 2^{147} , respectively.

Outline. The paper is structured as follows. In Section 2 we provide a short description of Keccak. In Section 3 we give an overview of the basic attack strategy. Section 4 presents our method to find high probability characteristics. In Section 5, we discuss our approach to connect these characteristics to the input, using an automated search tool.

2 Description of Keccak

Keccak is a family of hash functions based on the sponge construction, with state sizes $b \in \{25, 50, 100, 200, 400, 800, 1600\}$. Keccak uses a b -bit permutation Keccak- $f[b]$ and a multi-rate padding scheme. A specific instance of Keccak is defined by the two parameters r (the rate) and c (the capacity) with $b = r + c$. For the NIST SHA-3 competition, the Keccak designers have defined one instance for each output size $n \in \{224, 256, 384, 512\}$ bits. All four instances use the 1600-bit permutation with capacity $c = 2n$, where n is the hash output size. Additionally, to these four variants the Keccak designers also specify a variant supporting arbitrary output sizes which they named Keccak[\square]. This variant has capacity $c = 576$ and rate $r = 1024$.

The permutation Keccak- f used in Keccak operates on a three-dimensional state with elements in \mathbb{F}_2 . The dimensions for this state are $5 \times 5 \times w$ with $w \in \{1, 2, 4, 8, 16, 32, 64\}$. This allows to represent each lane as a w -bit word. A three-dimensional array is used, $S[x][y][z]$, to describe the state. The hash h for a message m is computed in the following way for Keccak with rate r and capacity c :

1. Initialise the state $S[x][y][z] = 0$ for $x = 0 \dots 4$, $y = 0 \dots 4$ and $z = 0 \dots w$.
2. Compute the padded message $M = m || 10^*1$ such that M is a multiple of r .
3. Absorb the next r -bit message block by computing $S[x][y] = S[x][y] \oplus M_i$ and update the state by computing $S = \text{Keccak-}f(S)$. Repeat this process until all message blocks are absorbed.

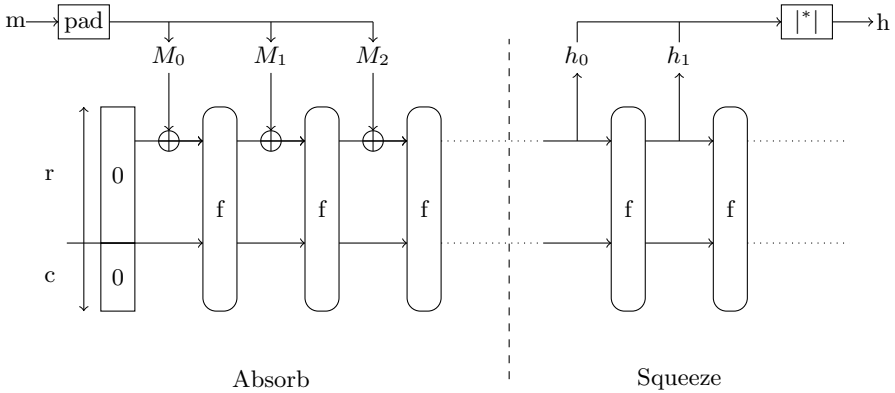


Fig. 1. Overview of the sponge construction, which is used in the Keccak hash function

- Concatenate the first r bits to the hash value. Compute $S = \text{Keccak-}f(S)$ and repeat this process to get the desired number of output bits.

This procedure is also outlined in Fig. 1.

2.1 Keccak- f

Keccak uses the iterative permutation Keccak- f operating on \mathbb{F}_2^w , with w being the lane size. The permutation consists of multiple rounds in which five functions are used in sequence $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$. The number of rounds n_r depends on the lane size:

$$n_r = 12 + 2 \log_2(w) \tag{1}$$

Apart from ι , all functions are equivalent for each round.

Description of θ . This step adds to every bit of the state $S[x][y][z]$ the bitwise sum of the neighbouring columns $S[x - 1][*][z]$ and $S[x + 1][*][z - 1]$. This procedure can be described by the following equation:

$$\theta : S[x][y][z] \leftarrow S[x][y][z] + \sum_{n=0}^4 S[x - 1][n][z] + \sum_{n=0}^4 S[x + 1][n][z - 1] \tag{2}$$

Description of ρ . This step rotates the bits in every lane by a constant value.

$$\rho : S[x][y][z] \leftarrow S[x][y][z + C(x, y)] \tag{3}$$

where $C(x, y)$ is a constant value.

Description of π . This function transposes the lanes using the following function:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} \tag{4}$$

Description of χ . This step is the only non-linear step in Keccak and operates on each row of 5 bits.

$$\chi : S[x][y][z] \leftarrow S[x][y][z] \oplus ((-S[x + 1][y][z]) \wedge S[x + 2][y][z]) \quad (5)$$

It can be seen as applying a 5-bit S-box in parallel to all rows.

Description of ι . This step adds a round dependent constant to the state. For a list of the constants see [2].

3 Differential Cryptanalysis of Keccak

In this section, we give a brief overview of our attack strategy on the Keccak hash function. We use standard differential cryptanalysis which has first been published to cryptanalyze the block cipher DES [3] and was later applied to hash functions as well. The basic idea of our attack is the same as used by Wang et al. in the cryptanalysis of the MD4-family of hash functions [19, 20]. The hash function is split into two parts. We first construct a high-probability differential characteristic for the second part and then, use a low-probability differential characteristic and message modification to connect with the input in the first part. In more detail, we perform the following 4 steps:

1. Find a differential characteristic for the hash function that results in a collision and holds with a high probability for the last few rounds of the hash function.
2. Find a differential characteristic (not necessary with high probability) for the first few rounds of the hash function.
3. Use message modification techniques to find conforming message pairs for the differential characteristic in the first few rounds.
4. Use the message pairs of the previous step to find a solution for the high-probability characteristic in the last few rounds of the hash function.

This strategy has already been used in [8] by Dinur et al. in the attack on 4 rounds of Keccak with $c = 512$ and $n = 256$. In this paper, we revisit the attack, extend it, and apply it to other variants of Keccak. Our attack differs from the attack of Dinur et al. in several ways. First of all, we show how to construct high probability differential characteristics for the last few rounds of Keccak that result in a (internal) collision for more than 256 bits. This allows us to construct collisions for larger output sizes of Keccak reduced to 4 rounds including the variant of Keccak supporting variable output sizes.

Second, we present a more general technique to perform Step 2 and 3 of the attack. We use an automatic search tool implementing a guess-and-determine strategy that constructs a differential characteristic and uses message modification techniques to find conforming message pairs. For this purpose, we use a similar tool as published by Mendel et al. in the analysis of SHA-2 [13, 14]. Fig. 2 shows an high-level overview of our attack strategy.

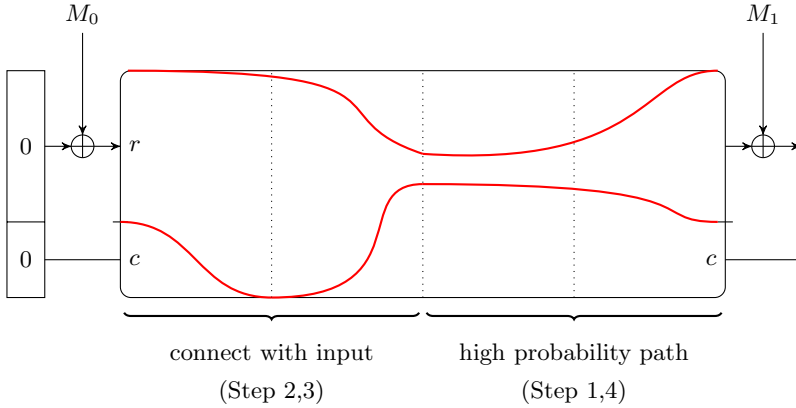


Fig. 2. Outline of our 4-round differential attack strategy

In Step 1 of the attack, we search for 2-round high-probability characteristics which lead to (internal) collisions. To find collisions for larger output sizes than in [8], we use a linearized version of Keccak and methods from coding theory (see Sect. 4).

In Step 2 and Step 3 of the attack, we need to connect the input difference of the high-probability characteristic with the fixed input value given by the capacity c . We solve this problem by searching for a differential characteristic and conforming message pair using an automatic search tool (see Sect. 5). The difficulty of finding a solution depends on the size of the capacity c . By improving the search strategy of our tool, we are able to solve the problem for larger values of the capacity c .

4 Finding Colliding High-Probability Characteristics

High-probability differential characteristics can be constructed for Keccak by using the column parity property of θ . Using this property, Naya-Plasencia et al. presented the first practical collision attacks on round-reduced versions of Keccak in [17]. Similar differential characteristics over 2 rounds were also used by Dinur et al. in [8]. A method to construct all column parity paths up to a given Hamming weight is described in [17], and a full characterization of kernel paths was done in [6]. However, no low-weight paths over three consecutive rounds exist [2].

4.1 Differential Characteristics and Coding Theory

To find a good characteristic for 2 rounds of Keccak, we use a linearized model of the Keccak hash function. Therefore, we replace all non-linear operations by a linear approximation resulting in a linear code over \mathbb{F}_2 . Finding characteristic in the linear code is not difficult, since it depends only on the differences

at the input. The probability that the characteristic holds in the original hash function is related to the Hamming weight of the characteristic. In general, a characteristic with low Hamming weight has a higher probability than one with a high Hamming weight. Hence, for finding a characteristic with high probability, i.e. with low Hamming weight, we use probabilistic algorithms from coding theory. It has been shown in the past [4, 11, 12, 18] that these algorithms work quite well. Furthermore, we can impose additional restrictions on the characteristic by forcing certain bits/words to zero. Note that this is needed to find suitable characteristics for Keccak resulting in an (internal) collision for the hash function. In the following we will briefly discuss the linear approximation and algorithms we were using to find the characteristics.

Linear Approximation of Keccak. The only non-linear transformation in Keccak is χ . There are many ways to approximate χ by a linear function. For our analysis we decided to use the identity function, since it comes very close to the original definition and we are aiming for sparse characteristics.

$$\chi : S[x][y][z] \leftarrow S[x][y][z] \quad (6)$$

All other transformations are linear facilitating our approach.

4.2 Finding Low-Weight Codewords

To find codewords with low Hamming weight we use the publicly available CodingTool Library¹. It implements the probabilistic algorithm from Canteaut and Chabaud [5] to search for codewords with low Hamming weight. Moreover, it provides some other usable functionalities that turned out to be very useful for our purpose. With this tool we can find good characteristics for different choices of c and n in a few seconds on a standard PC. Table 2 and Table 3 show the best (lowest Hamming weight) characteristics we have found for a different set of parameters. It has to be noted that we can use these characteristics to construct internal collisions for Keccak with capacity up to 416 resp. 832 bits. However, for Keccak with variants with $c = 2n$ this is too small to attack versions with output sizes larger than 208 resp. 416 bits. Therefore, we also give the results for characteristics resulting only in a collision for the hash function. The results are characteristics that can be used for collision attacks for up to 448 resp. 832 bits.

Using this general approach the whole (linear) search space is covered and arbitrary differences in the state words are possible. However, it turned out that the best characteristics we have found are indeed column parity kernel paths. In hindsight the same differential characteristics could have been found using the method described in [17].

Extending the Approach to 3 Rounds. Using the same method one could try to construct differential characteristics for more than 2 rounds. Unfortunately, we did not find any sparse solutions which is conform with the work by

¹ <http://www.iaik.tugraz.at/content/research/krypto/codingtool/>

Table 2. Low-weight differential characteristics for 2 rounds resulting in an internal collision for Keccak with capacity c .

Permutation	Capacity	Weight	Kernel Path
800	320	16	yes
	352	16	yes
	384	16	yes
	416	20	yes
1600	640	20	yes
	704	20	yes
	768	20	yes
	832	28	yes

Table 3. Low-weight differential characteristics for 2 rounds resulting in a collision for Keccak with hash size n

Permutation	Hash Size	Weight	Kernel Path
800	320	12	yes
	352	12	yes
	384	20	yes
	416	30	no
	448	32	yes
1600	512	16	yes
	640	20	yes
	704	20	yes
	768	20	yes
	832	28	no

Daemen and Van Assche in [6]. Another approach we tried was to non-linearly propagate the linear paths for 2 rounds forward using the automatic tool described in Sect. 5. As we can not linearly combine these paths, we use a brute force algorithm to check if a combination results in a collision after three rounds. However, since the search space is too large to cover, we restricted ourselves to a combination of only a few candidates. Unfortunately, we could not find a solution, which confirms that a sparse 3-round path is unlikely to exist.

5 Non-linear Characteristics and Message Modification

Once we have determined a high-probability characteristic for the second half of Keccak, we need to connect this path with the constraints at the input of the Keccak permutation. In [8], Dinur et al. have used their target difference algorithm to find a solution for both differences and values of the input message.

In our work, we use the improved differential path search algorithm of Mendel et al. [14]. Using this automated search tool, complex nonlinear differential characteristics can be found. Additionally, the tool can be used for solving nonlinear equations involving conditions on state and message words (i.e. for message modification).

Using the bitsliced propagation method used in [14], we were not able to find a solution for the first two rounds of Keccak. The problem is, that the linear functions used in Keccak are significantly larger than the linear functions used in SHA-2. However, using the linear propagation method proposed in [10] and some other minor improvements, we are able to find solutions for the first two rounds of Keccak. In the following, we give a brief description of the search algorithm.

5.1 Search for Differential Characteristics and Message Pairs

The basic idea of this search algorithm is to pick and guess previously unrestricted bits. After each guess, the information due to these restrictions is propagated to other bits. If an inconsistency occurs, the algorithm backtracks to an earlier state of the search and tries to correct it. Similar to [14], we denote these three parts of the search by decision (guessing), deduction (propagation), and backtracking (correction). Then, the search algorithm proceeds as follows.

Let U be a set of bits. Repeat the following until U is empty:

Decision (Guessing)

1. Pick randomly (or according to some heuristic) a bit in U .
2. Impose new constraints on this bit.

Deduction (Propagation)

3. Propagate the new information to other variables and equations as described in [14].
4. If an inconsistency is detected start backtracking, else continue with step 1.

Backtracking (Correction)

5. Try a different choice for the decision bit.
6. If all choices result in an inconsistency, mark the bit as critical.
7. Jump back until the critical bit can be resolved.
8. Continue with step 1.

In the deduction, we use generalized conditions on bits [7]. To propagate information, we use the techniques of [10] and during the search, we backtrack as shown in [14]. In the message search, we additionally consider linear conditions on two related bits ($X_j \oplus X_k = \{0, 1\}$) as proposed in [13].

Note that in each stage different bits are chosen (guessed). In total we have two stages which can be summarized as follows.

1. **Characteristic Search:** In the first phase we search for a consistent differential characteristic in the state words. Therefore, we only add unconstrained bits '?' to the set U .
2. **Message Search:** In the second stage we search for a conforming message. In this phase, we only add bits with many linear two-bit conditions to the set U . This ensures that bits which influence a lot of other bits are guessed first.

Note that we dynamically switch between the two stages. Additionally, we restart the search from scratch after a certain amount of inconsistencies to terminate branches which appear to be stuck because of exploring a search space far from a solution.

5.2 Improved Linear Propagation in Keccak

Using the bitsliced propagation method used in [14], we were not able to find a solution for the first two rounds of Keccak. The problem is, that the linear layer $\lambda = \pi \circ \rho \circ \theta$ of Keccak is significantly larger than the linear functions used in SHA-2. We have tried to split the linear layer into bitslices and at least 320 bitslices are needed. In this case the linear information propagates very badly and many contradictions in the linear layer are not detected.

To avoid this problem, we use the linear propagation method of [10]. In this case, a linear system of equations is defined, which contains all equations of the linear functions for X_i and X_i^* , as well as the equations for the linear generalized conditions at the input and output of the function. The resulting system of equation is solved using Gauss-Jordan elimination to detect contradictions and propagate information.

We get the best results when applying this linear approach to the complete linear layer $\lambda = \pi \circ \rho \circ \theta$ of Keccak. We have also performed experiments which include the XORs of the S-box layer χ , or by combining the linear parts of two Keccak rounds. However, the best performance/propagation trade-off was achieved for the linear layer λ .

5.3 Finding Solutions for 2 Rounds of Keccak

Using the automated search tool combined with the linear propagation method, we can efficiently find both, differential characteristics and conforming message pairs for up to two rounds of Keccak. However, the difficulty of finding a solution depends on fine-tuning of the search algorithm based on a number of parameters.

The parameter which influences the search most is the capacity c . If too many bits are fixed by the capacity, then we are not able to find a solution. For the 1600-bit permutation, we could find solutions for capacities of up to 640 bits (ten 64-bit lanes), and for the 800-bit permutation, we could find solutions for capacities of up to 352 bits (eleven 32-bit lanes).

Our experiments have shown, that for a zero value at the input it is harder to find a solution, in particular for larger capacities. The running time of the search algorithm can be improved by prepending a random first message block with random differences. This was used for the results given in the appendix.

6 Results

For the 1600-bit permutation reduced to 4 rounds with a capacity of $c \leq 640$ we can find internal collisions and hence, collisions for the hash function with arbitrary output size. We want to note that that this also includes the Keccak variant with capacity $c = 576$, which was proposed by the Keccak designers for supporting variable output sizes. The confirming message pair and the according differential characteristic is given in Appendix A. Note that for the zero IV of Keccak, our automatic search tool does not work very well. By using a first

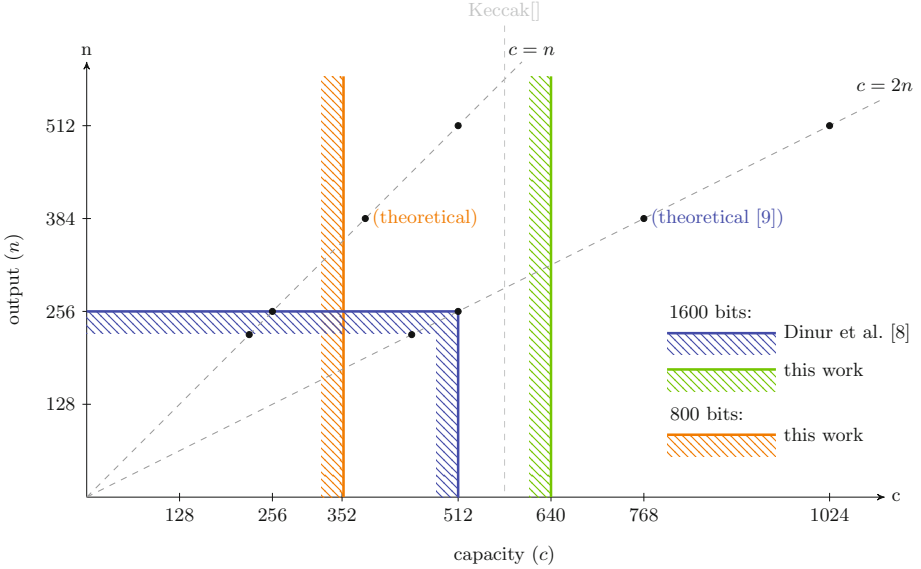


Fig. 3. Overview of all 4-round collision attacks on Keccak with permutation size of 1600 and 800 bits, respectively. Blue: attacks by Dinur et al. and green: our attacks on Keccak with permutation size of 1600 bits. Orange: our attacks on Keccak with permutation size of 800 bits. Additionally, a theoretical 5-round collision attack on Keccak-256 has been published in [9].

message block with differences (which could even be meaningful), the tool works much better. In this case a solution was found within minutes on a standard PC.

For the smaller 800-bit permutation we can show internal collisions for a capacity of $c \leq 352$ bits. A confirming message pair and the according differential characteristic is given in Appendix B. Finding this solution took about 140 minutes on a standard PC. This corresponds to about 2^{38} Keccak computations. Note that in an attack on the hash function with a capacity of 352 bits, the values and differences of 32 additional bits can be chosen freely. Based on this, we estimate the complexity to construct an internal collision with capacity 384 to be at most $2^{64+38} = 2^{104}$. However, we expect the complexity to be much smaller in practice. Our results are shown in Fig. 3.

Acknowledgements. This work has been supported in part by the Secure Information Technology Center-Austria (A-SIT), by the Austrian Science Fund (FWF) under grant number TRP 251-N23 (Realizing a Secure Internet of Things - ReSIT), and by the Austrian Research Promotion Agency (FFG) and Styrian Business Promotion Agency (SFG) under grant number 836628 (SeCoS).

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference. Submission to NIST (Round 3) (January 2011), http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html
3. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptology* 4(1), 3–72 (1991)
4. Brier, E., Khazaei, S., Meier, W., Peyrin, T.: Linearization Framework for Collision Attacks: Application to CubeHash and MD6. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 560–577. Springer, Heidelberg (2009)
5. Canteaut, A., Chabaud, F.: A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece’s Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Transactions on Information Theory* 44(1), 367–378 (1998)
6. Daemen, J., Van Assche, G.: Differential Propagation Analysis of Keccak. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 422–441. Springer, Heidelberg (2012)
7. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
8. Dinur, I., Dunkelman, O., Shamir, A.: New Attacks on Keccak-224 and Keccak-256. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 442–461. Springer, Heidelberg (2012)
9. Dinur, I., Dunkelman, O., Shamir, A.: Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials. In: Moriai, S. (ed.) FSE. LNCS, Springer (to appear, 2013)
10. Eichlseder, M., Mendel, F., Nad, T., Rijmen, V., Schläffer, M.: Linear Propagation in Efficient Guess-and-Determine Attacks. In: Budaghyan, L., Helleseht, T., Parker, M.G. (eds.) WCC (2013), <http://www.selmer.uib.no/WCC2013/>
11. Indestege, S., Preneel, B.: Practical Collisions for EnRUP. *J. Cryptology* 24(1), 1–23 (2011)
12. Mendel, F., Nad, T.: A Distinguisher for the Compression Function of SIMD-512. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 219–232. Springer, Heidelberg (2009)
13. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011)
14. Mendel, F., Nad, T., Schläffer, M.: Improving Local Collisions: New Attacks on Reduced SHA-256. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 262–278. Springer, Heidelberg (2013)
15. National Institute of Standards and Technology: Cryptographic Hash Algorithm Competition (November 2007), <http://csrc.nist.gov/groups/ST/hash/sha-3/index.html>
16. National Institute of Standards and Technology: SHA-3 Selection Announcement (October 2012), http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_selection_announcement.pdf

17. Naya-Plasencia, M., Röck, A., Meier, W.: Practical Analysis of Reduced-Round KECCAK. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 236–254. Springer, Heidelberg (2011)
18. Rijmen, V., Oswald, E.: Update on SHA-1. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 58–71. Springer, Heidelberg (2005)
19. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
20. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Results for Keccak with a 1600-Bit Permutation

Table 4. The 4-round characteristic used for the second block to find an internal collision for Keccak with a 1600-bit permutation and a capacity of 640. Note that a random first block was used in this case.

Round	State
0	737bc39f15b62ce3 4-ae-67d9-f67961 72c17e19ecf12b7b 2ba7b749c7949634 fc-cfc935859fb2e 3d196398efcd8-85 fce83de1dec57822 585c3e88-e91a216 7abfed54f57e1dd9 d9a96ed7944d8ede 147b6be6e6-24fdb --4a7743-1159181 -1df19ab97369543 77a1e8bca7-c--6f -5e697e1852d7fd5 1a9b2c7d9b5a9abf 2913f4ef6ca6b829 4--b84511febcb4ff 236c8edaa59db4a3 fa16a175b84e4326 6c34feb1242754fb cb2ea33a4c-db176 b2c5aa5a8-df6238 7bafafd7ee121941 8b4cf1f55781e-9f
1	96--3182f1fad467 22--9-644fa7e-f- de--54fb5f2e9a6b 7e--726f824-bd4c d2--114a6bb11583 96-171-2f1fad467 26--9-644fa7e-f- de--54fb5f2e9a6b 7e--726f8244b14c d2--114a6fb51583 96-17112f1fad467 22--b-244fa7e-f- de--54fb5f2e9a4b 7e--726f8244b14c d2--114a6bb11583 96-171-2f1fad467 26--9-644fa7e-f2 de--54fb5f2e9a6b 7e--726f884-b14c d2--114a6bb11583 96-171-2f1fad467 22--9-644fa7e-f- da--5-fb5f2e9a6b fe--726f8244b14c d2--114a6ab11583
2	---4-8-----4-----1----- ---4-8-----4----- -----1-----8----- -----
3	---4-8-----4----- -----8----- -----4-8-----8----- ---4-----
4	---4-8-----8-----1-----8--1-----8-4----- -----1-8-----1--4-----4-----81----- -----1-8-----1-----1-----1-----1-----1-----

Table 5. A 4-round internal collision for Keccak with a 1600-bit permutation and a capacity of 640

 M_0 :

```
0000000082784B27 0000000027B97209 00000000F9E7B4C3 00000000FE890B5C 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

 M_1 :

```
1AE4DA9BCE0F58C5 3E41B66FEC61367F 60EFB06502B1E522 8F2689B944C6ADA4 3679B40E76AEE052
29023AF14A8D1931 0589A067B9C0882B 9CDCF37544841411 52448031E1488314 295FB9F654DD515D
58783A446CC0DF27 DC575C851C1DA5C0 9F82D47401FC7A76 7D7971B3C8B6D25A EA79DD2396CA4FEE
```

 M_2 :

```
0000408000000000 0080000000000000 0000000000100000 0000008000100000 0080400000000000
0000000000000000 0100040000000000 0000040000000000 8100000000000000 0000000000000000
0000000001080000 0000000000000000 0000000001000001 0000000000000001 0000000000000001
```

 M_0^* :

```
000000006DF2B918 00000000EF86D9FE 0000000040DD1D22 00000000326C57A3 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

 M_1^* :

```
93A3B74748B2D4D0 D1224F333B3E30CD E37E9B50203D12F1 9558EABAE983C68 036275C12894EBCD
E56F4097FFB56F5F 06070F676C145DFD FB11961465177857 C831E04FD29B424E 04AFAA83CF448D0B
59A7AC2AE2163340 E0E482684E961996 778732CDA01B329D BED51FB2554F233A 7830FB4DC95AB3C4
```

 M_2^* :

```
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

State after processing third block:

```
8E485EC7CC0271CA BA770EFBBD69EE16 5A3DA8FFD2F4C521 081E39496F095437 756E97B6477B1ED9
833FB0900600EB96 26A93661FE6F9531 86ADA9C976EB9861 9DD0D44634EC35AC F0D14E73C1916C96
3C0FF867406BB4EA 8EDF8F16DABCB AE9 DE3EDE57965FEE6F B34B3B20F466A277 7726B4B7AA8A84D8
272A11E9F2AD4981 046F4AF7DA9F98EC 4788C486729AC3A7 F95AFA8787C36990 06E3748CA8574FDC
929C857723322ED0 6706560C7EE7A3E3 313BB48B67DCDB2 795A30724698D71C 3BC9CFF2827373AC
```

B Results for Keccak with a 800-Bit Permutation

Table 6. The 4-round characteristic used for the second block to find a internal collision for Keccak with a 800-bit permutation and a capacity of 352. Note that a random first block was used in this case.

Round	State
0	551c2e5a 5b7cc9a2 d7fab224 893-5cd9 f3a536f5 7198b13f c8fb3e45 c82abe5e e85886f1 226465c- -a9f5-d6 d5b9-fb6 47926282 2538236- 996272ee 16f3b671 2fa-3-dd 3aad9-1e 6252-cfd 777383b4 7f928adf c7dcafa85 d8b21bf2 5bf4c55- 27dfd4dd
1	9-25244a 4-721523 612e18b- 8-ae-689 b-e28e-c b-242442 5-731563 61ae18b- 81ae-688 b-e28f-c b-27244a 5-721563 612a18b- 81ae4688 b-e28e-c b-25244a 5-7295e3 6-2e189- 81ae-688 3-c28e-c b-25244a 5-721563 e1ae18b- 91ae8688 b-e2-e-c
2	2----- 1----- ----- ----- 1----- ----- --1----- 2----- -----8----- ----- -----8- --1----- -----
3	2----- ----1-- ----- 1----- ----- ----1-- ----- 1----- 2----- -----1 1----- -----
4	2----8-- --1--8-- ----8-- 2----- 2-1----- -1----1 -----1----- --2--24- ----4- --2----- --2----- ----- -----

Table 7. A 4-round internal collision for Keccak with a 800-bit permutation and a capacity of 352

 M_0 :

```

8F1075E0 EDFC1488 58EFE9FC 433877BD 00000000
00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

```

 M_1 :

```

B051BED6 1DEC80DA D26B2923 01734BC7 2D2002D8
120AB268 10634585 16F789D1 4AD2F036 AF13E319
3DD3C552 0FF14835 8049189A 9786F56E

```

 M_2 :

```

20000800 00100800 00000800 20000000 20100000
01000001 00000000 00000001 00000000 00000000
00200240 00000040 00200000 00000200

```

 M_0^* :

```

3BBBA6F0 97006B85 09124595 C63FA0D6 00000000
00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

```

 M_1^* :

```

1B9837F7 1FFE60AD 13269184 9F567790 6C0191B1
DF45607B 97EE79C4 D963BDDA 00FF2D4A BE6F08C8
1FFBDA2C B787E7C8 34F8358A 8EB37499

```

 M_2^* :

```

00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

```

State after processing third block:

```

85AF483E 0CED18F7 BC101BB2 2F2CC963 0DCE54BE
30C5B7F8 B5CE439A 465B540D 760424F3 006BB414
045BBB7A 7C14CDA7 F082AF8E 2BA59219 A730ACCD
D5DBAE77 E1598F25 89373578 552A4091 E7D9C411
043CF740 A1D66CA3 F454A015 0E2A1D74 5FA83840

```

Recovering Private Keys Generated with Weak PRNGs

Pierre-Alain Fouque¹, Mehdi Tibouchi², and Jean-Christophe Zavalowicz³

¹ Université de Rennes 1 and Institut universitaire de France

`pierre-alain.fouque@ens.fr`

² NTT Secure Platform Laboratories

`tibouchi.mehdi@lab.ntt.co.jp`

³ Inria

`jean-christophe.zavalowicz@inria.fr`

Abstract. Suppose that the private key of discrete logarithm-based or factoring-based public-key primitive is obtained by concatenating the outputs of a linear congruential generator. How seriously is the scheme weakened as a result?

While linear congruential generators are cryptographically very weak “pseudorandom” number generators, the answer to that question is not immediately obvious, since an adversary in such a setting does not get to examine the outputs of the congruential generator directly, but can only obtain an implicit hint about them—namely the public key.

In this paper, we take a closer look at that problem, and show that, in most cases, an attack does exist to retrieve the key much faster than with a naive exhaustive search on the seed of the generator.

The problem is similar to the one considered by Bellare, Goldwasser and Micciancio regarding DSA and “pseudorandomness”, and this line of work arguably has renewed relevance in view of the sensitive role that random number generation has been found to play in a number of recent noted papers, such as the one by Lenstra et al. at CRYPTO 2012.

Keywords: linear congruential generator, discrete logarithm, factoring, cryptanalysis.

1 Introduction

Recently Lenstra et al. have proposed a *sanity check* of public keys collected on the web [20] and concluded that *generating secure public keys in the real world is challenging*. A related study by Heninger et al. [17] pointed in particular to the role of (pseudo)randomness generation as the chief cause for the weak keys observed in the wild. Cryptographers have to look at the security of pseudorandom generators used to generate the secret keys.

Pseudorandom generators are one of the main component of security products and their importance for security is hard to overestimate. A number of practical attacks stem from problems with randomness generators. Indeed, it is difficult to obtain randomness on computers and embedded devices, which tend to aggregate

various more or less reliable sources of entropy (mouse movements, passwords, network interrupts, electronic noise) and use pseudorandom number generators to expand them into arbitrarily long, hopefully uniform-looking bit strings.

Such practical generators have been analyzed and a framework has been given by Barak and Halevi [1] for the Linux generators, who discuss the importance of the randomness collector which maintains a state of enough entropy and an extraction function whose aim is to output random bitstring from the state, which is then expanded into an arbitrarily long pseudorandom string. Cryptographers tend to concentrate on the second aspect: they assume that a pseudorandom generator is seeded with a uniformly distributed bitstring and the goal of the generator is to stretch the seed to a longer bitstring. They define security notions and a generator is called secure if it is hard to distinguish its output from uniformly distributed bitstring given the previous output bits.

The security of the first cryptographically secure generators has been based on some number-theoretic hard problems, in the sense that the problem of distinguishing the outputs from uniform is reduced to a number-theoretic problem. In 1981, Shamir proposed one generator based on the strong RSA problem in [25]; Blum and Micali proposed a generator based on the discrete logarithm problem [5]; Blum, Blum and Shub proposed a generator based on the factorization problem in [4]; and Micali and Schnorr defined another generator based on the problem of distinguishing small e -th root modulo a RSA modulus from uniform values in [23].

These generators are interesting from a theoretical point of view, but they are rather inefficient and in practice more efficient generators using symmetric cryptographic functions have been preferred. For instance, ISO and NIST propose generators using symmetric primitives such as hash functions and block ciphers, which are more efficient and can be modeled as random oracles, ideal ciphers, pseudorandom functions or permutations. In [14], Desai et al. proved the security of these symmetric primitive-based generators, provided that the underlying primitives satisfy certain standard assumptions.

Linear Congruential Generators. Cryptographers have also studied the security of the linear congruential generators (LCG), widely used in simulation. These generators are efficient and have a very small memory footprint. Moreover, with suitable parameters, they can have good statistical properties such as a large period length and their output distribution is uniform. As they are also very easy to implement, they tend to be used in the standard libraries of many languages and compilers: the `rand` function proposed in the POSIX standard and the ones used in many C/C++ standard libraries, Java's `java.util.Random`, most implementations of `RAND` in Fortran, etc. Their efficiency and ubiquity make them attractive to implementors, especially in constrained environments, even in some security-sensitive applications.

Unfortunately, LCGs are cryptographically insecure: Boyar [9] proved that, with a sufficiently long run of the pseudorandom sequence, one can recover the seed in polynomial time in the size of the internal state and Stern [26] proved

that this is also the case even if only the most significant bits of each successive states is revealed (see also [8,19,16]). These attacks are based on lattice reduction, usually LLL [21]. However, Contini and Shparlinski have proposed a careful analysis of these algorithms in [12] and established some limits to their applicability, indicating that with properly chosen parameters, the generator might become secure.

Related Work. In any case, these attacks assume that the adversary has direct access to a certain number of outputs (although the parameters of the generator may remain unknown). As a result, using such a cryptographically weak generator in a cryptographic protocol does not automatically make the resulting protocol insecure, because an adversary against the protocol may not have access to the actual outputs of the generator. This led Bellare et al. [2] to analyze the security of the Digital Signature Algorithm (DSA) when the random nonces used in signature generation are computed using a linear congruential generator. They showed that this does seriously break security: a few signatures are enough to recover the secret key. This started an important line of research on the security of DSA when partial information on the nonces is revealed. For instance, Smart and Howgrave-Graham [18] and later Nguyen and Shparlinski [24] showed that the knowledge of a small number of the most significant bits of the nonces allows to efficiently recover the secret key using LLL. Bleichenbacher even established [3] that the bias on the single most significant bit of the nonce that occurs when using some version of the NIST generator can be sufficient to efficiently recover the secret key.

Our Contributions. In this paper, we investigate a question similar to the one considered by Bellare et al. but in a different direction than the DSA cryptanalysis papers. We consider the problem of the security of public-key schemes based on the hardness of the discrete logarithm problem or the factoring problem when the secret keys are constructed by concatenating the outputs of a linear congruential generator. Since the attacker does not get those outputs directly, but only an implicit hint, namely the corresponding public key, recovering the secret key is not trivial even if a cryptographically weak generator such as the LCG is used. We show that this is usually enough to recover the secret key much faster than using the trivial exhaustive search on the seed of the generator.

Our attack relies on the assumption that the secret key is obtained as the concatenation of successive outputs of a linear congruential generator. We also assume as is usual in cryptography that the parameters of the LCG are public and therefore an exhaustive search on the seed allows to recover the secret key in time 2^k where k is the size of the seed. Typical parameters for the LCG are 32 bits or 64 bits internal state to allow fast implementation without using a library for large integer arithmetic. The main observation of our work is that if we split the seed of size k into two parts $(A \cdot 2^{k/2} + B)$ where A and B are $(k/2)$ -bit long, the linearity of the LCG makes it “almost” possible to write the secret key as a sum $U + V \cdot 2^{k/2}$ where U (respectively V) only depends on B

(resp. A) and the parameters of the LCG. This is correct up to carries, which do occur but can be taken care of separately. As a result, we can obtain a time-memory tradeoff on the search for the LCG seed when such generators are used to generate the secret key of a discrete logarithm-based scheme or to the prime factors of an RSA modulus. The discrete logarithm case is mainly a baby-step giant-step attack on the lower and upper halves of the seed, while the factoring case proceeds similarly using multipoint polynomial evaluation.

The main advantage of these attacks is that they allow key recovery from the public key alone, independently of any further information on the underlying cryptographic schemes.

Organization of the Paper. The paper is organized as follows. After some preliminaries in §2, we present our attack in the discrete logarithm case in §3 and in the factoring case in §4. Finally, in §5, we give an overview of the complexity of our attacks for typical parameter sizes.

2 Preliminaries

We first recall the definition of the linear congruential generator and fix some notations which will be used in the following sections; then we briefly discuss multipoint evaluation of univariate polynomials, which will be used in the factoring case.

2.1 Linear Congruential Generator

For M an integer of size m bits, we denote by \mathbb{Z}_M the ring of integers modulo M . The internal state of a linear congruential generator evolves according to the following recurrence relation:

$$v_{i+1} = a \cdot v_i + b \pmod{M} \quad (1)$$

where a and b are fixed constant in \mathbb{Z}_M (the parameters of the generator) and $v_0 = s$ is the secret seed. The successive outputs o_i of the generator are the k least (or most) significant bits v_i at each iteration (for some fixed $k \in \{1, \dots, m\}$). Note that the recurrence equation is easily solved as:

$$v_i = a^i \cdot s + b \cdot (1 + a + \dots + a^{i-1}) = a^i \cdot s + b_i \pmod{M}. \quad (2)$$

The following attacks rely on the assumption that a certain secret x is computed as a concatenation of successive outputs of such a linear congruential generator, with known parameters a , b and M . In other words, x can be written as:

$$x = o_0 + 2^k o_1 + \dots + 2^{(r-1)k} o_{r-1} \quad (3)$$

for some fixed constant r . The secret x is then of size rk bits.

2.2 Multipoint Evaluation of Univariate Polynomials

Let $P(x) \in \mathbb{Z}_N[x]$, with N an arbitrary integer, be a polynomial of degree less than $d = 2^k$. The multipoint evaluation problem is the task of evaluating P at d distinct points $\alpha_0, \dots, \alpha_{d-1} \in \mathbb{Z}_N$. Using Horner's rule, it is easy to propose a solution that uses $O(d^2)$ additions and multiplications in \mathbb{Z}_N but it is well-known that one can propose an algorithm with quasi-linear complexity $\tilde{O}(d)$ operations in \mathbb{Z}_N using a divide-and-conquer approach [15]; a better, more involved algorithm based on the middle product of polynomials has later been proposed in some special cases by Bostan and Schost [7,6]. That observation has found several applications in cryptanalysis [11,13].

A succinct description of the classical approach, based on product and remainder trees of polynomials is given in Appendix A. The complexity $T(d)$ of the recursive algorithm satisfies $T(d) = 2T(d/2) + O(M(d))$, where $M(i)$ denotes the arithmetic complexity to compute the product of two polynomials of degree i in $\mathbb{Z}_N[x]$, and therefore $T(d) = O(M(d) \log d)$.

3 The Discrete Logarithm Case

We now consider key generation in a public-key scheme whose security is related to the discrete logarithm problem in some cyclic group \mathbb{G} of prime order q and generator g . Typically, for such a scheme, \mathbb{G} , q and g are public parameters, the secret key contains a random element $x \in \mathbb{Z}_q$, and $h = g^x$ is revealed as part of the public key.

Assume that x is obtained from the outputs of a linear congruential generator of known parameters, as in Equation (1). The problem is to recover x from the public data faster than by an exhaustive search on the seed s .

Our approach in a nutshell is as follows. Separate the seed in its lower-order and higher-order halves: $s = u + 2^{k/2} \cdot v$ (we can assume for simplicity's sake that k is even). Then, by Equation (1), the internal state of the generator can be written as:

$$v_i = (a^i \cdot u + 2^{k/2} \cdot a^i \cdot v + b_i) \bmod M.$$

Thus, the corresponding output o_i can essentially be written, in turn, as the sum of a part depending only on u and another part depending only on v —only “essentially” because of possible carry bits and of possible overflows in the addition modulo M , but this can be taken care of, and we will ignore that for the moment.

Then, x is itself of the form $x = U + V$ where U is a publicly computable function of u , and V of v . In the group \mathbb{G} , this gives $h = g^U \cdot g^V$, or equivalently:

$$g^U = h \cdot g^{-V}.$$

Now, in time and space $O(2^{k/2})$, we can find a collision between the lists of elements of \mathbb{G} of the form g^U for all $2^{k/2}$ possible values of u on the one hand, and $h \cdot g^{-V}$ for all $2^{k/2}$ possible values of v on the other hand, and hence recover the secret $x = U + V$.

The real algorithm has a slightly higher complexity due to the need to take carries and overflows into account, which we work out below first when $M = 2^k$ (the output is the full internal state) and then in the general case.

Note that since the parameters a and b are known, the constants b_i can be computed publicly and are thus irrelevant to the attack. To simplify notations, we can thus assume without loss of generality that $b = 0$.

Remark 1. The attack discussed here is generic and can of course be carried out in any cyclic group: it applies to (subgroups of) the multiplicative group of a finite field and to elliptic curves or abelian varieties alike. In the case of an elliptic curve group, the problem is to recover a secret value x from two points P, Q such that $Q = xP$, and when x is obtained from a linear congruential generator as before, it is again possible to divide x into two parts U and V , the first depending on u , the second on v . We can find a collision by checking an equality of the form $Q - UP = VP$.

3.1 Attack for Non-truncated Linear Congruential Generators

We first work out the details of this attack for a non-truncated linear congruential generator, which satisfies that $M = 2^k$. The non-truncated linear congruential generator is the most efficient of the linear congruential generator in the sense that it outputs the maximal number of available bits at each iteration.

Theorem 1. *Given two group elements $g, h \in \mathbb{G}$ with $h = g^x$, where x is an $(r \cdot k)$ -bit exponent generated with a non-truncated linear congruential generator with public parameters and k -bit state, there exists an algorithm which retrieves the secret x in time and memory $O(2^{\frac{k+r}{2}})$.*

Proof. As mentioned previously, we may assume without loss of generality that the LCG has parameters such that $b = 0$. By Equation (2), its successive outputs are thus of the form:

$$o_i = v_i = (a^i \cdot s) \bmod M = (a^i \cdot s) \bmod 2^k.$$

Now write the seed as $s = u + 2^{k/2} \cdot v$, with u, v of $k/2$ bits. We get:

$$o_i = (a^i \cdot (u + 2^{k/2} \cdot v)) \bmod 2^k.$$

We can expand that expression for o_i using the following elementary lemma.

Lemma 1. *For all $\alpha, \beta, \gamma \in \mathbb{Z}$, $\gamma \neq 0$, there exists $\varepsilon \in \{0, 1\}$ such that:*

$$(\alpha + \beta) \bmod \gamma = (\alpha \bmod \gamma) + (\beta \bmod \gamma) - \varepsilon\gamma.$$

Proof. Indeed, let $L = (\alpha + \beta) \bmod \gamma$ and $R = (\alpha \bmod \gamma) + (\beta \bmod \gamma)$. Clearly, L and R are congruent modulo γ , so they must differ by a multiple of γ . Moreover, $0 \leq L < \gamma$ and $0 \leq R < 2\gamma$, hence $-\gamma < R - L < 2\gamma$, so $R - L$ must be of the form $\varepsilon \cdot \gamma$ with $\varepsilon \in \{0, 1\}$ as required. \square

Thus, for all indexes i (and any choice of the two seed halves u, v), there exists $\varepsilon_i \in \{0, 1\}$ such that:

$$\begin{aligned} o_i &= (a^i \cdot u \bmod 2^k) + (a^i \cdot 2^{k/2}v \bmod 2^k) - \varepsilon_i \cdot 2^k \\ &= (a^i \cdot u) \bmod 2^k + 2^{k/2}(a^i \cdot v \bmod 2^{k/2}) - \varepsilon_i \cdot 2^k. \end{aligned}$$

If u and v are the two halves of the *correct* seed used to generate x , summing the $2^{ik}o_i$ yields, according to Equation (3):

$$x = U + V - Y$$

where:

$$\begin{aligned} U &= \sum_{i=0}^{r-1} 2^{ik} \cdot (a^i u \bmod 2^k) \\ V &= \sum_{i=0}^{r-1} 2^{ik+k/2} \cdot (a^i v \bmod 2^{k/2}) \\ Y &= \sum_{i=0}^{r-1} 2^{(i+1)k} \cdot \varepsilon_i. \end{aligned}$$

We can also decompose Y into a sum $W + Z$ where each of W and Z consist of $r/2$ terms, and obtain the relation $U - Z = x + W - V$, or equivalently:

$$g^{U-Z} = h \cdot g^{W-V}. \tag{4}$$

We can thus recover x by finding a collision between two lists of $2^{\frac{k+r}{2}}$ elements of \mathbb{G} , namely the g^{U-Z} (for all values of the half-seed u and all possible choices of the bits ε_i in Z) on the one hand, and the $h \cdot g^{W-Z}$ (for all values of the half-seed v and all possible choices of the bits ε_i in W) on the other. Using hash tables, this can be achieved in time and space $O(2^{\frac{k+r}{2}})$.

More precisely, one first computes the $2^{k/2}$ possible values U_i , the $2^{r/2}$ possible values Z_j and stores i, j in a hash table under the key $g^{U_i-Z_j}$. This table contains $2^{\frac{k+r}{2}}$ different values accessible in constant time. Then one computes the $2^{k/2}$ possible values V_s , the $2^{r/2}$ possible values W_t and tests, for each of them, whether $h \cdot g^{W_t-V_s}$ is a key of the hash table. When the test succeeds, one obtains the correct values of u and v and can deduce the value x . The attack is summarized in Algorithm 1. □

3.2 Attack for Truncated Linear Congruential Generators

We now consider a truncated linear congruential generator with a modulus $M > 2^k$ of size m (with $m < rk$). It is typically less efficient than the non-truncated one, but may have better properties in statistical and security terms. The attack we obtain has a slightly worse complexity than in the non-truncated setting.

Algorithm 1. Attack overview

Require: $q, g, h = g^x, a, b, M$

Ensure: x such as $h = g^x$

Compute the hash table H by storing i, j at $H(g^{U_i - Z_j})$

for each (V_s, W_t) **do**

if $H(h \cdot g^{W_t - V_s})$ exists **then**

return $x \leftarrow U_i + V_s - Z_j - W_t$

end if

end for

Theorem 2. *Given two group elements $g, h \in \mathbb{G}$ with $h = g^x$, where x is an $(r \cdot k)$ -bit exponent generated with a truncated linear congruential generator outputting the k most (or least) significant bits at each iteration, with public parameters and m -bit state, there exists an algorithm which retrieves the secret x in time and memory $O(2^{m/2} \cdot 5^{r/2})$.*

Proof. The principle of the attack remains similar however the carry is in a larger set of values. As a consequence the complexity in time and in memory is increased. Indeed, starting from Equation (2) with $b = 0$, the successive outputs are now of the form:

$$o_i = v_i \bmod 2^k = ((a^i \cdot s) \bmod M) \bmod 2^k$$

in the case where the least significant bits are output, and:

$$o_i = v_i \gg (m - k) = ((a^i \cdot s) \bmod M) \gg (m - k)$$

in the most significant bits case. By writing s as $u + 2^{m/2}v$, we get:

$$v_i = (a^i \cdot (u + 2^{m/2} \cdot v)) \bmod M.$$

and Lemma 1 ensures that:

$$(a^i \cdot u) \bmod M + 2^{m/2}(a^i \cdot v \bmod M) = v_i \text{ or } v_i + M.$$

Using the same lemma and the fact that $o_i = v_i \bmod 2^k$ (LSB case), we obtain:

$$(a^i \cdot u \bmod M) \bmod 2^k + (2^{m/2} \cdot a^i \cdot v \bmod M) \bmod 2^k = \begin{cases} o_i \\ o_i + 2^k \\ o_i + (M \bmod 2^k) \\ o_i + (M \bmod 2^k) + 2^k \\ o_i + (M \bmod 2^k) - 2^k \end{cases}$$

In the MSB case, we have (by denoting $j = m - k$) $o_i = v_i \gg j$ and thus:

$$(a^i \cdot u \bmod M) \gg j + (2^{m/2} \cdot a^i \cdot v \bmod M) \gg j = \begin{cases} o_i \\ o_i - 1 \\ o_i + (M \gg j) \\ o_i + (M \gg j) + 1 \\ o_i + (M \gg j) - 1 \end{cases}$$

Therefore, by applying the attack as before and using Equation (4), we have to find a collision between two sets of $2^{m/2} \cdot 5^{r/2}$, the factor $2^{m/2}$ coming from the search of U (respectively V) and the factor $5^{r/2}$ coming from the search of Z (respectively W). \square

4 The Factoring Case

The attacks extend to public-key schemes whose security is related to the hardness of factoring, or of the RSA problem.

Denoting p and q two secret primes obtained from outputs of a linear congruential generator, and N the resulting product published as part of the public key, we would like to find p and q given N and the parameters of the generator.

The idea is again to separate the seed into a lower-order and a higher-order part, and to obtain a time-memory tradeoff compared to exhaustive search. The key ingredient is multipoint polynomial evaluation.

4.1 Basic Prime Generation

We first consider a prime number generation algorithm (see [22]) which consists in, from a random seed, computing the required number of outputs, concatenating them and using a probabilistic primality test such as Miller-Rabin or a deterministic one such as the AKS primality test. If the test fails, one selects another random seed and restarts the algorithm: all primality tests are independent.

As before, we consider the case where the linear congruential generator is not truncated and the case where it is truncated.

Theorem 3. *Given a RSA modulus N with $N = pq$, where p is an $(r \cdot k)$ -bit prime generated with a non-truncated linear congruential generator (resp. a truncated linear congruential generator outputting the k most or least significant bits at each iteration), with public parameters and k -bit state (resp. m -bit state), there exists an algorithm which factorizes N in time and memory $\tilde{O}(2^{\frac{k+r}{2}})$ (resp. $\tilde{O}(2^{m/2} \cdot 5^{r/2})$) with overwhelming probability.*

Proof. For simplicity, we will treat the case of the non-truncated LCG since the use of a truncated one implies only a difference in the exhaustive search of the carry. By splitting the seed as $s = u + 2^{k/2} \cdot v$, we can write p as $p = U + V - Y$ with:

$$\begin{aligned} U &= \sum_{i=0}^{r-1} 2^{ik} (a^i u \bmod 2^k) \\ V &= \sum_{i=0}^{r-1} 2^{ik} (2^{k/2} (a^i v \bmod 2^{k/2})) \\ Y &= \sum_{i=0}^{r-1} 2^{(i+1)k} \cdot \varepsilon_i. \end{aligned}$$

We can also cut Y into two $r/2$ -bit elements W, Z . Let us denote $A = U - Z$ and $B = V - W$ and suppose having $c(A + B) \bmod N = cp \bmod N$ with c an integer. Then, except the rare case where c is a multiple of q , this value is necessarily a multiple of p . Indeed $cp \bmod pq$ can only have the values 0 (case where $q|c$), $p, \dots, (q-1)p$. Thus, a greater common divisor computation (GCD) with N will reveal p .

As an attacker, one does not have access to the correct value of the seed. Since u and v can take $2^{k/2}$ distinct values, one can compute the same amount of values U and V . Moreover there are $2^{r/2}$ possibilities for W and Z . In other words, the values $A = U - Z$ and $B = V - W$ are in two sets of $2^{\frac{k+r}{2}}$ elements and we have to find a test in order to determine the good ones.

More precisely, one first computes the $2^{\frac{k+r}{2}}$ different values $B_{s,t}$ by generating the values V_s and W_t and we consider the following polynomial of degree $2^{\frac{k+r}{2}}$:

$$P(X) = \prod_{s,t} (X + B_{s,t}) \bmod N$$

Then one computes the $2^{\frac{k+r}{2}}$ possible values $A_{i,j}$ by generating the values U_i and Z_j and proceeds a multi-evaluation of the polynomial P at the points $A_{i,j}$. The result is a set of $2^{\frac{k+r}{2}}$ values of the form:

$$\left\{ \prod_{s,t} (A_{i,j} + B_{s,t}) \bmod N \mid i = 0, \dots, 2^{k/2} - 1, j = 0, \dots, 2^{r/2} - 1 \right\}.$$

Finally one has to compute a test to detect the correct values A and B . It is done by computing a GCD between each value $(A_{i,j})$ and the public modulus N . Since all the values of the seed and all the values of the carry are efficiently tested, the prime p will be recovered except if $P(A)$ is equal to 0. However this failure case is extremely rare: it requires that at least one of the $d - 1$ integers composing the product with p is the prime q . The probability is thus equal to $\frac{d-1}{2^{\log q}}$. The attack is summarized in Algorithm 2. □

Algorithm 2. Attack overview (case $M = 2^k$)

Require: $N = pq, a, b, M$

Ensure: p such as $N = pq$

Generate the polynomial $P(X) = \prod_{s,t} (X + V_s - W_t) \bmod N$

Multi-evaluate P at the points $A_{i,j} = U_i - Z_j$

for each point $A_{i,j}$ **do**

if $\text{gcd}(P(A_{i,j}), N) \neq 1$ **then**

return $\text{gcd}(P(A_{i,j}), N)$

end if

end for

4.2 Improved Prime Generation

Since there is no link between each probabilistic primality test in the first prime number generating algorithm, the failed tests are useless and free of cost from the point of view of the attacker. We now propose another one with a link by using a counter.

PRIMEINC Method. The PRIMEINC algorithm is a prime number generating algorithm proposed by Brandt and Damgård in [10] which basically picks a random number and increases it until a prime is found. In other words, if p is not a prime (but odd), then $p = p + 2$ and repeat. According to the prime number theorem, we expect to find a prime number after $\log p$ trials on average.

Corollary 1. *Considering the two cases of Theorem 3 coupled with the PRIMEINC algorithm, there exists an algorithm which factorizes N in time and memory $\tilde{O}(2^{\frac{k+r}{2}})$ (resp. $\tilde{O}(2^{m/2} \cdot 5^{r/2})$) with overwhelming probability.*

Proof. In our attack, we now search the correct value of p such as $p = A + B + 2\epsilon$ with $\epsilon \in \{0, \dots, \log p\}$ (see Remark 2 for the size of the set). Thus, after the multi-evaluation, our algorithm should have computed a set covering the entire space of search as follows:

$$\left\{ \prod_{s,t} (A_{i,j} + B_{s,t} + 2\gamma) \bmod N \mid i \leq 2^{k/2} - 1, j \leq 2^{r/2} - 1, \gamma \leq \log p \right\}.$$

An efficient way to compute the search of the correct value of γ (i.e. $\gamma = \epsilon$) consists in applying the same trick as before, i.e. writing γ as $\gamma = \gamma_{MSB} + \gamma_{LSB}$ by splitting the bits into two parts.

In other words, in the first part of the algorithm, one computes the different values $B_{s,t}$ and the $\sqrt{\log p}$ values of γ_{LSB} . In the second part, one focus on the different values $A_{i,j}$ and the $\sqrt{\log p}$ values of γ_{MSB} . Thus, after the multi-evaluation, the resulted set corresponds to $2^{\frac{k+r}{2}} \sqrt{\log p}$ values of the form (case $M = 2^k$):

$$\prod_{s,t,\gamma} (A_{i,j} + B_{s,t} + 2(\gamma_{MSB} + \gamma_{LSB})) \bmod N.$$

With overwhelming probability, the value containing p does not contain q too and the test using the greatest common divisor will reveal p .

The modification due to the PRIMEINC method thus increases the complexity in time and in memory by a factor of $\sqrt{\log p}$, which disappears in the \tilde{O} since $\sqrt{\log p} = O(\sqrt{rk}) = O(\frac{r+k}{2})$. \square

Remark 2. Brandt and Damgård prove in [10] that the failure is about equal to $e^{-2\ell}$ when applying $\ell \cdot \log p$ iterations of the PRIMEINC algorithm. In the proof, we put $\ell = 1$, leading to a success rate of 86%.

5 Complexity Estimates for Concrete Parameter Sizes

Table 1 below presents the time complexity of our attack in the discrete logarithm case for typical parameter LCG sizes, as found in implementation of the `random` functions of common compilers and standard libraries, namely a modulus equal to either 2^{32} or 2^{64} (so that modular addition and multiplication can be implemented as simple operations on standard size registers), and an output size equal to either the full modulus size or half of it (corresponding to the top or bottom half of the state). The complexities are to be compared with that of the trivial attack: exhaustive search on seed.

Table 1. Overview of our Attacks complexities in the discrete logarithm case. When the output size is smaller than the modulus, the first number corresponds to the LSB case, and the second one to the MSB case.

Secret size	Modulus	Output size	Attack complexity	
160	2^{32}	32	$2^{18.5}$	
160	2^{32}	16	$2^{23.9}$	$2^{27.7}$
160	2^{64}	64	$2^{33.5}$	
160	2^{64}	32	2^{36}	$2^{37.8}$
256	2^{32}	32	2^{20}	
256	2^{32}	16	$2^{28.7}$	$2^{34.6}$
256	2^{64}	64	2^{34}	
256	2^{64}	32	$2^{38.3}$	$2^{41.3}$
512	2^{32}	32	2^{24}	
512	2^{32}	16	$2^{41.4}$	$2^{53.2}$
512	2^{64}	64	2^{36}	
512	2^{64}	32	$2^{44.7}$	$2^{50.6}$
1024	2^{64}	64	2^{40}	
2048	2^{64}	64	2^{48}	

Remark 3. Note that the differences of complexity between a linear congruential generator which outputs the least significant bits or the most significant bits is due to the fact that there are only three possibilities for the value of $(a^i \cdot u \bmod M) \bmod 2^k + (2^{m/2} \cdot a^i \cdot v \bmod M) \bmod 2^k$. Indeed, taking $M = 2^{32}$ or $M = 2^{64}$ yields $M \bmod 2^k = 0$.

Remark 4. In a few cases, for 16-bit output size, the complexity is in fact worse than the exhaustive search on the seed. This happens in the truncated case only, when r (the number of LCG outputs used to construct the secret) is particularly large, namely when $5^{r/2}$ (MSB case), resp. $3^{r/2}$ (LSB case), is greater than $2^{m/2}$.

Remark 5. In the factoring case, the complexities are larger by a logarithmic factor, from the use of quasilinear multipoint polynomial evaluation.

6 Conclusion

In this paper, we present new key-recovery attacks on discrete logarithm and factoring-based public-key schemes whose private keys are generated by concatenating the outputs of a linear congruential generator. Even though the LCG itself is known to be a cryptographically weak pseudorandom generator, it is not *a priori* obvious that its use would make key generation insecure, as its outputs are never revealed in clear to an adversary. It turns out, however, that the implicit hint about those outputs provided by the public key is enough to recover the private key significantly faster than with an exhaustive search on the seed.

It is hoped that this attack can be generalized to other, less naive pseudorandom generators in further work. Moreover, even in the case of the LCG, it would be interesting to extend it to different scenarios, such as the generation of randomness used in padding functions for encryption and signatures, or to settings where LCG parameters are unknown to the attacker.

References

1. Barak, B., Halevi, S.: A model and architecture for pseudo-random generation with applications to `/dev/random`. In: ACM CCS, pp. 203–212 (2005)
2. Bellare, M., Goldwasser, S., Micciancio, D.: “Pseudo-random” number generation within cryptographic algorithms: The DSS case. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 277–291. Springer, Heidelberg (1997)
3. Bleichenbacher, D.: On the generation of one-time keys in DL signature schemes. Presentation at the IEEE P1363 Working Group Meeting (November 2000)
4. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM J. Comput. 15(2), 364–383 (1986)
5. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput. 13(4), 850–864 (1984)
6. Bostan, A., Schost, É.: On the complexities of multipoint evaluation and interpolation. Theor. Comput. Sci. 329(1-3), 223–235 (2004)
7. Bostan, A., Schost, É.: Polynomial evaluation and interpolation on special sets of points. J. Complexity 21(4), 420–446 (2005)
8. Boyar, J.: Inferring sequences produced by a linear congruential generator missing low-order bits. J. Cryptology 1(3), 177–184 (1989)
9. Boyar, J.: Inferring sequences produced by pseudo-random number generators. J. ACM 36(1), 129–141 (1989)
10. Brandt, J., Damgård, I.B.: On generation of probable primes by incremental search. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 358–370. Springer, Heidelberg (1993)
11. Chen, Y., Nguyen, P.Q.: Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 502–519. Springer, Heidelberg (2012)
12. Contini, S., Shparlinski, I.E.: On stern’s attack against secret truncated linear congruential generators. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 52–60. Springer, Heidelberg (2005)

13. Coron, J.-S., Joux, A., Mandal, A., Naccache, D., Tibouchi, M.: Cryptanalysis of the RSA subgroup assumption from TCC 2005. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 147–155. Springer, Heidelberg (2011)
14. Desai, A., Hevia, A., Yin, Y.L.: A practice-oriented treatment of pseudorandom number generators. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 368–383. Springer, Heidelberg (2002)
15. Fiduccia, C.M.: Polynomial evaluation via the division algorithm: The fast fourier transform revisited. In: STOC, pp. 88–93 (1972)
16. Frieze, A.M., Håstad, J., Kannan, R., Lagarias, J.C., Shamir, A.: Reconstructing truncated integer variables satisfying linear congruences. *SIAM J. Comput.* 17(2), 262–280 (1988)
17. Heninger, N., Durumeric, Z., Wustrow, E., Alex Halderman, J.: Mining your Ps and Qs: Detection of widespread weak keys in network devices. In: Kohno, T. (ed.) USENIX Security 2012 (2012)
18. Howgrave-Graham, N., Smart, N.P.: Lattice attacks on digital signature schemes. *Des. Codes Cryptography* 23(3), 283–290 (2001)
19. Joux, A., Stern, J.: Lattice reduction: A toolbox for the cryptanalyst. *J. Cryptology* 11(3), 161–185 (1998)
20. Lenstra, A.K., Hughes, J.P., Augier, M., Bos, J.W., Kleinjung, T., Wachter, C.: Public keys. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 626–642. Springer, Heidelberg (2012)
21. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. *Math. Ann.* 261(4), 515–534 (1982)
22. Menezes, A., Van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography* (1996)
23. Micali, S., Schnorr, C.-P.: Efficient, perfect polynomial random number generators. *J. Cryptology* 3(3), 157–172 (1991)
24. Nguyen, P.Q., Shparlinski, I.: The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology* 15(3), 151–176 (2002)
25. Shamir, A.: On the generation of cryptographically strong pseudo-random sequences. In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 544–550. Springer, Heidelberg (1981); U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04
26. Stern, J.: Secret linear congruential generators are not cryptographically secure. In: FOCS, pp. 421–426. IEEE Computer Society (1987)

A Multipoint Evaluation of Univariate Polynomials

Here is a succinct description of the classical approach, based on product and remainder trees of polynomials. Let $P_0 = \prod_{\ell=0}^{d/2-1} (x - \alpha_\ell)$ and $P_1 = \prod_{\ell=d/2}^{d-1} (x - \alpha_\ell)$ and let us define $R_0 = P \bmod P_0$ and $R_1 = P \bmod P_1$. We have $R_0(\alpha_i) = P(\alpha_i)$ for all $i \in \{0, \dots, d/2-1\}$ and $R_1(\alpha_i) = P(\alpha_i)$ for all $i \in \{d/2, \dots, d-1\}$ and this gives immediately a recursive algorithm (i.e. compute P_0, P_1, R_0, R_1 and reduce the problem to the multipoint evaluation of R_0 and R_1 of degree $d/2 = 2^{k-1}$).

Let $A_i(x) = (x - \alpha_i)$ for $i \in \{0, \dots, d-1\}$ and $P_{i,j} = A_{j2^i} A_{j2^i+1} \dots A_{j2^i+2^i-1}$ for $i \in \{0, \dots, k\}$ and $0 \leq j < 2^{k-i}$. We have $P_{0,j} = A_j$ and $P_{i+1,j} = P_{i,2j} P_{i,2j+1}$ so for $i \in \{0, \dots, k\}$ we can compute recursively all polynomials $P_{i,j}$ and $0 \leq j < 2^{k-i}$ in $2^{k-i-1} O(M(2^i)) = O(M(d))$ operations in \mathbb{Z}_N where $M(i)$ denotes the arithmetic complexity to compute the product of two polynomials of degree i in $\mathbb{Z}_N[x]$. Overall, the computation of all polynomials $P_{i,j}$ requires $O(M(d) \log d)$ operations in \mathbb{Z}_N .

The polynomials R_0 and R_1 can be computed using $O(M(d))$ operations in \mathbb{Z}_N (using a Newton inversion), hence the complexity $T(d)$ of the recursive algorithm satisfies $T(d) = 2T(d/2) + O(M(d))$ and therefore $T(d) = O(M(d) \log d)$.

A Leakage-Resilient Pairing-Based Variant of the Schnorr Signature Scheme

David Galindo¹ and Srinivas Vivek²

¹ CNRS, Loria, France

david.galindo-chacon@loria.fr

² University of Luxembourg, Luxembourg

srinivasvivek.venkatesh@uni.lu

Abstract. Leakage-resilient cryptography aims at capturing side-channel attacks within the provable security framework. Currently there exists a plethora of schemes with provably secure guarantees against a variety of side-channel attacks. However, meeting the strongest security levels (resilience against continual leakage attacks) under the weakest assumptions leads currently to costly schemes. Additionally, recent results show the impossibility to achieve the strongest leakage-resilient security levels for cryptosystems whose secret key is uniquely determined by its public key.

The above justifies the use of stronger assumptions to achieve simpler, more efficient schemes, since most deployed and practical cryptosystems satisfy the above-mentioned uniqueness of the secret key property. In particular, the Schnorr-based leakage-resilient digital signature schemes proposed up to now are built by gluing together ℓ -copies of the basic signature scheme, resulting in a public key that admits exponentially-many secret keys. Furthermore, the space needed to store the secret key material is proportional to the leakage tolerated by these schemes.

We aim at designing a leakage-resilient variant of the Schnorr signature scheme whose secret key's storage space is constant, independently of the amount of leakage that it can tolerate. We assume that at any given time only the parts of the memory in use leak (split-state/only computation leaks information model); we ease the problem of exhibiting a security reduction by relying on generic groups (generic bilinear group model). We proceed by first proposing a pairing analogue of the Schnorr signature scheme, that we next transform to include split signing key updates. We give a leakage-resilience lower bound in generic bilinear groups against continual leakage attacks for the new scheme.

Keywords: Digital signatures, generic group model, leakage-resilient cryptography, continual leakage, efficiency, min-entropy.

1 Introduction

Over the last 30 years the theory of cryptography has been robustly built. It started with the proposal of simple, elegant and sound definitions [19,13,20], it

was followed by plausibility results under the weakest assumptions, and currently culminating in the practical constructions used nowadays by the information security community [1].

Concurrently, the theory and practice of cryptanalysis has been no less successful. In particular, the exploitation of the physical nature of the devices where cryptographic primitives are run, pioneered from an academic perspective in [24,25,9], has rendered many of the beautiful and theoretically robust constructions broken. Typical examples of side channel information are the analysis of primitives' running-time, power consumption or electromagnetic radiation leak, to name just but the most well-known.

The area of provable security that provides security reductions even in the presence of secret key leakage is called *leakage-resilient cryptography* and it has been an increasingly active field in recent years. In this work we assume leakage to be *continual leakage*, i.e. the useful leakage data per signature invocation is bounded in length, but unbounded overall; and adhere to the *independent leakage/split-state model*, meaning that the computation can be divided into rounds, where each such round access independent parts of the memory that leak independently.

The continual split-state leakage model has been previously used in the works [15,31,23,16,18,26]. The first assumption is restrictive but overall reasonable; in practice many side-channel attacks only exploit a polylogarithmic amount of information. The second assumption allows us to divide the memory of a device, at every computing step, into two parts - an *active* and a *passive* part. The part of the memory being currently accessed by a computation is the active part, and only the active part leaks information at any given time. We stress that even if leakage is local with respect to each part of the memory, it still captures some global functions of the secret key, for instance any affine leakage function. We refer to the work by Dziembowski and Faust [14] and by Faust *et al.* [17] for a discussion on the significance and limitations of this leakage model.

In the last few years the interplay between provable security and side-channel attacks has experienced great progress, as the works [22,16,8,11,27,7] bear witness for the case of digital signatures. However, the schemes that do not use any idealized assumption (random oracle, generic groups), are much more involved than their non-leakage counterparts and depart significantly from the schemes in the standard cryptography tool-box. Interestingly, recent work by Wichs [38] seems to indicate that it might be impossible to achieve continual leakage-resilience for cryptosystems whose secret key is uniquely determined by its public key, unless we weaken the security model. Furthermore, existing strongly secure proposals are not yet efficient enough. A rough estimation of the efficiency of current leakage-resilient schemes is that they are a linear number of times in the security parameter slower than their non-leakage counterparts.

In this work we study a signature scheme secure against continual leakage in the split-state model that builds on the Schnorr signature scheme [33]. Notice that several works [21,3,16] have already built leakage-resilient signature schemes based on Schnorr. All of these works confirm the finding by Wichs: they are

built by gluing together several copies of the basic Schnorr signature scheme (a technique that was first used by Okamoto [30]), and thus given its public key there are exponentially many possible secret keys. The works [21,3] only allow a bounded leakage during the life-time of the protocol, although in their model every part of the memory is susceptible to leak (as opposed to the split-state model); the work [16] uses the split-state model and allows roughly $1/36$ leakage ratio at every signing step, but the number of signature queries is bounded in advance. Our goal is to provide a Schnorr-like signature scheme where the secret key material to be stored is constant at any time, since in the aforementioned works the secret keys' storage is proportional to the leakage ratio allowed. In particular we propose a scheme where the secret key is uniquely determined by its public key, the secret key consists of only two group elements at any given time and it is unforgeable even if the number of adversarial signature queries is not known in advance.

Our positive results are of course far from the ideal achievement, that is, to prove leakage-resilience of the original Schnorr scheme instantiated over any cryptographic group \mathbb{G} where the discrete logarithm problem is assumed to be hard. However, this is presently out of reach using standard techniques [38]. This is why we state our theorems with respect to a transposition of the modified Schnorr signature scheme to pairing groups, where the secret key is no longer $x \in \mathbb{Z}_p$ but $X = g^x \in \mathbb{G}$, where \mathbb{G} is the base pairing group with $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. This allows us to use an idealized model of computation called the *generic bilinear group* (GBG) model that will ease our analysis. We proceed by first showing that our transposition of the Schnorr signature scheme to pairing groups is existentially unforgeable [20] in the GBG model. This is achieved by showing that the security reduction in the generic group model [36] for elliptic-curve based Schnorr signatures recently given by Neven, Smart and Warinschi [29] can be translated to the GBG and allows to deal with data leakage. Secondly, we modify the pairing-based Schnorr scheme by multiplicatively sharing $X = X_1 \cdot X_2$, where $X_1, X_2 \in \mathbb{G}$, and by breaking the signing scheme into two phases, each one using the corresponding share X_1 or X_2 . Again, at each new signature invocation a fresh sharing (X'_1, X'_2) of X is computed. Our main theorem (Theorem 2) states that allowing λ bits of leakage at each phase of every round overall decreases the security of the scheme by a factor of at most $2^{2\lambda}$ in our leakage model.

The GBG model has been previously used for stating leakage-resilience properties by Kiltz and Pietrzak [23], and Galindo and Vivek [18]. Kiltz and Pietrzak propose a bilinear version of the ElGamal key encapsulation mechanism which enjoys provable leakage-resilience in the presence of continual leakage. Their scheme is very efficient, less than a handful of times slower than standard ElGamal. Galindo and Vivek propose a leakage-resilient existentially unforgeable signature scheme based on the Boneh-Boyen identity-based encryption scheme [4]. Their scheme enjoys efficiency and leakage-resilience properties similar to the scheme by Kiltz and Pietrzak. Our Schnorr-like scheme has efficiency comparable to that of Galindo and Vivek's scheme. Additionally it is so far the only Schnorr-based leakage-resilient scheme whose secret key is uniquely determined

by its public key (thus bypassing the impossibility result by Wichs [38] at the cost of the GBG assumption), and the secret key material storage is constant and independent of the leakage rate (two elements in the pairing base group \mathbb{G}).

Organization of the Paper. We start in Section 2 by recalling some basic facts and definitions. In Section 3, we introduce a bilinear variant of the Schnorr signature scheme and prove its security in the GBG model. In Section 4, we split the secret state of the bilinear Schnorr scheme and prove its leakage resilience under continual leakage in the GBG model. Finally, we conclude in Section 5 by summarizing the achievements and limitations of our methodology.

2 Definitions

In this section, we recollect some basic notions of security of signature schemes, bilinear groups, and the generic bilinear group model. We also describe the model of leakage we shall consider in this paper and formulate a definition of security of signature schemes in the presence of continual leakage. We adapt the leakage model specified in [23] to signature schemes, exactly as done in [18].

Let \mathbb{Z} denote the set of integers and \mathbb{Z}_p ($p > 0$) denote, depending upon the context, either the set of integers $\{0, 1, \dots, p - 1\}$ or the ring modulo p . We denote a random sampling of an element $a \in A$ from a set A , and also denote a (possibly probabilistic) output of an algorithm A , by $a \leftarrow A$. If we want to explicitly denote the randomness r used during the sampling/output, then we do so by $s \stackrel{r}{\leftarrow} S$. Unless otherwise mentioned or implicit from the context, any sampling is from an uniform distribution. The symbol “:=” is used to define a notation in an expression, as in $A := \mathbb{Z}$, or to explicitly indicate an output of a deterministic algorithm or a function.

A signature scheme $\Pi = (\text{KeyGen}, \text{Sign}, \text{Verify})$ consists of three probabilistic polynomial-time algorithms KeyGen , Sign , and Verify . Let κ denote the security parameter. $\text{KeyGen}(\kappa)$ on input κ produces a public- and secret-key pair (pk, sk) along with other public parameters \mathbb{PP} . The algorithm $\text{Sign}(sk, m)$ on input a secret key sk and a message $m \in M$, where M is the message space, outputs a signature σ . $\text{Verify}(pk, m, \sigma)$ on input a public key pk , a message $m \in M$ and a signature σ , outputs a bit $b = 1$ meaning *valid*, or $b = 0$ meaning *invalid*. We require the following *correctness* requirement to be satisfied by Π :

$$\Pr[\text{Verify}(pk, m, \text{Sign}(sk, m)) = 1 : (pk, sk) \leftarrow \text{KeyGen}(\kappa), m \in M] = 1.$$

The standard security notion for signature schemes is existential unforgeability under adaptive chosen-message attacks (EUF-CMA), and it is defined through the following experiment:

$$\begin{array}{l|l} \text{Sign-Forge}_{\Pi}(\mathcal{A}, \kappa) & \text{Sign-Oracle } \Omega_{sk}(m) \\ (pk, sk) \leftarrow \text{KeyGen}(\kappa) & w := w \cup m \\ w := \emptyset & \sigma \leftarrow \text{Sign}(sk, m) \\ (m, \sigma) \leftarrow \mathcal{A}^{\Omega_{sk}(\cdot)}(pk) & \text{Return } \sigma \\ \text{If } m \in w, \text{ then return } b := 0 & \\ b \leftarrow \text{Verify}(pk, m, \sigma) & \end{array}$$

Definition 1. [Existential Unforgeability] *A signature scheme Π is existentially unforgeable under adaptive chosen-message attacks, in short “secure”, if $\Pr [b = 1]$ is negligible in $\text{Sign-Forge}_{\Pi}(\mathcal{A}, \kappa)$ for any efficient adversary \mathcal{A} .*

2.1 Leakage Model

We split the secret state into two parts that reside in different parts of the memory, and structure any computation that involves access to the secret state into a sequence of steps. Any step accesses only one part of the secret state (*active* part) and the other part (*passive* part) is assumed not to leak in the current step of computation. For simplicity, we define a security notion for leakage-resilient signature schemes assuming that the signing process is carried out in two steps. We also refer to a single invocation of the signature generation algorithm as a *round*.

Let us consider the problem of achieving leakage resilience under continual leakage even when a significant fraction of the bits of the secret state are leaked per round. Then it is necessary that the secret state must be *stateful*, i.e. the secret state must be refreshed during every round [23]. Otherwise, after many rounds the entire secret state will be completely leaked.

Formally, a stateful signature scheme $\Pi^* = (\text{KeyGen}^*, \text{Sign}_1^*, \text{Sign}_2^*, \text{Verify}^*)$ consists of four probabilistic polynomial-time algorithms KeyGen^* , Sign_1^* , Sign_2^* and Verify^* . $\text{KeyGen}^*(\kappa)$ is same as the set-up phase KeyGen of Π except that instead of a “single” secret key sk , it outputs two initial secret states (S_0, S'_0) . From the point of view of an adversary, the signing algorithm Sign of Π and $(\text{Sign}_1^*, \text{Sign}_2^*)$ have the same functionality. First, Sign_1^* is executed and later Sign_2^* is executed. That is, the i^{th} round of the signing process is carried out as:

$$(S_i, w_i) \stackrel{r_i}{\leftarrow} \text{Sign}_1^*(S_{i-1}, m_i) ; (S'_i, \sigma_i) \stackrel{r'_i}{\leftarrow} \text{Sign}_2^*(S'_{i-1}, w_i). \tag{1}$$

In the above expression, r_i and r'_i are the randomness used by Sign_1^* and Sign_2^* , respectively. The parameter w_i is some state information passed onto Sign_2^* by Sign_1^* . The signature σ_i is generated for the message m_i , and the internal state is updated from (S_{i-1}, S'_{i-1}) to (S_i, S'_i) .

We model the leakage during signature generation by giving an adversary \mathcal{A} access to a leakage oracle $\Omega_{(S_{i-1}, S'_{i-1})}^{\text{leak}}(\cdot)$. This oracle, in addition to giving \mathcal{A} signatures for the messages of its choice, also allows \mathcal{A} to obtain leakage from the computation used to generate signatures. More precisely, let λ be a *leakage parameter*. During the i^{th} signing round, \mathcal{A} is allowed to specify two functions f_i and h_i , each of range $\{0, 1\}^\lambda$, that can be efficiently computed. The outputs of the leakage functions are

$$A_i = f_i(S_{i-1}, r_i) ; A'_i = h_i(S'_{i-1}, r'_i, w_i). \tag{2}$$

Since the value of m can be included in the description of f_i and h_i , hence it is not explicitly included as an input. Note that it also possible for \mathcal{A} to specify h_i after obtaining A_i . But for simplicity of the exposition, we only describe here

the case where f_i and h_i are specified along with the message m_i to the oracle. The security of the signature scheme Π^* in the presence of (continual) leakage is defined through the following experiment $\text{Sign-Forge-Leak}_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$. In the description below, $|f_i|$ refers to the length of the output of f_i .

$\text{Sign-Forge-Leak}_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$ $(pk, (S_0, S'_0)) \leftarrow \text{KeyGen}^*(\kappa)$ $i := 1, w := \emptyset$ $(m, \sigma) \leftarrow \mathcal{A}^{\Omega_{(S_{i-1}, S'_{i-1})}^{\text{leak}}(\cdot)}(pk)$ $\text{If } m \in w, \text{ then return } b := 0$ $b \leftarrow \text{Verify}^*(pk, m, \sigma)$	$\text{Sign-Leak-Oracle } \Omega_{(S_{i-1}, S'_{i-1})}^{\text{leak}}(m_i, f_i, h_i)$ $\text{If } f_i \neq \lambda \text{ or } h_i \neq \lambda, \text{ return } \perp$ $(S_i, w_i) \stackrel{r_i}{\leftarrow} \text{Sign}_1^*(S_{i-1}, m_i)$ $(S'_i, \sigma_i) \stackrel{r'_i}{\leftarrow} \text{Sign}_2^*(S'_{i-1}, w_i)$ $\Lambda_i := f_i(S_{i-1}, r_i)$ $\Lambda'_i := h_i(S'_{i-1}, r'_i, w_i)$ $i := i + 1$ $w := w \cup m_i$ $\text{Return } (\sigma_i, \Lambda_i, \Lambda'_i)$
---	--

Definition 2. [Existential Unforgeability with Leakage] *A signature scheme Π^* is existentially unforgeable under adaptive chosen-message attacks in the presence of (continual) leakage if $\Pr[b = 1]$ is negligible in the Experiment $\text{Sign-Forge-Leak}_{\Pi^*}(\mathcal{A}, \kappa, \lambda)$ for any efficient adversary \mathcal{A} .*

2.2 Bilinear Groups

Let $\text{BGen}(\kappa)$ be a probabilistic bilinear group generator that outputs $(\mathbb{G}, \mathbb{G}_T, p, e, g)$ such that:

1. $\mathbb{G} = \langle g \rangle$ and \mathbb{G}_T are (multiplicatively written) cyclic groups of prime order p with binary operations \cdot and \star , respectively. The size of p is κ bits.
2. $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map that is:
 - (a) bilinear: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
 - (b) non-degenerate: $e(g, g) \neq 1$.

Such a group \mathbb{G} is said to be a bilinear group if the above properties hold and the group operations in \mathbb{G} and \mathbb{G}_T , and the map e are efficiently computable. The group \mathbb{G} is called as *base group* and \mathbb{G}_T as *target group*.

2.3 Generic Bilinear Group Model

The generic bilinear group (GBG) model [6] is an extension of the generic group model [36]. The encodings of the elements of \mathbb{G} and \mathbb{G}_T are given by random bijective maps $\xi : \mathbb{Z}_p \rightarrow \Xi$ and $\xi_T : \mathbb{Z}_p \rightarrow \Xi_T$, respectively, where Ξ and Ξ_T are sets of bit-strings. The group operations in \mathbb{G} and \mathbb{G}_T , and evaluation of the bilinear map e are performed by three public oracles \mathcal{O} , \mathcal{O}_T and \mathcal{O}_e , respectively, defined as follows. For all $a, b \in \mathbb{Z}_p$

- $\mathcal{O}(\xi(a), \xi(b)) := \xi(a + b \bmod p)$
- $\mathcal{O}_T(\xi_T(a), \xi_T(b)) := \xi_T(a + b \bmod p)$

$$- \mathcal{O}_e(\xi(a), \xi(b)) := \xi_T(ab \bmod p)$$

We assume that the (fixed) generator g of \mathbb{G} satisfies $g = \xi(1)$, and also the (fixed) generator g_T of \mathbb{G}_T satisfies $g_T = e(g, g) = \xi_T(1)$. The encoding of g is provided to all users of the group oracles. The users can thus efficiently sample random elements in both \mathbb{G} and \mathbb{G}_T .

We further assume that $\Xi \cap \Xi_T = \emptyset$, $|\Xi| = |\Xi_T| = p$, and that the elements of Ξ and Ξ_T are efficiently recognizable. For instance, the encodings in Ξ can comprise of the binary representation of the set $\{0, 1, \dots, p - 1\}$, where every string begins with ‘0’ and all are of uniform length. The encodings in Ξ_T are similarly defined but instead begin with ‘1’. Since the encodings are efficiently recognizable, the queries to a group oracle with an invalid encoding can be detected and an error can be raised. For simplicity, we assume that the users’ queries to the oracles are all valid.

2.4 Min-entropy

Let X be a finite random variable with probability distribution \Pr . The *min-entropy* of X , denoted $\mathbf{H}_\infty(X)$, is defined as $\mathbf{H}_\infty(X) := -\log_2 \left(\max_x \Pr[X = x] \right)$. Min-entropy is a standard measure of the worst-case predictability of a random variable. Let Z be a random variable. The *average conditional min-entropy* of X given Z , denoted $\tilde{\mathbf{H}}_\infty(X | Z)$, is defined as $\tilde{\mathbf{H}}_\infty(X | Z) := -\log_2 \left(\mathbb{E}_{z \leftarrow Z} \left[\max_x \Pr[X = x | Z = z] \right] \right)$. Average conditional min-entropy is a measure of the worst-case predictability of a random variable given a correlated random variable.

Lemma 1. [[12]] *Let $f : X \rightarrow \{0, 1\}^{\lambda'}$ be a function on X . Then $\tilde{\mathbf{H}}_\infty(X | f(X)) \geq \mathbf{H}_\infty(X) - \lambda'$.*

The following result is a variant of the Schwartz-Zippel Lemma [34,39,18].

Lemma 2. [Schwartz-Zippel; min-entropy version] *Let $F \in \mathbb{Z}_p[X_1, \dots, X_n]$ be a non-zero polynomial of (total) degree at most d . Let P_i ($i = 1, \dots, n$) be probability distributions on \mathbb{Z}_p such that $\mathbf{H}_\infty(P_i) \geq \log p - \lambda'$, where $0 \leq \lambda' \leq \log p$. If $x_i \stackrel{P_i}{\leftarrow} \mathbb{Z}_p$ ($i = 1, \dots, n$) are independent, then $\Pr[F(x_1, \dots, x_n) = 0] \leq 2^{\lambda'} \frac{d}{p}$.*

Corollary 1. *If $\lambda' < \log p - \omega(\log \log p)$ in Lemma 2, then $\Pr[F(x_1, \dots, x_n) = 0]$ is negligible (in $\log p$).*

3 Basic Signature Scheme

We propose a bilinear variant of the Schnorr signature scheme [32,33].

Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be a hash function. The signature scheme $\Pi_{\text{Sc}} = (\text{KeyGen}_{\text{Sc}}, \text{Sign}_{\text{Sc}}, \text{Verify}_{\text{Sc}})$, defined on the message space $\{0, 1\}^*$, is as follows:

1. $\text{KeyGen}_{\text{Sc}}(\kappa)$: Compute $\text{PP} := (\mathbb{G}, \mathbb{G}_T, p, e, g) \leftarrow \text{BGen}(\kappa)$. Choose random $x \leftarrow \mathbb{Z}_p$. Set $X := g^x$, $g_T := e(g, g)$, and $X_T := e(g, X) = g_T^x$. The public key is $pk := (\text{PP}, X_T, \text{H})$ and the secret key is $sk := X$.
2. $\text{Sign}_{\text{Sc}}(sk, m)$: Choose a random $t \leftarrow \mathbb{Z}_p$. Set $\gamma := \text{H}(g_T^t || m)$, $Y := g^t \cdot X^\gamma$ and $\sigma := (Y, \gamma)$. Output the signature σ .
3. $\text{Verify}_{\text{Sc}}(pk, m, \sigma)$: Let $\sigma = (Y, \gamma) \in \mathbb{G} \times \mathbb{Z}_p$. Set $\rho := e(Y, g) \star (g_T^x)^{-\gamma}$. Output the bit $b = 1$ (*valid*) if $\text{H}(\rho || m) = \gamma$. Otherwise output $b = 0$ (*invalid*).

We now prove the security of the above scheme in the GBG model relative to two hardness assumptions about the hash function H that were introduced in [29], and which are recalled below. These two assumptions are *weaker* than collision-resistance [29]. We adapt the proof techniques of [29] to the bilinear setting.

Definition 3. [Random-Prefix (Second-) Preimage problem [29]] *The advantage of an adversary \mathcal{A} in solving the Random-Prefix Preimage (RPP) problem (respectively, Random-Prefix Second-Preimage (RPSP) problem) for a hash function $\text{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$, with prefix in a set of bit-strings D , is given by*

$$\begin{aligned} \text{Adv}_{\text{H}}^{\text{RPP}[D]}(\mathcal{A}) &= \Pr [\text{H}(R || m) = \gamma : \gamma \leftarrow \mathcal{A}, \text{random } R \leftarrow D, m \leftarrow \mathcal{A}(R)], \\ \text{Adv}_{\text{H}}^{\text{RPSP}[D]}(\mathcal{A}) &= \Pr [\text{H}(R || m) = \text{H}(R || m') : m \leftarrow \mathcal{A}, \text{random } R \leftarrow D, \\ &\quad m' \leftarrow \mathcal{A}(R), m' \neq m], \end{aligned}$$

where the probability is taken over R and the coins of \mathcal{A} . The RPP problem (respectively, RPSP problem) for H is said to be (t, ϵ) hard if no adversary \mathcal{A} with running time at most t has advantage greater than ϵ in solving it.

Theorem 1. *The signature scheme Π_{Sc} is EUF-CMA secure in the generic bilinear group model if the RPP $[\Xi_T]$ and RPSP $[\Xi_T]$ problems are hard for H .*

Proof. Let \mathcal{A} be a $(\Gamma$ -time, q -query) adversary that can break the security of Π_{Sc} . Hence \mathcal{A} can make totally at most q group oracle, pairing oracle and signing oracle queries, and runs in time at most Γ . Let $q_{\mathcal{O}}$ denote the total number of calls to the oracles \mathcal{O} , \mathcal{O}_T and \mathcal{O}_e , and q_{Ω} denote the number of calls to the signing oracle Ω_{Sc} . Thus $q_{\mathcal{O}} + q_{\Omega} \leq q$.

$\text{Pr}_{\mathcal{A}, \Pi_{\text{Sc}}}^{\text{forge}}$ denote the advantage of the adversary \mathcal{A} in computing a forgery against Π_{Sc} . Also let RPP $[\Xi_T]$ and RPSP $[\Xi_T]$ problems be $(\Gamma', \epsilon_{\text{RPP}})$ - and $(\Gamma', \epsilon_{\text{RPSP}})$ -hard for the hash function H . We show that

$$\text{Pr}_{\mathcal{A}, \Pi_{\text{Sc}}}^{\text{forge}} \leq 2q \cdot \epsilon_{\text{RPSP}} + 36q^2 \cdot \epsilon_{\text{RPP}} + \frac{15q^2}{p} + \frac{108q^3}{p}$$

for any $(\Gamma$ -time, q -query) adversary \mathcal{A} in the GBG model, where $\Gamma' \approx \Gamma$. More precisely, Γ' is the sum of Γ and the time required by simulator to maintain the environment.

The main idea is to use \mathcal{A} to construct an adversary \mathcal{B} that solves both the RPP $[\Xi_T]$ and the RPSP $[\Xi_T]$ problems for the hash function H . \mathcal{B} will simulate

EUF-CMA experiment for \mathcal{A} in the naive way, through the game \mathcal{G} that we later describe below. In the game, \mathcal{B} also simulates the generic bilinear group oracles in the usual way by maintaining lists of pairs of encodings and polynomials that represent the relation amongst group elements. Let \mathcal{C} be a challenger trying to prove the hardness of both the RPP $[\Xi_T]$ and the RPSP $[\Xi_T]$ problems for H against \mathcal{B} .

There are only two possibilities for \mathcal{A} to output a forgery:

1. \mathcal{A} uses a signature previously obtained to output a forgery (on a distinct message).
2. \mathcal{A} does *not* output a previously obtained signature as a forgery.

Note that in a forgery of type 1, the “random prefix” for the hash function input (during verification) is the same as that for the corresponding previously obtained signature. In this case \mathcal{B} will attempt to solve the RPSP $[\Xi_T]$ problem for H . There are two issues that \mathcal{B} needs to address in this case to solve the RPSP problem. First, it must correctly guess at the beginning of the simulation when \mathcal{A} outputs a forgery of type 1 (and accordingly inform \mathcal{C} that it attempts to solve the RPSP problem). Secondly, \mathcal{B} needs to guess a priori which one of the previous signatures will \mathcal{A} use for the forgery. Then during that step \mathcal{B} needs to forward the corresponding message to the (now RPSP) challenger \mathcal{C} to obtain a random prefix as part of its RPSP challenge. This random prefix will be used as the encoding of the corresponding element g_T^t during the above signing step. \mathcal{B} solves the RPSP problem by forwarding to its (now RPSP) challenger \mathcal{C} the forged message output by \mathcal{A} . Note that the probability that \mathcal{B} will succeed in both the guesses is at least $\frac{1}{2q}$.

In the case of forgery of type 2, \mathcal{B} will attempt to solve the RPP $[\Xi_T]$ problem for H . Again, \mathcal{B} must first guess correctly when this type of forgery occurs (and accordingly inform \mathcal{C} that it attempts to solve the RPP problem). Secondly, \mathcal{B} must commit to a value γ to obtain a random prefix $R \in \Xi_T$ as part of a RPP challenge. Eventually when \mathcal{A} outputs a forgery on a (distinct) message m , it must turn out that the encoding of the “corresponding g_T^t ” must be R and that $H(R||m) = \gamma$. The tricky question is how to commit to the value γ before seeing R and m ? We overcome this problem by assuming that \mathcal{A} executes the verification algorithm $\text{Verify}_{\mathcal{S}_C}(\cdot)$ before outputting its forgery (Y, γ) , as done in [29]. This is w.l.o.g. because for every adversary that does not verify its forgery, we can build an adversary that has the same advantage but verifies its attempted forgery. This step guarantees that the elements $Y \in \mathbb{G}$ and $g_T^t \in \mathbb{G}_T$ appears as outputs of group oracles, with Y appearing before g_T^t . We bound the probability that Y appears later than g_T^t to be ϵ_{RPP} .

Hence \mathcal{B} simply needs to guess a priori which group oracle query outputs g_T^t . During this step, \mathcal{B} recovers the value of γ using the coefficients of the polynomials representing Y and g_T^t , as explained in (7) and proved in Lemma 3. Note that \mathcal{B} also needs to guess a priori which element will be output as Y . \mathcal{B} forwards the value of γ to the (now RPP) challenger \mathcal{C} and obtains the random prefix R , which it uses as the encoding of g_T^t . Note that the probability that \mathcal{B} will succeed in all the three guesses is at least $\frac{1}{2(3q)^2}$, where we later show

that the number of elements to choose from is at most $3q$ in both the cases. We would like to note that recovering γ in the proof of [29] (for the original Schnorr signature scheme) is easier than in the bilinear setting. This is because in [29] it involved guessing an element in only one list and the polynomials involved are all binomials.

We now formally describe the game \mathcal{G} . The description of the group oracles is typical for proofs in the generic group model (see [36,28,5,18]).

Description of Game \mathcal{G} : Initially, \mathcal{B} will choose a random bit $\beta_C \xleftarrow{\$} \{0, 1\}$. This bit decides which of the two problems RPSP (if $\beta_C = 0$) or RPP (if $\beta_C = 1$) will \mathcal{B} attempt to solve using the forgery output by \mathcal{A} . If $\beta_C = 0$, then \mathcal{B} randomly chooses $i^* \xleftarrow{\$} \{1, \dots, q\}$, else it randomly chooses $i^*, j^* \xleftarrow{\$} \{1, \dots, 3q\}$. The quantity i^* indicates the step in which \mathcal{B} interacts with \mathcal{C} to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$. This step may be a signature query (if $\beta_C = 0$) or a group oracle query to \mathcal{O}_T (if $\beta_C = 1$). More on this will be discussed later when describing the simulation of signature queries and queries to the group oracle \mathcal{O}_T .

Let $X, \{T_i : i \geq 1\}, \{U_i : i \geq 1\}$ and $\{V_i : i \geq 1\}$ be indeterminates, and $\{m_i : i \geq 1\}$ be bit-strings (messages) that are chosen by \mathcal{A} . Intuitively, these (or other) polynomials represent the relation amongst the group elements that are output by a group oracle, or guessed by \mathcal{A} . The indeterminate X corresponds to the quantity x (discrete logarithm of the secret key), whereas T_i corresponds to the parameter t_i chosen in the i^{th} signing step ($1 \leq i \leq q_\Omega$). Since \mathcal{A} can query the group oracles with representations (from Ξ and Ξ_T) not previously obtained from the group oracles, in order to accommodate this case, we introduce the indeterminates U_i, V_i . The U_i correspond to the guessed elements of \mathbb{G} , whereas V_i correspond to the guessed elements of \mathbb{G}_T . We denote the lists $\{T_i : i \geq 1\}, \{U_i : i \geq 1\}$ and $\{V_i : i \geq 1\}$ by $\{T\}, \{U\}$ and $\{V\}$, respectively.

\mathcal{B} maintains three lists of pairs

$$\mathcal{L} = \{(F_{1,i}, \xi_{1,i}) : 1 \leq i \leq \tau_1\}, \tag{3}$$

$$\mathcal{L}_T = \{(F_{T,i}, \xi_{T,i}) : 1 \leq i \leq \tau_T\}, \tag{4}$$

$$\mathcal{L}_\Omega = \{(m_i, \xi_{\Omega,i}, \gamma_i) : 1 \leq i \leq \tau_\Omega\}. \tag{5}$$

The entries $F_{1,i} \in \mathbb{Z}_p[X, \{U\}, \{T\}]$, $F_{T,i} \in \mathbb{Z}_p[X, \{U\}, \{V\}, \{T\}]$ are multivariate polynomials over \mathbb{Z}_p , whereas $\xi_{1,i}, \xi_{\Omega,i}$, and $\xi_{T,i}$ are bit-strings in the encoding sets Ξ (of \mathbb{G}), Ξ_Ω , and Ξ_T (of \mathbb{G}_T), respectively. We have $m_i \in \{0, 1\}^*$ and $\gamma_i \in \mathbb{Z}_p$. The polynomials in lists \mathcal{L} and \mathcal{L}_T correspond to elements of \mathbb{G} and \mathbb{G}_T , respectively, that \mathcal{A} will ever be able to compute or guess. The list \mathcal{L}_Ω records the signatures obtained by \mathcal{A} on the messages m_i of its choice. The values τ_1, τ_T and τ_Ω denote the respective list counters.

Initially, $\tau_1 = 1, \tau_T = 1, \tau_\Omega = 0, \mathcal{L} = \{(1, \xi_{1,1})\}, \mathcal{L}_T = \{(X, \xi_{T,1})\}$, and $\mathcal{L}_\Omega = \{\}$. The bit-strings $\xi_{1,1}, \xi_{T,1}$ are set to random distinct strings from Ξ and Ξ_T , respectively. We assume that there is some ordering among the strings in the sets Ξ and Ξ_T (say, lexicographic ordering), so that given a string $\xi_{1,i}$ or $\xi_{T,i}$, it is possible to efficiently determine its index in the lists, if it exists.

The initial state of the lists \mathcal{L} and \mathcal{L}_T correspond to the generator of \mathbb{G} and the public key, respectively.

The game begins by \mathcal{B} providing \mathcal{A} with the string $\xi_{1,1}$ from \mathcal{L} , and the string $\xi_{T,1}$ from \mathcal{L}_T .

Signature Query: Signature queries by \mathcal{A} are modeled as follows. \mathcal{A} provides a message $m_{\tau_\Omega} \in \{0, 1\}^*$ of its choice to \mathcal{B} . In response \mathcal{B} first increments the counters $\tau_1 := \tau_1 + 1$, $\tau_T := \tau_T + 1$ and $\tau_\Omega := \tau_\Omega + 1$, and sets $F_{T,\tau_T} := T_{\tau_\Omega}$.

- **(RPSP Challenge)** If $\beta_C = 0$ and $i^* = \tau_\Omega$, then \mathcal{B} passes on m_{τ_Ω} to \mathcal{C} to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$ as part of an RPSP challenge. If ξ_{T,i^*} is already present in \mathcal{L}_T , then \mathcal{B} completes the RPSP challenge with \mathcal{C} by returning arbitrary values, after \mathcal{A} terminates. Denote this event by **Abort**. Else \mathcal{B} sets $\xi_{T,\tau_T} := \xi_{T,i^*}$.
- Else if $\beta_C \neq 0$ or $i^* \neq \tau_\Omega$, then \mathcal{B} sets ξ_{T,τ_T} to a random string distinct from those already present in \mathcal{L}_T .

Append \mathcal{L}_T with $(F_{T,\tau_T}, \xi_{T,\tau_T})$. \mathcal{B} computes $\gamma_{\tau_\Omega} := H(\xi_{T,\tau_T} || m_{\tau_\Omega})$, sets $F_{1,\tau_1} := T_{\tau_\Omega} + \gamma_{\tau_\Omega} X$, sets ξ_{1,τ_1} to a random distinct string, appends \mathcal{L} with $(F_{1,\tau_1}, \xi_{1,\tau_1})$, sets $\xi_{\Omega,\tau_\Omega} := \xi_{1,\tau_1}$, and appends \mathcal{L}_Ω and provides \mathcal{A} with $(m_{\tau_\Omega}, \xi_{\Omega,\tau_\Omega}, \gamma_{\tau_\Omega})$.

Group Operation of \mathbb{G} : The calls made by \mathcal{A} to the group oracle \mathcal{O} are modeled as follows. For group operations in \mathbb{G} , \mathcal{A} provides \mathcal{B} with two operands (bit-strings) $\xi_{1,i}, \xi_{1,j}$ ($1 \leq i, j \leq \tau_1$) in \mathcal{L} and also specifies whether to multiply or divide them. \mathcal{B} answers the query by first incrementing the counter $\tau_1 := \tau_1 + 1$, and computes the polynomial $F_{1,\tau_1} := F_{1,i} \pm F_{1,j}$. If $F_{1,\tau_1} = F_{1,k}$ for some $k < \tau_1$, then \mathcal{B} sets $\xi_{1,\tau_1} := \xi_{1,k}$. Otherwise, ξ_{1,τ_1} is set to a random string distinct from those already present in \mathcal{L} . The pair $(F_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to \mathcal{L} and \mathcal{B} provides \mathcal{A} with ξ_{1,τ_1} . Note that the (total) degree of the polynomials $F_{1,i}$ in \mathcal{L} is at most one.

If \mathcal{A} queries \mathcal{O} with an encoding ξ not previously output by the oracle, then \mathcal{A} increments the counter $\tau_1 := \tau_1 + 1$, sets $\xi_{1,\tau_1} := \xi$, and sets $F_{1,\tau_1} := U_{\tau_1}$. The pair $(F_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to \mathcal{L} . This step is carried out for each guessed operand.

Group Operation of \mathbb{G}_T : The group oracle \mathcal{O}_T is modeled similar to \mathcal{O} , instead appropriately updating the counter τ_T , and appending the list \mathcal{L}_T with the output $(F_{T,\tau_T}, \xi_{T,\tau_T})$. \mathcal{B} provides \mathcal{A} with ξ_{T,τ_T} . For guessed operands in \mathbb{G}_T , a new variable T_{τ_T} is introduced instead.

Pairing Operation: For a pairing operation, \mathcal{A} queries \mathcal{B} with two operands $\xi_{1,i}, \xi_{1,j}$ ($1 \leq i, j \leq \tau_1$) in \mathcal{L} . \mathcal{B} first increments $\tau_T := \tau_T + 1$, and then computes the polynomial $F_{T,\tau_T} := F_{1,i} \cdot F_{1,j}$. Again, if $F_{T,\tau_T} = F_{T,k}$ for some $k < \tau_T$, then \mathcal{B} sets $\xi_{T,\tau_T} := \xi_{T,k}$. Otherwise, ξ_{T,τ_T} is set to a random string distinct from those already present in \mathcal{L}_T . The pair $(F_{T,\tau_T}, \xi_{T,\tau_T})$ is appended to \mathcal{L}_T , and \mathcal{B} provides \mathcal{A} with ξ_{T,τ_T} . Note that the degree of the polynomials $F_{T,i}$ in \mathcal{L}_T is at most two.

RPP Challenge: Recollect that \mathcal{B} has earlier sampled $i^*, j^* \stackrel{\$}{\leftarrow} \{1, \dots, 3q\}$. Since \mathcal{A} makes at most $q\mathcal{O} < q$ group oracle queries and that in each query \mathcal{A} can

guess at most two new elements, it is easy to see that lists \mathcal{L} and \mathcal{L}_T together have at most $3(q_{\mathcal{O}} + q_{\Omega}) \leq 3q$ elements. Hence

$$|\mathcal{L}| + |\mathcal{L}_T| \leq 3q. \quad (6)$$

If $\beta_C = 1$, then during each of the queries above the counter τ_T is checked while adding an element to the list \mathcal{L}_T . If $i^* = \tau_T$, then \mathcal{B} computes

$$\gamma^* = \sum_{i=1}^{q_{\Omega}} a_i \gamma_i - a_X, \quad (7)$$

where a_X is the coefficient of X in F_{T,i^*} , a_i is the coefficient of T_i in F_{1,j^*} ($1 \leq i \leq q_{\Omega}$), and γ_i is, as defined previously, the hash value in the i^{th} signature query.

If F_{1,j^*} does not exist, or $i^* > \tau_T$ at the end of the game \mathcal{G} , then \mathcal{B} completes the RPP challenge with \mathcal{C} by returning arbitrary values. Else, \mathcal{B} passes $\gamma^* \in \mathbb{Z}_p$ to \mathcal{C} to obtain a random prefix $\xi_{T,i^*} \in \Xi_T$, as part of an RPP challenge. If $F_{T,\tau_T} = F_{T,k}$ for some $k < \tau_T$ and $\xi_{T,i^*} \neq \xi_{T,k}$, then \mathcal{B} completes the RPP challenge with \mathcal{C} by returning arbitrary values (Abort). Else, if there is no such k but ξ_{T,i^*} is already present in \mathcal{L}_T , then also Abort. Else \mathcal{B} sets $\xi_{T,\tau_T} := \xi_{T,i^*}$.

If \mathcal{B} has made right guesses for i^* and j^* , then F_{1,j^*} and F_{T,i^*} corresponds to the forgery and satisfy

$$F_{T,i^*} := F_{1,j^*} - \gamma X, \quad (8)$$

where γ is the hash value corresponding to the forgery. Note again that both the polynomials exist (in case of successful forgery) because we assume that \mathcal{A} always verifies its attempted forgery before it is output. Lemma 3 below proves that indeed $\gamma^* = \gamma$. Because \mathcal{A} has access to the oracle \mathcal{O}_T , it is easy to see that it is not possible to recover γ from F_{T,i^*} alone.

Lemma 3. *Let $F_{T,i^*} = F_{1,j^*} - \gamma X$, as computed in (8). Let a_X be the coefficient of X in F_{T,i^*} , and a_i be the coefficient of T_i in F_{1,j^*} ($1 \leq i \leq q_{\Omega}$). Also let γ_i be the hash value in the i^{th} signature query. Then $\gamma = \sum_{i=1}^{q_{\Omega}} a_i \gamma_i - a_X$.*

Proof. Any polynomial in \mathcal{L} , in particular F_{1,j^*} , is of the form $F_{1,j^*} = c_1 + \sum_{i=1} c_{2,i} U_i + \sum_{i=1}^{q_{\Omega}} a_i (T_i + \gamma_i X)$, where $c_1, c_{2,i}, a_i \in \mathbb{Z}_p$ are chosen by \mathcal{A} . Hence the lemma follows. \square

End of Game \mathcal{G} : When \mathcal{A} terminates it outputs $(m, (\xi_{1,\alpha}, \gamma)) \in \{0, 1\}^* \times \Xi \times \mathbb{Z}_p$, where $\xi_{1,\alpha} \in \mathcal{L}$ and $1 \leq \alpha \leq \tau_1$. This corresponds to the “forgery” output by \mathcal{A} in the actual interaction. \mathcal{B} simply forwards m to its challenger \mathcal{C} .

Let **Forge** denote the event of successful forgery. Next, \mathcal{B} chooses random values $x, \{u\}, \{v\}, \{t\} \leftarrow \mathbb{Z}_p$ for the indeterminates $X, \{U\}, \{V\}, \{T\}$, respectively. Then it evaluates the polynomials in lists \mathcal{L} and \mathcal{L}_T . \mathcal{B} will abort if:

1. $F_{1,i}(x, \{u\}, \{t\}) = F_{1,j}(x, \{u\}, \{t\})$ in \mathbb{Z}_p , for any $F_{1,i} \neq F_{1,j}$ in \mathcal{L} .
2. $F_{T,i}(x, \{u\}, \{v\}, \{t\}) = F_{T,j}(x, \{u\}, \{v\}, \{t\})$ in \mathbb{Z}_p , for any $F_{T,i} \neq F_{T,j}$ in \mathcal{L}_T .

Let Collide denote either of the above events, i.e. a *collision* occurring in lists \mathcal{L} and/or \mathcal{L}_T . This completes the description of game \mathcal{G} and simulator \mathcal{B} .

Analysis of $\text{Pr}_{\mathcal{A}, \Pi_{\text{Sc}}}^{\text{forge}}$: The success probability $\text{Pr}_{\mathcal{A}, \Pi_{\text{Sc}}}^{\text{forge}}$ of \mathcal{A} in the actual EUF-CMA game satisfies

$$\text{Pr}_{\mathcal{A}, \Pi_{\text{Sc}}}^{\text{forge}} \leq \text{Pr}[\text{Forge} \mid \overline{\text{Collide}}] + \text{Pr}[\text{Collide}]. \tag{9}$$

This is because the event $\overline{\text{Collide}}$ ensures that \mathcal{A} will get to see only distinct group elements in the actual interaction. In other words, \mathcal{A} is unable to cause *collisions* among group elements. As long as the event Collide does not occur, then the view of \mathcal{A} is identical in the game \mathcal{G} and the actual interaction. Hence if \mathcal{A} is unable to provoke collisions, then adaptive strategies are no more powerful than non-adaptive ones (see [28, Lemma 2 on pp. 12], also [36]). This observation allows us to choose group elements and their representations independently of the strategy of \mathcal{A} .

First we bound $\text{Pr}[\text{Collide}]$. The τ_1 polynomials $F_{1,i}$ in \mathcal{L} have degree at most one. Note that $F_{1,i} \neq F_{1,j} \Leftrightarrow F_{1,i} - F_{1,j} \neq 0$ as polynomials. From Lemma 2 (with $\lambda' = 0$), the probability that two distinct polynomials in \mathcal{L} evaluate to the same value for randomly and independently chosen values for the indeterminates is at most $\frac{1}{p}$. Summing up over at most $\binom{\tau_1}{2}$ distinct pairs (i, j) , the probability that the condition 1 above holds is at most $\binom{\tau_1}{2} \cdot \frac{2}{p}$. Similarly, the probability that the condition 2 above holds is at most $\binom{\tau_T}{2} \cdot \frac{2}{p}$. Using (6) we obtain

$$\text{Pr}[\text{Collide}] \leq \binom{\tau_1}{2} \cdot \frac{1}{p} + \binom{\tau_T}{2} \cdot \frac{2}{p} \leq \frac{1}{p}(\tau_1 + \tau_T)^2 \leq \frac{9q^2}{p}. \tag{10}$$

Next we bound $\text{Pr}[\text{Forge} \mid \overline{\text{Collide}}]$ in terms of the advantage of \mathcal{B} against \mathcal{C} . Whenever \mathcal{A} succeeds in outputting a forgery $(m, (\xi_{1,\alpha}, \gamma))$, there are only two possibilities that can arise:

- **(Solving RPSP Challenge)** There exists an i ($1 \leq i \leq q_\Omega$) such that $(m_i, (\xi_{1,\alpha}, \gamma)) \in \mathcal{L}_\Omega$. In other words, \mathcal{A} uses a signature previously obtained to output its forgery on a distinct message. Let Forge_1 denote this event. If $\beta_C = 0$ and $i^* = i$, then \mathcal{B} can successfully use the forgery to solve the $\text{RPSP}[\Xi_T]$ problem for \mathcal{H} . This is because \mathcal{B} will attempt to solve the RPSP problem only when $\beta_C = 0$, the probability of which is $\frac{1}{2}$. Since at the beginning itself \mathcal{B} will decide at which signing step (step i^*) it will interact with \mathcal{C} when $\beta_C = 0$, the probability that $i^* = i$ is at least $\frac{1}{q_\Omega} > \frac{1}{q}$. Hence the advantage of \mathcal{B} in solving RPSP problem is at least $\frac{1}{2q} \text{Pr}[\text{Forge}_1 \mid \overline{\text{Collide}}] - \left(\frac{3q}{p}\right)$, where $\left(\frac{3q}{p}\right)$ is an upper bound on the probability that \mathcal{B} does not Abort due to a repeated entry in \mathcal{L}_T during RPSP challenge step. It may be noted that if \mathcal{B} attempts to solve the RPP problem using this type of forgery, then Abort will occur with overwhelming probability. Therefore, $\text{Pr}[\text{Forge}_1 \mid \overline{\text{Collide}}] \leq 2q \cdot \epsilon_{\text{RPSP}} + \frac{6q^2}{p}$.

- **(Solving RPP Challenge)** The complementary event of $\text{Forge}_1, \overline{\text{Forge}_1}$. That is, \mathcal{A} does not use a signature previously obtained to output its forgery. Since \mathcal{A} verifies its forgery before it is output, then there exists some i^{th} entry $(F_{T,i}, \xi_{T,i})$ in the list \mathcal{L}_T such that $H(\xi_{T,i}||m) = \gamma$. Also let this entry be the first occurrence of this pair in \mathcal{L}_T . If $\beta_C = 1$, $i^* = i$ and $j^* = \alpha$, then \mathcal{B} can successfully use the forgery to solve the RPP $[\Xi_T]$ problem for H . Hence the advantage of \mathcal{B} in solving the RPP problem is at least $\frac{1}{2(3q)^2} \Pr[\overline{\text{Forge}_1} | \overline{\text{Collide}}] - \left(\frac{3q}{p} + \epsilon_{\text{RPP}}\right)$, where again $\left(\frac{3q}{p}\right)$ is an upper bound on the probability that \mathcal{B} does not Abort due to a repeated entry in \mathcal{L}_T during RPP challenge step.

The quantity ϵ_{RPP} appearing above is an upper bound on the probability that the entry $(F_{T,i}, \xi_{T,i})$ does not appear before $(F_{1,\alpha}, \xi_{1,\alpha})$. Because $F_{T,i} = F_{1,\alpha} - \gamma X$ (c.f. (8)) and that encodings are random, this means that \mathcal{A} is able to compute the value γ even before getting an encoding $\xi_{T,i}$ such that $H(\xi_{T,i}||m) = \gamma$. In other words, \mathcal{A} has solved the RPP $[\Xi_T]$ problem for H . Therefore, $\Pr[\overline{\text{Forge}_1} | \overline{\text{Collide}}] \leq 36q^2 \cdot \epsilon_{\text{RPP}} + \frac{108q^3}{p}$.

Since $\Pr[\text{Forge} | \overline{\text{Collide}}] = \Pr[\text{Forge}_1 | \overline{\text{Collide}}] + \Pr[\overline{\text{Forge}_1} | \overline{\text{Collide}}]$, we obtain

$$\Pr[\text{Forge} | \overline{\text{Collide}}] \leq 2q \cdot \epsilon_{\text{RPSP}} + 36q^2 \cdot \epsilon_{\text{RPP}} + \frac{6q^2}{p} + \frac{108q^3}{p}. \quad (11)$$

From (9), (10) and (11), we have $\Pr_{\mathcal{A}, \Pi_{\text{Sc}}}^{\text{forge}} \leq 2q \cdot \epsilon_{\text{RPSP}} + 36q^2 \cdot \epsilon_{\text{RPP}} + \frac{15q^2}{p} + \frac{108q^3}{p}$. Hence if $q = \text{poly}(\log p)$, then $\Pr_{\mathcal{A}, \Pi_{\text{Sc}}}^{\text{forge}}$ is negligible provided $(\epsilon_{\text{RPSP}} + \epsilon_{\text{RPP}})$ is negligible. This completes the proof of Theorem 1. \square

4 A Leakage-Resilient Signature Scheme

In this section, we describe a leakage-resilient variant Π_{Sc}^* of the scheme Π_{Sc} . We use the techniques of [23] to transform Π_{Sc} to Π_{Sc}^* . A major difference between the two variants is that the secret key $X = g^x$ of Π_{Sc} is now split into two parts as $(S_0 := g^{l_0}, S'_0 := g^{x-l_0})$ for a random $l_0 \leftarrow \mathbb{Z}_p$. The two shares reside in different parts of the memory. The key generation step $\text{KeyGen}_{\text{Sc}}^*$ of Π_{Sc}^* is obtained by suitably modifying the $\text{KeyGen}_{\text{Sc}}$ step of Π_{Sc} . The signing step of Π_{Sc}^* is also split into two steps $\text{Sign}_{\text{Sc}1}^*$ and $\text{Sign}_{\text{Sc}2}^*$. After every signature query, the two shares of the secret key are randomly refreshed. This is required because, as seen in Section 2.1, if the secret state is not stateful, then the scheme cannot be secure in the presence of continual leakage.

Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be a hash function. The stateful signature scheme $\Pi_{\text{Sc}}^* = (\text{KeyGen}_{\text{Sc}}^*, \text{Sign}_{\text{Sc}1}^*, \text{Sign}_{\text{Sc}2}^*, \text{Verify}_{\text{Sc}}^*)$, defined on $\{0, 1\}^*$, is as follows:

1. $\text{KeyGen}_{\text{Sc}}^*(\kappa)$: Compute $\text{PP} := (\mathbb{G}, \mathbb{G}_T, p, e, g) \leftarrow \text{BGen}(\kappa)$. Choose random $x, l_0 \leftarrow \mathbb{Z}_p$. Set $X := g^x$ and $X_T := e(g, X) = e(g, g)^x$. The public key is $pk := (\text{PP}, X_T, H)$ and the secret key is $sk^* := (S_0 := g^{l_0}, S'_0 := g^{x-l_0} = X \cdot g^{-l_0}) \in \mathbb{G}^2$.

2. $\text{Sign}_{\mathbb{S}_{c1}}^*(S_{i-1}, m_i)$: Choose random $t_i, l_i \leftarrow \mathbb{Z}_p$. Set $S_i := S_{i-1} \cdot g^{l_i}$, $\gamma_i := H(g_T^{t_i} || m_i)$, and $Y_i' := g^{t_i} \cdot S_i^{\gamma_i}$.
3. $\text{Sign}_{\mathbb{S}_{c2}}^*(S'_{i-1}, (Y_i', \gamma_i, l_i))$: Set $S'_i := S'_{i-1} \cdot g^{-l_i}$, $Y_i := Y_i' \cdot (S'_i)^{\gamma_i}$, and $\sigma_i := (Y_i, \gamma_i)$. Output the signature σ_i .
4. $\text{Verify}_{\mathbb{S}_{c2}}^*(pk, m, \sigma)$: Let $\sigma = (Y, \gamma) \in \mathbb{G} \times \mathbb{Z}_p$. Set $\rho := e(Y, g) \star (g_T^x)^{-\gamma}$. Output the bit $b = 1$ (*valid*) if $H(\rho || m) = \gamma$. Otherwise output $b = 0$ (*invalid*).

The index i used above refers to the number of times the signing algorithm has been invoked. For $i \geq 1$, let $Z_i := \sum_{j=0}^i l_j$. The correctness property of $\Pi_{\mathbb{S}_{c2}}^*$ follows from $\Pi_{\mathbb{S}_{c1}}$ since $S_i \cdot S'_i = g^{Z_i} \cdot g^{x-Z_i} = X$. The leakage functions $f_i()$ and $h_i()$ that the adversary specifies to the signing oracle would take the form $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (Y_i', \gamma_i, l_i))$ (cf. (1) and (2)).

The signing step of $\Pi_{\mathbb{S}_{c2}}^*$ requires totally six exponentiations - four for $\text{Sign}_{\mathbb{S}_{c1}}^*$ and two for $\text{Sign}_{\mathbb{S}_{c2}}^*$. This quantity can be reduced to five if g^{l_i} is also passed on from $\text{Sign}_{\mathbb{S}_{c1}}^*$ to $\text{Sign}_{\mathbb{S}_{c2}}^*$. Note that the $\text{Sign}_{\mathbb{S}_{c2}}$ step of $\Pi_{\mathbb{S}_{c2}}$ requires only three exponentiations.

Since the input/output behaviour of $\Pi_{\mathbb{S}_{c2}}^*$ and $\Pi_{\mathbb{S}_{c1}}$ is identical, from Theorem 1 we obtain that $\Pi_{\mathbb{S}_{c2}}^*$ is secure in the GBG model in a non-leakage setting.

Lemma 4. *The signature scheme $\Pi_{\mathbb{S}_{c2}}^*$ is EUF-CMA secure in the generic bilinear group model if the RPP[\mathcal{E}_T] and RPSP[\mathcal{E}_T] problems are hard for \mathbb{H} .*

The following theorem shows that $\Pi_{\mathbb{S}_{c2}}^*$ is resilient to continual leakage in the GBG model if RPP[\mathcal{E}_T] and RPSP[\mathcal{E}_T] problems are hard for the hash function \mathbb{H} , and $\lambda < \frac{\log p}{2} - \omega(\log \log p)$, where λ is the leakage parameter.

Theorem 2. *The signature scheme $\Pi_{\mathbb{S}_{c2}}^*$ is secure with leakage w.r.t. Definition 2 in the generic bilinear group model relative to the hardness of RPP[\mathcal{E}_T] and RPSP[\mathcal{E}_T] problems for \mathbb{H} . Let the RPP and RPSP problems be $(\Gamma, \epsilon_{\text{RPP}})$ and $(\Gamma, \epsilon_{\text{RPSP}})$ -hard, respectively. Then the advantage of a $(\Gamma$ -time, q -query) adversary who gets at most λ bits of leakage per each invocation of $\text{Sign}_{\mathbb{S}_{c1}}^*$ or $\text{Sign}_{\mathbb{S}_{c2}}^*$ is $O\left(q^2 \epsilon_{\text{RPP}} + q \epsilon_{\text{RPSP}} + \frac{q^3}{p} + \frac{q^2}{p} 2^{2\lambda}\right)$.*

Let us briefly sketch the main ideas of the proof. Working on the lines of (9), the advantage of \mathcal{A} is bounded by its success probabilities conditioned on the event whether or not a collision has occurred in the lists consisting of elements of \mathbb{G} and \mathbb{G}_T . It is important to note that the proofs for the non-leakage setting (i.e. proof of Theorem 1) and the leakage setting would be the same conditioned on the fact that a collision has not occurred. The reason is that in the event of no collision, the adversary must either solve the RPP or the RPSP problem for the hash function in order to output a forgery (let us recall that a solution to either the RPP or the RPSP problem implies a collision for the hash function). Hence leakage on the secret state will not be useful in this case. Hence the success probability of \mathcal{A} against $\Pi_{\mathbb{S}_{c1}}$ and $\Pi_{\mathbb{S}_{c2}}^*$ is the same in the event of no collision (that includes the event of guessing the representations of group elements using partial information about them).

However the probability that a collision occurs in the leakage setting is increased by a factor of at most $2^{2\lambda}$. This is because when \mathcal{A} has access to leakage

output $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (Y'_i, \gamma_i, l_i))$ during i^{th} signature query, then in adversary's view the parameters t_i, l_i ($i \geq 1$) are no longer uniformly distributed even though they are still independent. Hence \mathcal{A} can now cause collisions among polynomials (in Conditions 1-2 on page 184) with increased probability. Each value t_i can only be leaked by f_i , hence at most λ bits of t_i can be leaked. Since l_i appears in both $f_i()$ and $h_i()$, at most 2λ bits of l_i can be leaked.

The only useful information that the leakage functions can provide to \mathcal{A} is about the secret key X and the values t_i . This is because the values l_i are independent of the signatures generated. However \mathcal{A} can use the leakages of l_i to eventually leak X . If \mathcal{A} is able to compute X , then it can trivially forge a signature on a distinct message. The event of no collision, and the fact that X is not a "linear combination" of the inputs to the leakage functions, guarantees that \mathcal{A} is unable to compute X .

Proof. Let \mathcal{A} be a (Γ -time, q -query) adversary that can break the security of Π_{Sc}^* . Hence \mathcal{A} can make totally at most q group oracle, pairing oracle and signing oracle queries, and runs in time at most Γ . In the count of q , even group oracle queries by leakage functions f_i, h_i ($i \geq 1$) specified by \mathcal{A} are also included. Let the adversary \mathcal{A} play the game \mathcal{G}' described below. This game is an extension of game \mathcal{G} described in the proof of Theorem 1. To avoid repetition, we only describe here the extensions that are not part of game \mathcal{G} . Let $\{L\}$ denote the list of indeterminates $\{L_i : 1 \leq i \leq q_\Omega\}$ that correspond to the values l_i in Π_{Sc}^* .

Game \mathcal{G}' : For each leakage function $f_i(S_{i-1}, (l_i, t_i))$ and $h_i(S'_{i-1}, (Y'_i, \gamma_i, l_i))$, \mathcal{A} maintains a pair of lists $(\mathcal{L}^{f_i}, \mathcal{L}_T^{f_i})$ and $(\mathcal{L}^{h_i}, \mathcal{L}_T^{h_i})$, respectively. These lists contain polynomial and bit-string pairs. The polynomials in \mathcal{L}^{f_i} and \mathcal{L}^{h_i} belong to $\mathbb{Z}_p[X, \{U\}, \{T\}, \{L\}]$, and the corresponding bit-strings are from the encoding set \mathcal{E} of group \mathbb{G} . The polynomials in $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ are in the ring $\mathbb{Z}_p[X, \{U\}, \{V\}, \{T\}, \{L\}]$, and the corresponding bit-strings are from the encoding set \mathcal{E}_T of group \mathbb{G}_T . Intuitively, the polynomials in lists \mathcal{L}^{f_i} and \mathcal{L}^{h_i} correspond to the elements of group \mathbb{G} that can be computed by f_i and h_i , respectively, whereas the lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ correspond to the elements of \mathbb{G}_T .

Every polynomial in \mathcal{L}^{f_i} is of the form $c_{1,i}L_i + c_{2,i} \sum_{j=0}^{i-1} L_j + c_{3,i}D_i$, where $c_{1,i}, c_{2,i}, c_{3,i} \in \mathbb{Z}_p$ are chosen by \mathcal{A} and $D_i \in \mathbb{Z}_p[X, \{U\}, \{T\}]$ is in \mathcal{L} (cf. (3)). Every polynomial in \mathcal{L}^{h_i} is of the form

$$d_{1,i}L_i + d_{2,i} \left(X - \sum_{j=0}^{i-1} L_j \right) + d_{3,i} \left(T_i + \gamma_i \left(\sum_{j=0}^i L_j \right) \right) + d_{4,i}W_i, \quad (12)$$

where $d_{1,i}, d_{2,i}, d_{3,i}, d_{4,i} \in \mathbb{Z}_p$ are also chosen by \mathcal{A} and $W_i \in \mathbb{Z}_p[X, X_0, X_1, \{U\}, \{T\}]$ is in the list \mathcal{L} . Note that the polynomials in lists \mathcal{L}^{f_i} and \mathcal{L}^{h_i} are of degree at most one, and that they do not contain the monomial X . The polynomials in lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ are of degree at most two.

The game \mathcal{G}' proceeds exactly as game \mathcal{G} except that \mathcal{A} can also obtain leakage through functions f_i and h_i in the i^{th} signature query. In particular, when \mathcal{A} terminates it outputs $(m, (\xi_{1,\alpha}, \gamma)) \in \{0, 1\}^* \times \Xi \times \mathbb{Z}_p$, where $\xi_{1,\alpha} \in \mathcal{L}$ and $1 \leq \alpha \leq \tau_1$. Let us denote by Forge^* the event of successful forgery by \mathcal{A} . Let Collide^* denote the event of a collision occurring in lists $\mathcal{L}, \mathcal{L}_T, \mathcal{L}^{f_i}, \mathcal{L}^{h_i}, \mathcal{L}_T^{f_i}, \mathcal{L}_T^{h_i}$ ($1 \leq i \leq q_\Omega$). The polynomials are now evaluated with values chosen from independent distributions with min-entropy $\log p - 2\lambda$, not necessarily from an uniform distribution. The exact distribution depends on the leakage functions chosen by \mathcal{A} . Since we are only interested to upper bound the collision probability, we can safely assume that the simulator chooses the right distribution. Note that even in the leakage setting, adaptive strategies are no more powerful than non-adaptive ones, as observed in [2, pp. 691]. This completes the description of the game \mathcal{G}' .

Let $\text{Pr}_{\mathcal{A}, \Pi_{\text{sc}}^*}^{\text{forge}}$ denote the advantage of \mathcal{A} in computing a forgery against Π_{sc}^* . On the lines of (9), we can write

$$\text{Pr}_{\mathcal{A}, \Pi_{\text{sc}}^*}^{\text{forge}} \leq \text{Pr}[\text{Forge}^* \mid \overline{\text{Collide}^*}] + \text{Pr}[\text{Collide}^*]. \tag{13}$$

As mentioned before, conditioned on the event $\overline{\text{Collide}^*}$, the view of the adversary \mathcal{A} will be same in both the games \mathcal{G}' and \mathcal{G} . This is because in both the cases \mathcal{A} will get to see only distinct group elements. Hence, from (11), we have

$$\text{Pr}[\text{Forge}^* \mid \overline{\text{Collide}^*}] \leq O\left(q^2 \epsilon_{\text{RPP}} + q \epsilon_{\text{RPSP}} + \frac{q^3}{p}\right). \tag{14}$$

Lemma 5. $\text{Pr}[\text{Collide}^*] \leq O\left(\frac{q^2}{p} 2^{2\lambda}\right)$.

Proof. To compute the required probability, the polynomials in lists $\mathcal{L}, \mathcal{L}_T, \mathcal{L}^{f_i}, \mathcal{L}^{h_i}, \mathcal{L}_T^{f_i}, \mathcal{L}_T^{h_i}$ ($1 \leq i \leq q_\Omega$) are evaluated by choosing values from \mathbb{Z}_p according to (independent) distributions with min-entropy at least $\log p - 2\lambda$. This is because \mathcal{A} can obtain at most 2λ bits of leakage about l_i ($i = 0, \dots, q_\Omega$), and at most λ bits of t_i ($i = 1, \dots, q_\Omega$). According to Lemma 1, the values l_i, t_i have min-entropy at least $\log p - 2\lambda$ in the view of \mathcal{A} . The total length of all the lists is at most $O(q_\Omega + q_\mathcal{O}) = O(q)$. Hence there can be at most $O(q^2)$ pairs of distinct polynomials (of degree at most two) evaluating to the same value. From Lemma 2 (with $\lambda' = 2\lambda$), we obtain $\text{Pr}[\text{Collide}^*] \leq O\left(\frac{q^2}{p} 2^{2\lambda}\right)$. \square

From (13), (14) and Lemma 5, we have $\text{Pr}_{\mathcal{A}, \Pi_{\text{sc}}^*}^{\text{forge}} \leq O\left(q^2 \epsilon_{\text{RPP}} + q \epsilon_{\text{RPSP}} + \frac{q^3}{p} + \frac{q^2}{p} 2^{2\lambda}\right)$. This completes the proof of Theorem 2. \square

5 Conclusions

In this work we presented the pairing-based split Schnorr scheme and quantified its security against independent and continual leakage in the generic bilinear

group model. In particular, we showed that allowing λ bits of leakage at each of the two phases of every round in the proposed scheme can be compared to decreasing the security of the pairing-based Schnorr scheme (without leakage) by a factor of at most $2^{2\lambda}$ in our leakage model.

Undoubtedly, the main advantage of our approach lies on its practicality: signing takes at most 5 exponentiations in \mathbb{G} plus 1 exponentiation in \mathbb{G}_T ; verification takes 1 pairing plus 1 exponentiation in \mathbb{G}_T . A suitable bilinear pairing group to implement our modification of the Schnorr scheme is the pairing-friendly curve BN-128 studied by Scott in [35]. Thus while our scheme offers continual leakage-resilience, its efficiency is comparable to that of standard pairing-based signature schemes [37]. This is currently out of reach for schemes that offer EUF-CMA security against continual leakage and dispense with the generic group model, be it in the standard or the random oracle models.

It is interesting to compare the relative efficiency and strength of our scheme and the FKPR scheme by Faust *et al.* [16]. The latter has a weak form of EUF-CMA security against continual independent leakages in the random oracle model, where the adversary can ask at most for D signatures queries, for D fixed before the key generation phase. The main advantage of that construction with respect to ours is that it can be implemented over any group \mathbf{G} where the DL problem is conjectured to be hard (our scheme needs pairing-based groups). Let us now examine its disadvantages against our scheme, which are all related to its practicality. The signer in the FKPR scheme needs to maintain a state consisting on roughly d Schnorr signatures and d public and corresponding secret keys, with the length of a signature being proportional to d and $D = 2^{d+1} - 2$; signing takes 9 exponentiations in the group \mathbf{G} , while verification time is proportional to d . FPKR only tolerates a leakage rate of roughly $1/36$. Thus, for reasonable values of d , e.g. $d = 20$, our scheme is more efficient in storage, computing time and leakage ratio than the FPKR scheme, while offering standard existential unforgeability against continual leakage in the split-state model. Finally both our scheme and the FPKR scheme use an idealized model of computation to prove security, namely the former uses the random oracle model, while ours uses generic groups.

Acknowledgements. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865.

References

1. ISO/IEC 18033-2:2006 - Information technology – security techniques – encryption algorithms – Part 2: Asymmetric ciphers
2. Aggarwal, D., Maurer, U.: The leakage-resilience limit of a computational problem is equal to its unpredictability entropy. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 686–701. Springer, Heidelberg (2011)

3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
4. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Boyen, X.: Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology* 21(2), 149–177 (2008)
6. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer [10], pp. 440–456
7. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011)
8. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: FOCS, pp. 501–510. IEEE Computer Society (2010)
9. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
10. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
11. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010)
12. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* 38(1), 97–139 (2008)
13. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. *SIAM J. Computing* 30(2), 391–437 (2000)
14. Dziembowski, S., Faust, S.: Leakage-resilient cryptography from the inner-product extractor. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 702–721. Springer, Heidelberg (2011)
15. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302. IEEE (2008)
16. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 343–360. Springer, Heidelberg (2010)
17. Faust, S., Pietrzak, K., Schipper, J.: Practical leakage-resilient symmetric cryptography. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 213–232. Springer, Heidelberg (2012)
18. Galindo, D., Vivek, S.: A practical leakage-resilient signature scheme in the generic group model. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 50–65. Springer, Heidelberg (2013)
19. Goldwasser, S., Micali, S.: Probabilistic encryption. *Journal of Computer and System Sciences* 28(2), 270–299 (1984)
20. 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)
21. Katz, J.: Signature schemes with bounded leakage resilience. *Cryptology ePrint Archive, Report 2009/220* (2009), <http://eprint.iacr.org/>

22. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
23. Kiltz, E., Pietrzak, K.: Leakage resilient elgamal encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 595–612. Springer, Heidelberg (2010)
24. 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)
25. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
26. Liu, F.-H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 517–532. Springer, Heidelberg (2012)
27. Malkin, T., Teranishi, I., Vahlis, Y., Yung, M.: Signatures resilient to continual leakage on memory and computation. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 89–106. Springer, Heidelberg (2011)
28. Maurer, U.M.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005)
29. Neven, G., Smart, N.P., Warinschi, B.: Hash function requirements for schnorr signatures. *J. Mathematical Cryptology* 3(1), 69–87 (2009)
30. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
31. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
32. 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)
33. Schnorr, C.-P.: Efficient signature generation by smart cards. *J. Cryptology* 4(3), 161–174 (1991)
34. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. *J. ACM* 27(4), 701–717 (1980)
35. Scott, M.: On the efficient implementation of pairing-based protocols. In: Chen, L. (ed.) IMACC 2011. LNCS, vol. 7089, pp. 296–308. Springer, Heidelberg (2011)
36. 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)
37. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer [10], pp. 114–127
38. Wichs, D.: Barriers in cryptography with weak, correlated and leaky sources. In: Kleinberg, R.D. (ed.) ITCS, pp. 111–126. ACM (2013)
39. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, K.W. (ed.) EUROCRYPT 1979 and ISSAC 1979. LNCS, vol. 72, pp. 216–226. Springer, Heidelberg (1979)

High-order Masking by Using Coding Theory and Its Application to AES

Guilhem Castagnos¹, Soline Renner^{1,2}, and Gilles Zémor¹

- ¹ Institut de Mathématiques de Bordeaux UMR 5251, Université Bordeaux 1
351, cours de la Libération, 33405 Talence cedex, France
{guilhem.castagnos,gilles.zemor}@math.u-bordeaux1.fr
- ² Oberthur Technologies Security Group
4, allée du Doyen Georges Brus, 33600 Pessac, France
s.renner@oberthur.com

Abstract. To guarantee that some implementation of a cryptographic scheme is secure against side channel analysis, one needs to formally prove its leakage resilience. A relatively recent trend is to apply methods pertaining to the field of Multi-Party Computation: in particular this means applying secret sharing techniques to design masking countermeasures. It is known besides that there is a strong connection between secret sharing schemes and error-correcting codes, namely every linear code gives rise to a linear secret sharing scheme. However, the schemes mostly used in practice are the so-called Boolean masking and Shamir's secret sharing scheme and it is widely thought that they are the most adapted to masking techniques because they correspond to MDS codes that are in some sense optimal. We propose alternative masking techniques that rely on *non-MDS* linear codes: these codes are non-binary but have an underlying binary structure which is that of a *self-orthogonal binary* code. Their being non-MDS is compensated by the fact that the distributed multiplication procedure is more efficient than with MDS codes due to an efficient encoding process and that the distributed computation of squares comes at almost no cost. In protecting AES against high-order side channel analysis, this approach is more efficient than methods using Shamir's secret sharing scheme and competitive with Boolean masking.

Keywords: High-Order Side Channel Analysis, Linear Secret Sharing Scheme, Self-Dual Codes.

1 Introduction

In the 90's, Kocher *et al.* published the so-called *Side Channel Analysis* (SCA for short) which generated a huge interest in both academic and industrial communities. Indeed, they noticed that side channel leakage of an embedded device such as its power consumption or its electromagnetic radiation can reveal information on any value manipulated ([Koc96, KJJ99]). When applied during the execution of a cryptographic algorithm, such an attack can be used for secret key recovery. Since then, a wide variety of attacks has been proposed

including t^{th} -order SCA which exploits leakage observations resulting from the handling of t intermediate variables during the cryptosystem processing (see *e.g.*, [Mes00, JPS05, PR10, FMPR10]).

One standard way to thwart this kind of attack is based on secret sharing and is called t^{th} -order masking when each sensitive variable is split into numerous shares in a way such that t of them give no information on this variable. These shares must then be propagated independently throughout the algorithm to ensure its resistance against t^{th} -order SCA. In particular, when applying a *linear* secret sharing for a masked implementation of the AES cipher, shares can be easily propagated through all *linear* operations. However, much more work is needed to deal with the inversion in the finite field \mathbf{F}_{2^8} involved in the AES Sbox. When masking AES, this inversion is usually (see *e.g.*, [TDG02, BMK04, OMPR05]) computed using exponentiation so masking this operation comes down to masking multiplications of sensitive variables. This last problem has been extensively addressed in the secure Multi-Party Computation literature. For instance in [BOGW88, CCD88], the authors have introduced a secure multiplication procedure for the secret sharing scheme of Shamir ([Sha79]). Much later, in [ISW03], Ishai, Sahai and Wagner have proposed another procedure applied for a basic secret sharing scheme, commonly used to design countermeasures, the so-called *Boolean masking*. These two methods have been used in the past few years to propose high-order masking schemes for AES ([RP10, KHL11, CPRR13] with Boolean Masking, and [GM11, PR11, CPR12] with Shamir's secret sharing scheme).

Let us mention that, quite recently, the authors of [BFGV12] have suggested an adaptation of the nonlinear masking technique described in [DF12] to design an implementation of AES resistant against high-order SCA. In particular, they improved the secure multiplication of [DF12]. However this scheme being nonlinear, the implementation of linear operations becomes expensive compared to linear schemes.

Another idea is to take advantage of the fact that every linear code gives rise to a linear secret sharing scheme as described in [Mas93] for example. In practice, MDS codes, such as the parity check code (corresponding to Boolean masking) and Reed-Solomon code (for Shamir's secret sharing scheme) are generally used to design countermeasures as t^{th} -order SCA resistance is achieved with only $t + 1$ shares. Moreover, the initial Multi-Party Computation protocol proposed in [BOGW88, CCD88] for Shamir's secret sharing scheme has been generalized by Cramer *et al.* ([CDM00]) from any linear secret sharing scheme and family of codes such as *self-dual* and *geometric* codes have been suggested to improve performance of the distributed multiplication procedure ([CC06, CDG⁺08]).

Our Contribution. In this paper, we propose to take advantage of results of coding theory and Multi-Party Computation to design new t^{th} -order masking techniques by selecting linear codes adapted to the masked operations. More precisely, we suggest to use *self-dual codes with a binary basis* to create secret sharing schemes in which secrets and shares belong to an extension field K of

the binary field \mathbf{F}_2 . These codes will provide low-cost square operations¹ and an efficient encoding which is extensively used as a subroutine of the secure multiplication procedure. Encoding require *zero multiplication* over the large field, contrary to all strategies based on Shamir’s secret sharing scheme. As a result, our secure multiplication procedure needs $\mathcal{O}(t)$ multiplications whereas the methods based on Shamir’s secret sharing scheme ([GM11, PR11]) involve $\mathcal{O}(t^3)$ multiplications², due to a costly encoding procedure, and the method given in [RP10], using Boolean masking, has complexity $\mathcal{O}(t^2)$ multiplications. Moreover, we propose an improvement of the multiplication procedure given in [BOGW88, CCD88, CDM00] which can also be applied for Shamir’s secret sharing scheme.

Our codes are non-MDS, so we need more than $t+1$ shares to achieve t^{th} -order SCA resistance. However, thanks to the underlying binary structure of our code, we show that it is possible to efficiently switch code to the same code used for Boolean masking during linear operations. As a result, these linear operations can be masked as efficiently as with Boolean masking. We also propose to work with an underlying self-dual code over \mathbf{F}_4 which provides a low-cost $x \mapsto x^4$ operation over K with less shares.

Finally, in the context of an implementation of a t^{th} -order secure AES, we show that our masking proposal dramatically improves the method with Shamir’s secret sharing scheme and is competitive with Boolean masking.

Paper Organization. In Section 2, we recall the connection between t^{th} -order SCA countermeasures and secret sharing schemes and present the approach with Shamir’s secret sharing scheme. In Section 3, we present the construction of linear secret sharing schemes derived from linear codes and the general Multi-Party Computation multiplication procedure of [CDM00] that generalizes [BOGW88, CCD88]. Section 4 is the core of our proposal: we present a new t^{th} -order *masking* technique based on self dual codes, and several implementation improvements. In Section 5, we apply our technique to secure an AES implementation present experimental results and make a comparison with previous works. Finally Section 5 concludes the paper.

2 Secret Sharing Scheme and t^{th} -Order Masking

An implementation of a secret key algorithm is said to be t^{th} -order secure if an adversary gains no information about the secret key from the knowledge of t intermediate values. This ensures that the observation of the physical leakage related to the manipulation of these intermediate values will not help the adversary in performing a key recovery attack. A sound approach to reach this level of

¹ We note that a similar trick have been proposed to make efficient squaring in the long version of [PR11] but only with Reed-Solomon codes which are different to the codes used in our proposal.

² This can be improved to $\mathcal{O}(t^2 \log^4 t)$ multiplications by using discrete Fourier transform cf. [CPR12].

security is to mask each sensitive variable s with a linear secret sharing scheme. For example, the so-called Boolean masking consists in randomly splitting s into $t+1$ shares s_1, \dots, s_{t+1} in a way such that $s = s_1 + \dots + s_{t+1}$ ([RP10, CPRR13]). The linear secret sharing scheme of Shamir has also been used in this context ([GM11, PR11, CPR12]).

Some mechanisms must be developed to make this masking procedure compatible with the operations performed in the protected algorithm, *i.e.*, to enable the computation on masked data. For example, for the AES cipher, linear operations are compatible with linear secret sharing. Nonlinear functions involved in the *SubBytes* transformation are more difficult to deal with. Usually the inversion involved in this transformation is computed using an exponentiation which requires a method to protect multiplications of sensitive variables. In the context of the AES, some solutions have been developed in [RP10, KHL11, CPRR13] for multiplication with Boolean masking.

In this section, after some definitions on secret sharing schemes, we describe the solution, close to our proposal, given in [GM11, PR11, CPR12] still in the context of AES, to perform multiplications with Shamir's secret sharing scheme.

2.1 Definitions

Let K be a field. A secret sharing scheme is a method to split a secret $s \in K$ among a set of n shares. More precisely a secret sharing scheme is composed of two algorithms, *encoding* and *decoding*. The *encoding* of s provides an n -vector of shares called *share vector*: $(s_1, \dots, s_n) \in K^n$. The *decoding* algorithm reconstructs the secret s from (s_1, \dots, s_n) .

A secret sharing scheme has *t-privacy* if any set of at most t shares reveals no information about the secret, and *r-reconstruction* if r shares reveal the entire secret. A secret sharing scheme is said to be *linear* if for any two secrets s and s' shared respectively by (s_1, \dots, s_n) and (s'_1, \dots, s'_n) , the vectors $(s_1 + s'_1, \dots, s_n + s'_n)$ and $(\lambda s_1, \dots, \lambda s_n)$ decode respectively to $s + s'$ and λs , $\lambda \in K$.

A t^{th} -order secure implementation of an algorithm can be based on a linear secret sharing scheme with t -privacy. The *encoding* procedure is applied to each input variable. Share vectors are then manipulated to reflect the protected algorithm. Additions and scalar multiplications of secrets are performed easily on the share vectors. In the following, we recall the linear secret sharing scheme of Shamir and the method to perform t^{th} -order secure multiplication of secrets described in [GM11, PR11, CPR12].

2.2 Shamir's Secret Sharing Scheme

In [Sha79], Shamir has introduced a linear secret sharing scheme over a finite field \mathbf{F}_q based on polynomial interpolation. The *encoding* step consists in generating a random secret polynomial P of degree t over \mathbf{F}_q such that $P(0) = s$. Then, the share vector of s denoted (s_1, \dots, s_n) is generated by evaluating $s_i = P(x_i)$ in n distinct non-zero points x_1, \dots, x_n of \mathbf{F}_q . Let A be a subset of $\{1, \dots, n\}$ such that $|A| \geq t + 1$. The *decoding* step consists in recovering the secret polynomial

by interpolation and then the secret by an evaluation at 0. This is done by computing:

$$s = \sum_{i \in A} s_i \beta_i(0), \tag{1}$$

where $\beta_i(X) = \prod_{j \in A, j \neq i} \frac{X - x_j}{x_i - x_j}$ are Lagrange polynomials. This gives a linear secret sharing scheme with t -privacy and $t + 1$ -reconstruction.

Linear operations on secrets can be securely executed as described in subsection 2.1. However nonlinear operations over \mathbf{F}_q are more complex to deal with. Multiplication of two secrets has been extensively studied to design secure multiparty computation schemes, *e.g.* in [BOGW88, CCD88, GRR98]. The method consists in multiplying the two share vectors share by share. This operation corresponds to the multiplication of two degree t polynomials which gives a polynomial of degree $2t$. The secret result of the multiplication can then be recovered with at least $2t + 1$ shares if $n \geq 2t + 1$. A method to reduce the number of shares required to reconstruct the secret is needed, otherwise k successive multiplications would require to take $n \geq kt + 1$. The solution consists in *re-encoding* $2t + 1$ shares and to compute the sum of the resulting share vectors.

In [GM11, PR11], this secure multiplication has been used to implement the AES cipher. Algorithm 1 recalls the solution to perform the secure multiplication procedure as given in [GM11]. In particular the authors have suggested to take $n = t + 1$ during the whole process and to generate on-the-fly the additional shares when a multiplication step is required reducing the overall complexity of their AES implementation.

Algorithm 1. Secure Multiplication [GM11, Algorithm 2]

INPUTS: $2n - 1$ distinct non-zero public elements x_1, \dots, x_{2n-1}
 (s_1, \dots, s_n) a share vector of s and (s'_1, \dots, s'_n) a share vector of s'
 $\beta_j(x_i)$ pre-computed for $1 \leq j \leq n$ and $n + 1 \leq i \leq 2n - 1$
 β_i^* pre-computed for $1 \leq i \leq 2n - 1$ with $\beta_i^* = \prod_{j=1, j \neq i}^{2n-1} \frac{-x_j}{x_i - x_j}$
OUTPUT: (z_1, \dots, z_n) a share vector of ss'

1. **For** $i = n + 1$ **to** $2n - 1$ **do**
 2. $s_i \leftarrow \sum_{j=1}^n s_j \beta_j(x_i)$
 3. $s'_i \leftarrow \sum_{j=1}^n s'_j \beta_j(x_i)$
 4. **For** $i = 1$ **to** $2n - 1$ **do**
 5. $w_i = s_i s'_i \beta_i^*$
 6. $(w_{i_1}, \dots, w_{i_n}) \leftarrow \text{encoding}(w_i)$
 7. **For** $j = 1$ **to** n **do**
 8. $z_j \leftarrow \sum_{i=1}^{2n-1} w_{i_j}$
 9. **Return** (z_1, \dots, z_n)
-

Remark 1. When the field \mathbf{F}_q has characteristic 2, the square of a secret can be performed more efficiently than general multiplication. Let us consider the square (s_1^2, \dots, s_n^2) of a share vector of s and let us denote P the secret polynomial used to share s . Relation (1) gives $s^2 = \sum_{i \in A} s_i^2 \beta_i(0)^2$, so the secret s^2 can be

recovered with only $(t + 1)$ shares. However each share s_i^2 is an evaluation of P^2 at x_i^2 instead of x_i . In [GM11] the authors propose again to apply a *re-encoding* step so that the shares are the evaluation of a polynomial at (x_1, \dots, x_n) . In the long version of [PR11] it is proposed to choose the points x_1, \dots, x_n such that the set of these points is stable with respect to the Frobenius map $x \mapsto x^2$. As a result, a simple re-ordering of the shares is needed instead of a *re-encoding* step. The situation will be even simpler with our proposal since no re-ordering will be needed.

The main cost of Algorithm 1 comes from the numerous multiplications by constant values, namely the multiplications by β_i^* and $\beta_j(x_i)$ and the ones required during the *encoding* steps. These steps correspond to evaluations of polynomials at n different points. In [CPR12], the authors propose to use the discrete Fourier transform for these computations. Hence the whole secure multiplication procedure can be improved to a complexity of $\mathcal{O}(n^2)$ multiplications instead of the naive approach in $\mathcal{O}(n^3)$ multiplications. With our proposal, no multiplication by constant values over \mathbf{F}_q will be needed, as a result of which the secure multiplication procedure will be improved to a complexity of $\mathcal{O}(n)$ multiplications.

3 Coding Theory Generalisation

It is well-known that a linear secret sharing scheme can be built from a linear code as described, for example, in [Mas93, CC06, CCG⁺07, CDG⁺08]. Moreover, the problem of computing on masked data (addition and multiplication of secret values) for general secret sharing schemes has been addressed to design secure multiparty computation protocol (see, *e.g.*, [CDM00]), generalizing the protocol initially proposed in [BOGW88, CCD88] with Shamir's scheme. In the following, we recall some basic definitions, explain the construction of a linear secret sharing scheme from a linear code and describe the algorithms to perform secure computation on secrets that generalize the algorithms given in the previous section.

3.1 Basic Definitions and Results from Coding Theory

Over a finite field \mathbf{F}_q , an $[n + 1, k + 1, d]_q$ linear code \mathcal{C} is a $(k + 1)$ -dimensional vector subspace of \mathbf{F}_q^{n+1} with minimum Hamming distance d . The generator matrix G of a linear code in systematic form can be written as $G = [Id_{k+1} | \mathcal{A}]$, where \mathcal{A} is a $(k+1) \times (n-k)$ -matrix. The elements $c = (c_0, \dots, c_n)$ of \mathcal{C} are called codewords and can be generated as $c = (r_0, \dots, r_k) \cdot G$, where $(r_0, \dots, r_k) \in \mathbf{F}_q^{k+1}$. The dual code \mathcal{C}^\perp of \mathcal{C} is an $[n + 1, n - k]_q$ linear code defined by

$$\mathcal{C}^\perp = \{c \in \mathbf{F}_q^{n+1} : \langle c, c' \rangle = 0 \text{ for all } c' \in \mathcal{C}\},$$

where $\langle \cdot, \cdot \rangle$ denotes the *inner product* defined by $\langle c, c' \rangle = \sum_{i=0}^n c_i c'_i$. When $\mathcal{C} = \mathcal{C}^\perp$ (resp. $\mathcal{C} \subseteq \mathcal{C}^\perp$), \mathcal{C} is said to be *self-dual* (resp. *self-orthogonal*).

From \mathcal{C} an $[n + 1, k + 1]_q$ linear code, the *squared code* denoted $\widehat{\mathcal{C}}$ is defined by

$$\widehat{\mathcal{C}} = \langle \{c * c', c, c' \in \mathcal{C}\} \rangle,$$

where $*$ denotes the *Schur product* $c * c' = (c_0c'_0, \dots, c_nc'_n)$. Moreover we define \mathcal{C}^2 as

$$\mathcal{C}^2 = \langle \{c^2 = c * c, c \in \mathcal{C}\} \rangle.$$

For more details on linear codes, the interested reader may refer to [MS78]. In the following we give two lemmas useful to select suitable codes for efficient operations on masked data.

Lemma 1. *Let \mathcal{C} be an $[n, k]$ linear code over \mathbf{F}_q of characteristic 2. The following assertions are equivalent:*

1. $\forall c \in \mathcal{C}, c^2 \in \mathcal{C}$,
2. \mathcal{C} has a binary basis.

Proof. Let \mathcal{C} be a linear code over \mathbf{F}_q of characteristic 2 having a binary basis (b_1, \dots, b_k) with $b_i \in \mathbf{F}_2^n$. For all i , $b_i^2 = b_i$. If c is a codeword of \mathcal{C} , then there exists a linear combination such that $c = \sum_{i=1}^k \lambda_i b_i$ with $\lambda_i \in \mathbf{F}_q$. We have $c^2 = \sum_{i=1}^k \lambda_i^2 b_i^2 = \sum_{i=1}^k \lambda_i^2 b_i \in \mathcal{C}$.

Conversely, let \mathcal{C} be a $[n, k]$ linear code over \mathbf{F}_q , such that for all codeword $c \in \mathcal{C}, c^2 \in \mathcal{C}$. Let G be a generator matrix of \mathcal{C} in systematic form and b_i be the i^{th} row of G . Let a_1, \dots, a_{n-k} be elements of \mathbf{F}_q such that $b_i = (0, \dots, 0, 1, 0, \dots, 0, a_1, \dots, a_{n-k})$.

By definition $b_i^2 = (0, \dots, 0, 1, 0, \dots, 0, a_1^2, \dots, a_{n-k}^2) \in \mathcal{C}$, so there exists a linear combination such that $b_i^2 = \sum_{i=1}^k \lambda_i b_i$. From the last equality, the j^{th} coordinate of b_i^2 is equal to λ_j for $j \in \{1, \dots, k\}$. By identification, we have for $1 \leq j \leq k$ and $j \neq i$, $\lambda_j = 0$, and $\lambda_i = 1$. Therefore $b_i^2 = b_i$ and then $a_j^2 = a_j$ for $j \in \{1, \dots, n - k\}$. Thus $b_i \in \mathbf{F}_2^n$. \square

Lemma 2. *If \mathcal{C} is a linear self-dual code (or a linear self-orthogonal code), then the codeword $\mathbf{1} = (1, \dots, 1) \in \widehat{\mathcal{C}}^\perp$.*

Proof. For all $c, c' \in \mathcal{C}$ of the self-dual (or self-orthogonal) code, we have: $\langle c, c' \rangle = \langle c * c', \mathbf{1} \rangle = 0$. \square

3.2 Construction of a Linear Secret Sharing Scheme from a Linear Code

Let \mathcal{C} be an $[n + 1, k + 1, d]_q$ linear code with G its generator matrix in systematic form. All codewords $c = (c_0, c_1, \dots, c_n)$ of \mathcal{C} can be identified with a share vector (c_1, \dots, c_n) of the secret $c_0 = s$. Hence the *encoding* procedure of a secret $s \in \mathbf{F}_q$ consists in generating a codeword $c = (s, r_1, \dots, r_k) \cdot G$, where r_1, \dots, r_k are random values of \mathbf{F}_q . In case of ambiguity, this procedure is denoted *encoding* $_{\mathcal{C}}$. Assuming that there exists a codeword $h = (h_0, \dots, h_n)$ of \mathcal{C}^\perp such that $h_0 = 1$,

the *decoding* procedure can be implemented by computing $s = \sum_{i=1}^n \lambda_i c_i$ where $\lambda_i = -h_i \in \mathbf{F}_q$. In this case, we call *recombination vector* of \mathcal{C} , such a vector $\lambda = (\lambda_1, \dots, \lambda_n)$ where the number of non-zero λ_i equals to $d^\perp - 1$.

In [CCG⁺07, Theorem 1], it is shown that such a linear secret sharing scheme has $(d^\perp - 2)$ -privacy and $(n - d + 2)$ -reconstruction, where d^\perp is the minimum distance of \mathcal{C}^\perp .

3.3 Operations on Masked Data

A linear code gives a linear secret sharing scheme, so addition and scalar multiplication of secrets correspond to addition and scalar multiplication of share vectors. Consider now the problem of multiplying two secrets shared by a general secret sharing scheme. In other words, from the shared vectors of two secrets one wishes to obtain a shared vector representing the product of the secrets by using operations on shares and without reconstructing the secrets. This problem has been addressed in [CDM00]. In this work, a procedure that generalizes Algorithm 1 is given, by considering any linear code \mathcal{C} such that $\widehat{d}^\perp \geq d^\perp$ where \widehat{d}^\perp is the minimum distance of $\widehat{\mathcal{C}}$. We describe this approach below.

We assume that there exists a recombination vector $\widehat{\lambda}$ of $\widehat{\mathcal{C}}$ (for example, according to Lemma 2, we can choose $\widehat{\lambda} = (1, \dots, 1)$ if \mathcal{C} is self-dual or self-orthogonal). Let $c, c' \in \mathcal{C}$ be such that $c_0 = s$ and $c'_0 = s'$. The secret multiplication ss' can be shared by $c * c' \in \widehat{\mathcal{C}}$ and recovered by computing $ss' = \sum_{i=1}^n \widehat{\lambda}_i c_i c'_i$. Furthermore $\widehat{\mathcal{C}}$ gives a secret sharing scheme with $(\widehat{d}^\perp - 2)$ -privacy and as $\widehat{d}^\perp \geq d^\perp$, the privacy level of the secret sharing scheme associated to \mathcal{C} is preserved. To be able to perform further multiplications, we need a method for *re-encoding* the codeword $c * c' \in \widehat{\mathcal{C}}$ into a new codeword $z \in \mathcal{C}$ such that $z_0 = ss'$. As seen in the previous section, this can be done by *re-encoding* each $\widehat{\lambda}_i c_i c'_i$ into a new codeword of \mathcal{C} . Then by summing these n codewords, we obtain z , a codeword of \mathcal{C} , such that $z_0 = ss'$. This procedure is described in the following algorithms: Algorithm 3 for the secure multiplication procedure, and Algorithm 2 for the *re-encoding* subroutine.

Algorithm 2. Secure Re-encoding Procedure

INPUTS: \mathcal{C} a $[n + 1, k + 1]_q$ linear code

\mathcal{C}' a $[n' + 1, k' + 1]_q$ linear code and λ' a recombination vector

$(w_1, \dots, w_{n'})$ a share vector corresponding to a codeword of \mathcal{C}'

OUTPUT: (z_1, \dots, z_n) a share vector of $\sum_{i=1}^{n'} \lambda'_i c_i$ corresponding to a codeword of \mathcal{C}

Function: *re-encoding* $\mathcal{C}' \rightarrow \mathcal{C}(\lambda', w_1, \dots, w_{n'})$

1. **For** $i = 1$ **to** n' **do**
 2. $(w_{i_1}, \dots, w_{i_n}) \leftarrow \text{encoding}_{\mathcal{C}}(\lambda'_i w_i)$
 3. **For** $j = 1$ **to** n **do**
 4. $z_j \leftarrow \sum_{i=1}^{n'} w_{i_j}$
 5. **Return** (z_1, \dots, z_n)
-

Algorithm 3. Secure Multiplication Procedure

INPUTS: \mathcal{C} a $[n + 1, k + 1]_q$ linear code

(c_1, \dots, c_n) a share vector of s and (c'_1, \dots, c'_n) a share vector of s' corresponding to codewords of \mathcal{C}

$\widehat{\lambda}$ a recombination vector of $\widehat{\mathcal{C}}$

OUTPUT: (z_1, \dots, z_n) a share vector of ss' corresponding to a codeword of \mathcal{C}

1. $(w_1, \dots, w_n) \leftarrow (c_1 c'_1, \dots, c_n c'_n)$
 2. $(z_1, \dots, z_n) \leftarrow \text{re-encoding } \widehat{c}_{\rightarrow \mathcal{C}}(\widehat{\lambda}, w_1, \dots, w_n)$
 3. **Return** (z_1, \dots, z_n)
-

These algorithms requires n multiplications of shares and numerous multiplications by constant values: the coordinates of $\widehat{\lambda}$ and the elements of the matrix G for the encodings. In our proposal, all these elements will be binary, so only n multiplications will be needed for a secure multiplication.

Fact 1. *From the properties of the secure Multi-Party Computation protocol given in [CDM00, Section 6], Algorithms 2 and 3 give a t^{th} -order secure multiplication, with $t = d^\perp - 2$ where d^\perp is the dual distance of the linear code \mathcal{C} .*

Remark 2. Over a finite field \mathbf{F}_q of characteristic 2, the square of a secret s can be computed without using the squared code $\widehat{\mathcal{C}}$ and $\widehat{\lambda}$. Indeed, if $(1, \lambda_1, \dots, \lambda_n) \in \mathcal{C}^\perp$, then $(1, \lambda_1^2, \dots, \lambda_n^2) \in \mathcal{C}^{2^\perp}$. As a result, if s is shared by (c_1, \dots, c_n) , one can apply Algorithm 2 to $\lambda^2, (c_1^2, \dots, c_n^2)$, to obtain a share vector of s^2 corresponding to a codeword of \mathcal{C} . Moreover, no re-encoding is needed if $c^2 \in \mathcal{C}$. According to Lemma 1, this will be the case if and only if \mathcal{C} has a binary basis. In this case, the secure squaring procedure only requires to compute n squares.

3.4 Application to Boolean Masking and to Shamir’s Secret Sharing Scheme

The well-known Boolean masking can be obtained with this framework by using the $[n + 1, n]_q$ linear parity check code with generator matrix

$$G = \left(\begin{array}{c|c} & 1 \\ Id_n & \vdots \\ & 1 \end{array} \right). \tag{2}$$

The dual of this code is generated by the codeword $(1, \dots, 1)$. As a consequence, the secret sharing scheme constructed from such a code has $(n - 1)$ -privacy and the secret $s \in \mathbf{F}_q$ can be recovered by computing $s = \sum_{i=1}^n c_i$, where (c_1, \dots, c_n) is a share vector of s . As said in Remark 2 this scheme allows a secure squaring procedure with a low computational cost. However for $n + 1 \geq 2$, it is easy to see that $\widehat{\mathcal{C}}^\perp = \{0\}$. As a consequence we cannot apply

Algorithm 3 to perform a secure multiplication. Nevertheless other methods are proposed in the literature to perform such an operation without using the framework of error-correcting codes (for instance the method given in [RP10] requires roughly n^2 multiplications of shares).

Shamir’s secret sharing scheme can be constructed from the Reed-Solomon code of parameters $[n + 1, t + 1]_q$ with the generator matrix

$$G = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & x_1 & \dots & x_n \\ 0 & x_1^2 & \dots & x_n^2 \\ \vdots & \vdots & \dots & \vdots \\ 0 & x_1^t & \dots & x_n^t \end{pmatrix},$$

with for all i, j , $i \neq j$, $x_i \neq x_j \in \mathbf{F}_q \setminus \{0\}$. The recombination vector λ can be chosen such that $\lambda_i = \beta_i(0)$ (*i.e.*, Lagrange polynomials evaluated in 0). Moreover, $\widehat{\mathcal{C}}$ is the $[n+1, 2t+1]_q$ Reed-Solomon code and if $2t+1 \leq n$, then we can apply Algorithm 3. This gives essentially Algorithm 1 where the only difference is the on-the-fly computation of the missing shares. Similarly the method of [GM11] to compute secure squaring for Shamir’s scheme explained in Remark 1, corresponds to the method given in Remark 2.

4 Our Contribution

The codes generally considered for the construction of secret sharing schemes are MDS codes, such as the parity check code (for Boolean masking) and Reed-Solomon codes (for Shamir’s scheme). This is because they give *perfect* secret sharing schemes, namely t -privacy and $t+1$ -reconstruction. In particular, when used as t^{th} -order masking, only $t + 1$ shares can be used to mask each input variable.

The efficiency of Boolean masking comes from the fact that it corresponds to a code with a *binary* generator matrix (2). By lemma 1, this binary basis implies that squaring in an extension of the binary field is a low-cost operation. Moreover, *encoding* requires only additions and no multiplications. As mentioned in subsection 3.4, the general secure multiplication (Algorithm 3) cannot be applied to Boolean masking, and the secure multiplication algorithm of [RP10] requires $\mathcal{O}(t^2)$ multiplications. For Shamir’s secret sharing scheme [PR11, GM11], secure multiplication corresponds to Algorithm 3. But the *encoding* subroutine needs numerous multiplications in the finite field and secure multiplication has complexity $\mathcal{O}(t^3)$ or $\tilde{\mathcal{O}}(t^2)$ multiplications with FFT techniques ([CPR12]), even if only $\mathcal{O}(t)$ multiplications of *shares* are needed. Moreover, some work has been done to make the squaring procedure more efficient (cf. remark 1).

We propose to select a family of non-MDS linear codes over an extension of the binary field such that Algorithm 3 is applicable like in Shamir’s scheme, and such that a binary basis is available like in Boolean masking. With these codes we will have *all the benefits*: an *encoding* subroutine which requires zero

multiplication, a low-cost square operation (like in Boolean masking), and a secure multiplication procedure which requires only $\mathcal{O}(t)$ multiplications and is thus more efficient than methods with MDS codes. The codes that we use have an underlying binary structure which is that of a self-dual or a self-orthogonal binary code. We also discuss the case of codes defined with a basis in \mathbf{F}_4 .

Moreover, we propose an improvement of the multiplication procedure described in Algorithm 3 which can also be applied for Shamir's secret sharing scheme.

Finally, to compensate the additional number of shares needed by the fact that our codes are non-MDS, we show that it is possible to efficiently switch code to the same code used for Boolean masking during linear operations, thanks to the underlying binary structure of our codes. As a result, these linear operations can be masked as efficiently as with Boolean masking.

Table 3 sums up the costs of our masking procedure when applying all these improvements.

4.1 Linear Secret Sharing Schemes Based on Self-Dual Codes

In all the following we consider a base field \mathbf{F}_q of characteristic 2. As previously said, we want to select a linear code \mathcal{C} over \mathbf{F}_q with a binary basis. Moreover, as described in Section 3, to ensure that the multiplication procedure given by Algorithm 3 is applicable, $\widehat{\mathcal{C}}^\perp$ must contain a particular codeword h such that $h_0 = 1$. Lemma 2 shows that by choosing for \mathcal{C} a self-dual or self-orthogonal code, such a property is fulfilled.

As described in Section 3, a linear $[n + 1, k + 1]$ code allows to construct a secret sharing scheme with n shares and t -privacy, where $t = d^\perp - 2$. For a fixed value t , the number of shares must be the smallest possible to reduce the total number of operations of a t^{th} -order masking. In the coding literature, self-dual and self-orthogonal binary codes are well-studied. In particular, we can give in Table 1 a list of binary codes \mathcal{C} with minimal length and dual distance $d^\perp = t + 2$ for $1 \leq t \leq 6$, such that for even length, the codes \mathcal{C} are self-dual and for odd length, self-orthogonal. The code used for our t^{th} -order masking is not strictly speaking the code \mathcal{C} but the code over \mathbf{F}_q generated by the binary generator matrix of \mathcal{C} . The code constructed over \mathbf{F}_q remains self-dual or self-orthogonal, and has the same properties (length, dimension, minimum distance and dual distance) than the underlying binary code.

In this table, the self-dual shortened Golay code [22,11,6] is built from the extended Golay code by using codewords beginning by 00 and 11. The code C_{21} [21,11,5] is obtained by removing a coordinate of the shortened Golay code. For larger values of t , the reader is referred to the codes given in [CS90, GO03], and it is known that n will be linear in t .

Interestingly, self-dual binary codes and some generalisations have been called upon [CGKS12, CG13] in order to improve resistance against side-channel analysis, through in contexts that do not invoke secret sharing.

To construct a t^{th} -order masking scheme, we select the $[n + 1, k + 1]$ code from the line t of Table 1. The *encoding* procedure (cf. subsection 3.2) requires the

Table 1. List of binary codes

t	Binary code \mathcal{C} [$n + 1, k + 1$]	Binary Dual code \mathcal{C}^\perp with $d^\perp = t + 2$	Number of additions required during an <i>encoding</i>
1	Code [7, 3]	Hamming code [7, 4, 3]	5
2	Extended Hamming code [8, 4, 4]		8
3	Code [21, 10]	C_{21} [21, 11, 5]	48
4	Shortened Golay code [22, 11, 6]		44
5	Code [23, 11]	Golay code [23, 12, 7]	64
6	Extended Golay code [24, 12, 8]		72

generation of k random values over \mathbf{F}_q and only L additions, where L is given in Table 1 and depends on the number of 1s in the generator matrix considered. For this *encoding* step, there is no multiplication by constant values of \mathbf{F}_q unlike in Shamir’s scheme. The addition and scalar multiplication of a secret are computed on each of the n shares, so n operations are needed. Squaring also consists in squaring the n shares. Secure multiplication is done with Algorithm 3. The vector $\hat{\lambda}$ is defined over \mathbf{F}_2 , so only n multiplications in \mathbf{F}_q are required.

From Table 1, we note that the number of shares n of our t^{th} -order masking scheme is important compared to a perfect scheme such as Shamir’s. For example for $t = 3$, we need $n = 20$ shares and with a perfect masking only 4. To improve the performance of our masking method, a solution is to consider an underling self-dual or self-orthogonal code over \mathbf{F}_4 instead of \mathbf{F}_2 if $\mathbf{F}_4 \subset \mathbf{F}_q$. Indeed, as we can see in [GO03] such codes provide a better ratio n/t . With such a code the generator matrix has now its coefficients in $\mathbf{F}_4 = \{0, 1, w, w + 1\} \subset \mathbf{F}_q$. As a result the *encoding* procedure requires some multiplications by w . However, unlike in Shamir’s scheme, here only one constant value, *i.e.*, w is manipulated and the products $wx, x \in \mathbf{F}_q$ can be precomputed in a table.

In Table 2, we give a list of optimal codes over \mathbf{F}_4 and indicate the number of additions and *the number of low-cost multiplications with w* required during an encoding procedure. As with Table 1, the code used in our masking scheme is a self-dual or a self-orthogonal code over \mathbf{F}_q built from the generator matrix of the code given in the table. With these codes, we lose the advantage of the low-cost squaring operation: the secure squaring now requires a re-encoding procedure as described in Remark 2. However the complexity of raising a sensitive variable to the power 4 is still small. Indeed by adapting Lemma 1, we can show that for any codeword $c \in \mathcal{C}$, we have $c^4 \in \mathcal{C}$.

4.2 Improvement of Secure Multiplication

During a secure multiplication (Algorithm 3), the most expensive step is the re-encoding process from $\hat{\mathcal{C}}$ to \mathcal{C} where n calls are done to the *encoding* procedure.

Table 2. List of codes over \mathbf{F}_4

t	Code \mathcal{C} over \mathbf{F}_4 [$n + 1, k + 1$]	Dual code \mathcal{C}^\perp with $d^\perp = t + 2$	encoding	
			add	mult with w
1	Extended quadratic residue code XQR(3) [4, 2, 3]		4	2
3	Code [11, 5]	Quadratic residue code QR(11) [11, 6, 5]	25	5
4	Extended quadratic residue code XQR(11) [12, 6, 6]		32	6
5	Code [19, 9]	Quadratic residue code QR(19) [19, 10, 7]	69	9

In order to reduce the complexity of this algorithm, we show that this number of calls can be decreased. This modified algorithm will still be t^{th} -order SCA secure with $t = d^\perp - 2$.

After the multiplication of two share vectors of s and s' , we obtain a share vector $(c_1c'_1, \dots, c_nc'_n)$ of ss' corresponding to a codeword of $\widehat{\mathcal{C}}$. If we add the vector $(c_1c'_1, \dots, c_nc'_n)$ and a random share vector of 0 in $\widehat{\mathcal{C}}$, then we obtain a *random*³ share vector in $\widehat{\mathcal{C}}$ of ss' denoted $w = (w_1, \dots, w_n)$. Furthermore, $\widehat{\mathcal{C}}$ gives a secret sharing scheme with $(\widehat{d}^\perp - 2)$ -privacy. Assuming that $e = \widehat{d}^\perp - d^\perp > 0$, we combine $(e + 1)$ elements of w giving a share vector

$$\tilde{w} = \left(\sum_{i=1}^{e+1} \widehat{\lambda}_i w_i, w_{e+2}, \dots, w_n \right)$$

associated to a linear code $\widehat{\mathcal{C}}^*$ of length $n - e$. By construction, the vector $\widehat{\lambda}^* = (1, \widehat{\lambda}_{e+2}, \dots, \widehat{\lambda}_n)$ is a recombination vector of this code if $(\widehat{\lambda}_1, \dots, \widehat{\lambda}_n)$ is a recombination vector of $\widehat{\mathcal{C}}$. The algorithm is still t^{th} -order SCA secure if we re-encode the coordinates of this share vector \tilde{w} instead of w . Indeed, suppose that an adversary has access to a subset of t shares of this vector \tilde{w} . If the first coordinate is not in this subset, the adversary has no information on the secret as the scheme is at least t -private. If he knows the first coordinate he has less information than with w_1, w_2, \dots, w_{e+1} and $t - 1$ others shares, *i.e.*, with $\widehat{d}^\perp - 2$ shares, so he has again no information on the secret.

Therefore, only $n - e$ shares can be re-encoded during the secure multiplication procedure as described in Algorithm 4.

This improvement can be applied for each linear code with $\widehat{d}^\perp > d^\perp$. In particular, the squared codes $\widehat{\mathcal{C}}$ associated to the codes given in Tables 1 and 2 are the parity check codes of length $n + 1$ and have $\widehat{d}^\perp = n + 1$, so only $n - e = t + 1$ shares have to be re-encoded in Step 5. Similarly, for Shamir's secret sharing scheme by taking $n = 2k + 1$, then we have that $t = k$, $d^\perp = k + 2$ and the squared code $\widehat{\mathcal{C}}$ associated is the Reed-Solomon code of parameters $[2k + 2, 2k + 1]$, so $\widehat{d}^\perp = 2k + 2$. Therefore $n - e = t + 1$.

³ If this step is omitted, the vector $y = c * c'$ of $\widehat{\mathcal{C}}$ may not have $(\widehat{d}^\perp - 2)$ -privacy. For example, if $s = s'$ and the two input share vectors are equals, $y \in \mathcal{C}^2$ and as $\mathcal{C}^2 = \mathcal{C}$ in our proposal, this vector has only $(d^\perp - 2)$ -privacy.

Algorithm 4. Improvement of Secure Multiplication

 INPUTS: \mathcal{C} a $[n+1, k+1]_q$ linear code

 (c_1, \dots, c_n) and (c'_1, \dots, c'_n) two share vectors respectively of s and s'
 $\widehat{\lambda}$ a recombination vector of $\widehat{\mathcal{C}}$ and $e = \widehat{d}^\perp - d^\perp > 0$

 OUTPUT: (z_1, \dots, z_n) a share vector of ss'

Function: SecMult($(c_1, \dots, c_n), (c'_1, \dots, c'_n)$)

 1. $(w_1, \dots, w_n) \leftarrow (c_1 c'_1, \dots, c_n c'_n) + \text{encoding}_{\widehat{\mathcal{C}}}(0)$

 2. $(w_1, \dots, w_n) \leftarrow (\widehat{\lambda}_1 w_1, \dots, \widehat{\lambda}_{e+1} w_{e+1}, w_{e+2}, \dots, w_n)$

 3. **For** $i = 1$ **to** e **do**

 4. $w_{e+1} \leftarrow w_{e+1} + w_i$

 5. $(z_1, \dots, z_n) \leftarrow \text{re-encoding}_{\widehat{\mathcal{C}}^* \rightarrow \mathcal{C}}((1, \widehat{\lambda}_{e+2}, \dots, \widehat{\lambda}_n), (w_{e+1}, \dots, w_n))$

 6. **Return** (z_1, \dots, z_n)

4.3 Code Switching to Perform Efficient Linear Operations

To compensate the additional number of shares needed by the fact that our codes are non-MDS, we propose a solution to *reduce the number of shares used during linear operations*, still achieving a t^{th} -order masking. Let us consider the masking scheme using a code \mathcal{C} built from Table 1. Thanks to the underlying binary structure, it is possible to efficiently re-encode the share vectors of \mathcal{C} to an MDS code \mathcal{C}^* for linear operations (additions and scalar multiplications). This simply consists in considering only the shares involved in the reconstruction, namely the $t+1$ shares corresponding to the non-zero coordinates of the recombination vector. As a result, the MDS code \mathcal{C}^* corresponds to Boolean masking. Hence all linear operations can be implemented with *the same complexity as Boolean masking*. At the end of the linear operations, when a multiplication has to be done, each share will be re-encoded to form a new share vector corresponding to a codeword of \mathcal{C} .

This method can be adapted for a masking scheme using the codes \mathcal{C} of Table 2. At the end of the multiplication procedure, the shares of a vector of $\widehat{\mathcal{C}}$ are re-encoded into \mathcal{C}^* (instead of \mathcal{C}) to form a share vector of length $t+1$. This code is used during the linear operations and then a re-encoding is applied so as to fall back on a codeword of \mathcal{C} when another multiplication is needed.

4.4 Comparison with Other Masking Schemes

We summarize in Table 3 the cost of secure operations when we use our t^{th} -order masking schemes derived from the codes of Table 1 (resp. from the codes of Table 2) denoted by *our masking scheme* \mathbf{F}_2 (resp. *our masking scheme* \mathbf{F}_4) using the improvements proposed in subsections 4.2 and 4.3.

In this table, *rand* indicates the number of random elements to generate, *add* and *mult* correspond to the numbers of additions and multiplications in the finite field \mathbf{F}_q . By *mult with w* , we indicate the number of small multiplications with the constant $w \in \mathbf{F}_4$ which can be performed with a look-up table. We also

give the cost of masked operations for Boolean masking using the multiplication procedure described in [RP10] and for Shamir’s secret sharing scheme using the multiplication procedure of [GM11, PR11]. For the multiplication procedure (denoted by Mult. of share vectors) with Shamir’s scheme, the cost of polynomial evaluations may be lowered by using the discrete Fourier transform as described in [CPR12].

According to this table, our masking procedure is dramatically more efficient than Shamir’s secret sharing scheme. When n behaves as a linear function of t , our solution is asymptotically the most efficient since the secure multiplication procedure needs a number of field multiplications linear in t while a quadratic number is needed for the Boolean masking scheme. In the next section, we compare these methods for securing AES, with concrete parameters.

Table 3. Complexity of masked operations against t^{th} -order SCA

	Our Masking Scheme F2	Our Masking Scheme F4
Add. with a constant	1 <i>add</i>	
Add. of share vectors	$t + 1$ <i>add</i>	
Mult. with a constant	$t + 1$ <i>mult</i>	
Mult. of share vectors	$(n - 1) + k(t + 1)$ <i>rand</i> $[(t + 1)(L + n - 1)$ $+ 2n - 1]$ <i>add</i> n <i>mult</i>	$(n - 1) + k(t + 1)$ <i>rand</i> $[(t + 1)(L + n - 1)$ $+ 2n - 1]$ <i>add</i> n <i>mult</i> $(t + 1)L'$ <i>mult. with w</i>
Square of share vectors	$t + 1$ <i>squares</i>	

	Boolean Masking [RP10]	Shamir Masking [GM11, PR11]
Add. with a constant	1 <i>add</i>	$t + 1$ <i>add</i>
Add. of share vectors	$t + 1$ <i>add</i>	
Mult. with a constant	$t + 1$ <i>mult</i>	
Mult. of share vectors	$t(t + 1)/2$ <i>rand</i> $2t(t + 1)$ <i>add</i> $(t + 1)^2$ <i>mult</i>	$t(2t + 1)$ <i>rand</i> $t(2t^2 + 2t)$ <i>add</i> $(2t + 1)^2$ <i>mult</i> $2t + 1$ <i>polynomial</i> <i>evaluations:</i> $(2t + 1) \times (t^2 + t)$ <i>add</i> $(2t + 1) \times (t^2 + t)$ <i>mult with const</i>
Square of share vectors	$t + 1$ <i>squares</i>	

Remarks: n and k corresponds to the parameters of the $[n + 1, k + 1]$ codes given in Table 1 and 2. L and L' refer respectively to the number of additions and *low-cost multiplication with w* for *encoding* given in Table 1 and 2.

5 Application to AES

In this section, we apply our masking scheme to design a secure implementation of AES against t^{th} -order SCA and compare its performance with Boolean masking [RP10] and Shamir's secret sharing scheme [GM11, PR11].

The AES [FIP01] is a block cipher algorithm which operates on a 4×4 bytes state. The bytes are viewed as elements of $\mathbf{F}_{2^8} = \mathbf{F}_2[x]/(x^8+x^4+x^3+x+1)$. During encryption, four transformations are involved. *AddRoundKey* is an addition between the state and the round key, *SubBytes* is a nonlinear transformation, *ShiftRows* and *MixColumns* are linear transformations.

In the following, we describe the implementation of our t^{th} -order masking on all the AES transformations.

5.1 Secure Implementation of Linear AES Transformations

To secure the linear transformations (*AddRoundKey*, *ShiftRows*, *MixColumns*) against t^{th} -order SCA, we propose to apply Boolean masking. More precisely, we consider the $[t+2, t+1]$ parity check code C^* over \mathbf{F}_{2^8} constructed from the generator matrix given by (2). Each element of the state is shared into $t+1$ elements of \mathbf{F}_{2^8} with the *encoding* procedure described in subsection 3.2. Hence all linear AES transformations can be performed share by share as for Boolean masking. As a result the masking of this transformation is as efficient as the method of [RP10]. More precisely, *AddRoundKey* requires $16 \times (t+1)$ additions in \mathbf{F}_{2^8} and *MixColumns* requires $4 \times 15 \times (t+1)$ additions and $4 \times 4 \times (t+1)$ multiplications by the constant $\{0x02\}$ which can be performed by a look-up table.

5.2 Secure Implementation of *SubBytes* Transformation

The nonlinear *SubBytes* transformation is the composition of two functions: the nonlinear calculation of the inverse in \mathbf{F}_{2^8} and an affine transformation over \mathbf{F}_2 , denoted Af . To secure the computation of the inverse in \mathbf{F}_{2^8} , one uses the fact that this operation can be defined as $X \mapsto X^{254}$. As shown in [RP10], this exponentiation requires a lower bound of 4 multiplications and 7 squares over \mathbf{F}_{2^8} :

$$X^{254} = [(X^2X)^4(X^2X)]^{16}(X^2X)^4X^2. \quad (3)$$

At the beginning of the *SubBytes* Transformation, we apply the code switching method of subsection 4.3. More precisely, each coordinate of the share vectors of the short code C^* are re-encoded in a code \mathcal{C} selected from Table 1 or 2. Then, the entire secure inversion is performed in this code with formula (3).

For the codes of Table 1, the square procedure require n squares performed share by share. Secure multiplications are done with Algorithm 4. For the codes of Table 2, the map $X \mapsto X^4$ can be computed efficiently share by share. The square procedure now needs a re-encoding procedure as described in Remark 2. For this reason, with these codes, we perform the first operation (*i.e.*, the

computation of the share vector of X^2) with the short parity check code C^* before switching code to \mathcal{C} .

Finally, we need to apply the affine transformation Af which can be decomposed as $X \mapsto A(X) + b$ where A is linear over \mathbf{F}_2 and b is a constant byte. The function A being linear over \mathbf{F}_2 (and not \mathbf{F}_{2^8}), we propose to compute this step by switching code to the parity check code C^* . Hence as for Boolean masking (cf [RP10]), this transformation requires roughly only $t + 1$ times the cost of a single A evaluation which is performed by using a look-up table.

5.3 Implementation Results and Comparisons

In order to give an idea of the global complexity of our masking scheme, we give in Table 4 estimations of the number of cycles needed for different implementations of a t^{th} -order masking of the AES Sbox for $t \in \{1, \dots, 6\}$. These estimations are based on 8051 assembly language with a 8-bit smartcard CPU. In such an environment, generation of a random byte requires 2 cycles, an addition requires 1 cycle, a secure multiplication over \mathbf{F}_{2^8} implemented by using so-called log/alog tables (see for instance [DR02]) requires roughly 20 cycles, and an access to a look-up table (describing the square operations, the multiplication with w and the affine transformation) requires 3 cycles. In particular the magnitude of complexity estimations given in Table 4 has been confirmed by a real implementation of our proposal, with the two first codes of Table 1, to design a first and second-order secure implementation.

Table 4. Estimation of timing for a secure AES Sbox on an 8-bit smartcard (in thousands of cycles)

Scheme \ Order t	1	2	3	4	5	6
Boolean masking [RP10]	0.4	0.9	1.5	2.4	3.4	4.6
Shamir [GM11, PR11]	1.3	4.8	11.6	22.7	39.3	62.3
Our masking scheme \mathbf{F}_2	0.8	1.1	3.8	4.3	5.3	6.2
Our masking scheme \mathbf{F}_4	0.5	-	2.2	2.9	5.7	-

From Table 4, we can see that, as t grows, our approach becomes more and more efficient than the method proposed in [GM11, PR11] using Shamir’s secret sharing scheme. Furthermore we can remark that the cost of our method is very close to the method using Boolean masking described in [RP10]. However a security flaw in this method has been very recently announced in [CPRR13] where the authors proposed a new solution. For future work, it will be interesting to compare our proposal to this solution and to see if some ideas using the framework of error correcting codes introduced in our work can be adapted to this solution to devise a more efficient masking of AES.

6 Conclusion

In this paper, we have presented a new high-order masking scheme and an application to the AES cipher. The masking scheme relies on secret sharing based on carefully chosen non-MDS linear codes and is significantly more efficient than the methods that rely on Shamir's secret sharing scheme. As a result, when applied to the secure implementation of AES, our masking scheme is a more attractive alternative to Boolean masking than Shamir's scheme. Moreover, the comparison given by Table 4 shows that the efficiency of our proposal is very close to Boolean masking and it could open new perspectives in masking scheme design.

References

- [BFGV12] Balasch, J., Faust, S., Gierlichs, B., Verbauwhede, I.: Theory and Practice of a Leakage Resilient Masking Scheme. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 758–775. Springer, Heidelberg (2012)
- [BMK04] 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)
- [BOGW88] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems For Non-Cryptographic Fault-Tolerant Distributed Computation. In: Symposium on Theory of Computing, pp. 1–10 (1988)
- [CC06] Chen, H., Cramer, R.: Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computations over Small Fields. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 521–536. Springer, Heidelberg (2006)
- [CCD88] Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Symposium on Theory of Computing, pp. 11–19 (1988)
- [CCG⁺07] 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)
- [CDG⁺08] Cramer, R., Daza, V., Gracia, I., Urroz, J.J., Leander, G., Martí-Farré, J., Padró, C.: On Codes, Matroids, and Secure Multiparty Computation from Linear Secret-Sharing Schemes. *IEEE Transactions on Information Theory* 54(6), 2644–2657 (2008)
- [CDM00] Cramer, R., Damgård, I.B., Maurer, U.M.: General Secure Multi-party Computation from any Linear Secret-Sharing Scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
- [CG13] Carlet, C., Guilley, S.: Side-channel indistinguishability. In: HASP 2013 Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy. ACM, New York (2013)
- [CGKS12] Carlet, C., Gaborit, P., Kim, J.-L., Solé, P.: A New Class of Codes for Boolean Masking of Cryptographic Computations. *IEEE Transactions on Information Theory* 58(9), 6000–6011 (2012)

- [CPR12] Coron, J.-S., Prouff, E., Roche, T.: On the Use of Shamir's Secret Sharing against Side-Channel Analysis. In: Mangard, S. (ed.) CARDIS 2012. LNCS, vol. 7771, pp. 77–90. Springer, Heidelberg (2013)
- [CPRR13] Coron, J.-S., Prouff, E., Rivain, M., Roche, T.: Higher-Order Side Channel Security and Mask Refreshing. In: Fast Software Encryption – FSE 2013 (2013)
- [CS90] Conway, J.H., Sloane, N.J.A.: A new upper bound on the minimal distance of self-dual codes. *IEEE Transactions on Information Theory* 36(6), 1319–1333 (1990)
- [DF12] Dziembowski, S., Faust, S.: Leakage-Resilient Circuits without Computational Assumptions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 230–247. Springer, Heidelberg (2012)
- [DR02] Daemen, J., Rijmen, V.: *The Design of Rijndael*. Springer (2002)
- [FIP01] FIPS PUB 197. Advanced Encryption Standard. National Institute of Standards and Technology (November 2001)
- [FMPR10] Fumaroli, G., Martinelli, A., Prouff, E., Rivain, M.: Affine masking against higher-order side channel analysis. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 262–280. Springer, Heidelberg (2011)
- [GM11] Goubin, L., Martinelli, A.: Protecting AES with Shamir's Secret Sharing Scheme. In: Preneel, Takagi (eds.) [PT11], pp. 79–94
- [GO03] Gaborit, P., Otmani, A.: *Experimental Constructions Of Self-Dual Codes. Finite Fields and Their Applications-Elsevier* (July 2003)
- [GRR98] Gennaro, R., Rabin, M., Rabin, T.: Simplified vss and fact-track multiparty computations with applications to threshold cryptography. In: *Symposium on Principles of Distributed Computing*, pp. 101–111 (1998)
- [ISW03] Ishai, Y., Sahai, A., Wagner, D.: Private Circuits: Securing Hardware against Probing Attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
- [JPS05] Joye, M., Paillier, P., Schoenmakers, B.: On Second-order Differential Power Analysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 293–308. Springer, Heidelberg (2005)
- [KHL11] Kim, H., Hong, S., Lim, J.: A Fast and Provably Secure Higher-Order Masking of AES S-Box. In: Preneel, Takagi (eds.) [PT11], pp. 95–107
- [KJJ99] Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
- [Koc96] 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)
- [Mas93] Massey, J.: Minimal Codewords and Secret Sharing. In: *Sixth Joint Swedish-Russian Workshop on Information Theory*, pp. 246–249 (1993)
- [Mes00] Messerges, T.S.: 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)
- [MS78] MacWilliams, F.J., Sloane, N.J.A.: *The Theory of Error-Correcting Codes*. North-holland Publishing Company (1978)
- [OMPR05] 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)

- [PR10] Prouff, E., Roche, T.: Attack on a Higher-Order Masking of the AES Based on Homographic Functions. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 262–281. Springer, Heidelberg (2010)
- [PR11] Prouff, E., Roche, T.: Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In: Preneel, Takagi (eds.) [PT11], pp. 63–78
- [PT11] Preneel, B., Takagi, T. (eds.): CHES 2011. LNCS, vol. 6917. Springer, Heidelberg (2011)
- [RP10] Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
- [Sha79] Shamir, A.: How to Share a Secret. CACM 22(11), 612–613 (1979)
- [TDG02] 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)

Hashing Mode Using a Lightweight Blockcipher

Hidenori Kuwakado¹ and Shoichi Hirose²

¹ Kansai University, 2-1-1 Ryozenji-cho, Takatsuki-shi, Osaka 569-1095 Japan

² University of Fukui, 3-9-1 Bunkyo, Fukui-shi, Fukui 910-8507, Japan
kuwakado@kansai-u.ac.jp, hrs_shch@u-fukui.ac.jp

Abstract. This article proposes a hashing mode using a lightweight blockcipher. Since the block size of a lightweight blockcipher is small, the hashing mode uses a double-block-length compression function that consists of two Matyas-Meyer-Oseas (MMO) modes. Tag-based applications often require a hash function to be a one-way function and a primitive for constructing a pseudorandom function. We analyze the one-wayness of the hashing mode and the pseudorandomness of the keyed hashing mode under standard assumptions of an underlying blockcipher. The analysis in the standard model is practically more significant than the analysis in the ideal-primitive model.

Keywords: hash function, computational security, preimage resistance, pseudorandom function, lightweight blockcipher.

1 Introduction

Background. Applications of RFID and sensor networks are likely to be deployed. In order to achieve secure communication in the applications, secure yet efficiently implementable cryptographic primitives are required, but standardized primitives such as AES [28] and SHA-2 [29] (and Keccak [5]) seem to be too expensive to implement in such a constrained device. Accordingly, lightweight secret-key ciphers and lightweight hash functions have been recently proposed.

Lightweight hash functions are classified into dedicated hash functions and blockcipher-based hash functions. The former include PHOTON [16], QUARK

Table 1. Lightweight blockciphers (κ : key, n : block [bits])

	Supported size (κ, n)
KATAN [11]	(80, 32), (80, 48), (80, 64)
KLEIN [15]	(64, 64), (80, 64), (96, 64)
LBlock [34]	(80, 64)
LED [17]	(64 + 4 * i , 64), $i = 0, 1, \dots, 16$
Piccolo [31]	(80, 64), (128, 64)
PRINCE [10]	(128, 64)
PRESENT [7]	(80, 64), (128, 64)
TWINE [33]	(80, 64), (128, 64)

[1], and SPONGENT [8]. The circuit size of the latter is expected to be smaller than that of the former if a lightweight blockcipher is implemented in the same device. Since the block size of lightweight blockciphers is usually 64 bits (Table 1), blockcipher-based hash functions inevitably use double-block-length compression functions. For example, the blockcipher-based hash function using PRESENT has been reported [9].

This article focuses on hash functions for resource-constrained devices such as an RFID tag. Bogdanov *et al.* [9] have identified five issues when a hash function is used in RFID tag-based applications. The five issues are briefly summarized below.

1. In tag-based applications, we are unlikely to hash large amounts of data. The typical input is usually much less than 256 bits.
2. In many tag-based applications we do not need the property of collision resistance. Most often the security of protocols depends on the one-way property.
3. Applications will only require moderate security levels. Consequently 80-bit security, or even less, may be adequate.
4. While the physical space for an implementation is often the primary consideration, the peak and average power consumption are also important.
5. Some protocols use a hash function to build a message authentication code (MAC), often by appealing to the HMAC construction [30].

From the second and the last item, we observe that a lightweight hash function is required to be a one-way function and a primitive for achieving a pseudorandom function.

The security of hash functions has been usually analyzed in an ideal-primitive model. The security of the sponge construction on which the above dedicated hash functions are based has been analyzed in the random-permutation model [3,4]. The security of blockcipher-based hash functions has been analyzed in the ideal-cipher model. The security proof in the ideal-primitive model suggests that the hash function has no structural weakness. However, when the ideal primitive is instantiated with a practical primitive, the proof does not necessarily guarantee that the resulting hash function achieves the expected security. Hence, the security proof in a standard model (e.g., computational assumptions) is generally preferable.

Contribution. This article describes a hashing mode using a double-block-length compression function that consists of two Matyas-Meyer-Oseas (MMO) modes of an underlying blockcipher. On the other hand, previous double-block-length compression functions such as Abreast-DM consist of two Davies-Meyer (DM) modes. When a lightweight blockcipher with a 80-bit key and a 64-bit block is used (Table 1), the previous double-block-length compression functions are inefficient because the size of a message block is 16 bits. The size of a message block in our double-block-length compression function is 64 bits. In addition, the hashing mode does not take a fixed initial value to improve the efficiency. The hashing mode can be used in a pseudorandom function by assigning a key to the first message block (the prf mode). The prf mode is slightly more efficient than the HMAC construction because the hashing mode is invoked only once.

We prove the preimage resistance of the hashing mode and the pseudorandomness of the prf mode under standard assumptions. The standard assumptions are that (i) an underlying blockcipher is a pseudorandom permutation against a related-key attack, (ii) an underlying blockcipher is secure against key-recovery attacks in the known-plaintext model, and (iii) the number of keys that give the same plaintext-ciphertext pair is small.

Related Works. MDC-2 and MDC-4 are double-block-length hash functions that have been standardized by ISO/IEC [21]. Since the work of Steinberger [32], their preimage resistance and their collision resistance have been rapidly analyzed in the ideal-cipher model [13,20,23,26]. Tandem-DM, Abreast-DM [24], and Hirose’s scheme [18] are well-known double-block-length compression functions, and their preimage resistance and their collision resistance have been analyzed in the ideal-cipher model. In particular, “super query” proposed by Lee *et al.* [25] is a significant technique for analyzing the preimage resistance of these compression functions. A key derivation function proposed by Jonsson and Robshaw [22] can be considered as a double-block-length compression function. The security of the key derivation function was analyzed in the ideal-cipher model. Unlike Tandem-DM, Abreast-DM, and Hirose’s scheme, the key derivation function consists of two MMO modes. Namely, the key derivation function by Jonsson and Robshaw is similar to the compression function of our hashing mode.

While the preimage resistance and the collision resistance have been of major concern to the security of hash functions, the indistinguishability from a random oracle has been recently discussed as an important property of hash functions since the seminal work of Coron *et al.* [12]. Weimar-DM proposed by Fleischmann *et al.* [14] has been analyzed in the ideal-cipher model. Naito [27] has proposed a double-block-length hash function with a post-processing function and proved that the hash function is indistinguishable from a random oracle up to the birthday bound of the hash length. The post-processing function is similar to the post-processing function of our hashing mode.

It should be noted that all the above double-block-length compression functions are inefficient if the key size of the underlying blockcipher is smaller than twice of the block size. Table 1 suggests that some lightweight blockciphers fall into this case. Furthermore, the security of all the above hash functions has been proved only in the ideal-cipher model. There are few works on preimage resistance in the standard model. Hirose *et al.* [19] discussed the preimage resistance of single-block-length compression functions under the computational assumption. However, their scheme requires an unconventional padding.

Organization. Section 2 proposes a new hashing mode and its prf mode. Section 3 describes definitions of security against attacks. Section 4 analyzes the preimage resistance of the hashing mode under standard assumptions of an underlying blockcipher. Section 5 summarizes the pseudorandomness of the prf mode under computational assumptions of an underlying blockcipher. Section 6 concludes this paper.

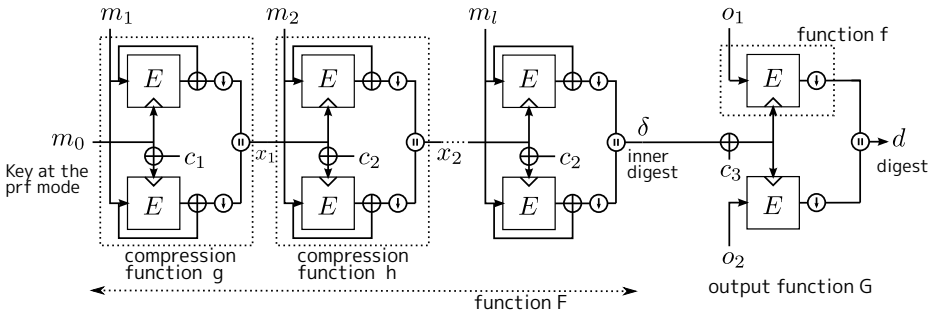


Fig. 1. Proposed hashing mode H

2 Construction

Hashing Mode. Let E be a blockcipher from $\{0, 1\}^\kappa \times \{0, 1\}^n$ to $\{0, 1\}^n$ where $n < \kappa < 2n$. A function f is defined as

$$f(x, z) = \text{tr}(E(x, z)) \tag{1}$$

where tr denotes truncation from $\{0, 1\}^n$ to $\{0, 1\}^{\kappa/2}$ (say, the first $\kappa/2$ bits).

A proposed hashing mode H is a function from $\{0, 1\}^*$ to $\{0, 1\}^\kappa$ (Fig. 1). Let m be an ℓ -bit message to be hashed. Padding is performed as follows: a single ‘1’ bit is appended to m , and then 0^t is appended where

$$t = \begin{cases} n + \kappa - (\ell + 1) & \text{if } \ell \leq \kappa - 1 \\ n - (\ell + 1 - \kappa \bmod n) & \text{if } \ell \geq \kappa. \end{cases}$$

The first κ bits of the padded message is assigned to the first message block m_0 and the remaining string is divided into n -bit message blocks m_i ($i = 1, 2, \dots, l$). Note that m_1 is 0^n if $\ell \leq \kappa - 1$. The index of the last message block, l , is given by the following formula.¹

$$l = \text{idxH}(\ell) = \begin{cases} 1 & \text{if } \ell \leq \kappa - 1 \\ \lceil \frac{\ell + 1 - \kappa}{n} \rceil & \text{if } \ell \geq \kappa. \end{cases}$$

Compression functions g, h from $\{0, 1\}^\kappa \times \{0, 1\}^n$ to $\{0, 1\}^\kappa$, which only differ in constants, are defined as follows:

$$g(x, z) = \text{tr}(E(x, z) \oplus z) \parallel \text{tr}(E(x \oplus c_1, z) \oplus z) \tag{2}$$

$$= f(x, z) \oplus \text{tr}(z) \parallel f(x \oplus c_1, z) \oplus \text{tr}(z)$$

$$h(x, z) = \text{tr}(E(x, z) \oplus z) \parallel \text{tr}(E(x \oplus c_2, z) \oplus z) \tag{3}$$

$$= f(x, z) \oplus \text{tr}(z) \parallel f(x \oplus c_2, z) \oplus \text{tr}(z)$$

¹ The last index l is not the number of message blocks because the index of message blocks begins with 0.

where each c_i is a distinct non-zero constant and \parallel denotes the concatenation operator on strings. Compute $x_1 = g(m_0, m_1)$. For $i = 2, 3, \dots, l$, compute a *chaining value* x_i as $x_i = h(x_{i-1}, m_i)$. The value of x_l , denoted by δ , is called an *inner digest*. To simplify the notation, g followed by the cascade of h is denoted by F , namely,

$$\begin{aligned} \delta &= h(\dots h(h(g(m_0, m_1), m_2), m_3) \dots, m_l) \\ &= F(m), \end{aligned}$$

where m denotes the (unpadded) message. An output function (a post-processing function) G from $\{0, 1\}^\kappa$ to $\{0, 1\}^\kappa$ is defined as

$$\begin{aligned} G(x) &= \text{tr}(E(x \oplus c_3, o_1)) \parallel \text{tr}(E(x \oplus c_3, o_2)) \\ &= f(x \oplus c_3, o_1) \parallel f(x \oplus c_3, o_2) \end{aligned} \tag{4}$$

where each o_i is a distinct constant and c_3 is a non-zero constant that is different from c_1 and c_2 . A *digest* d is given by $d = G(\delta) = G \circ F(m) = H(m)$ where m denotes the (unpadded) message.

PRF Mode. The hashing mode H is usable as a keyed function $\{0, 1\}^\kappa \times \{0, 1\}^* \rightarrow \{0, 1\}^\kappa$ by assigning a key to the first message block m_0 . We call the keyed function a *prf mode of H* and denote it by $\hat{H}_k(m)$ for a κ -bit key k and a message m . Given an ℓ -bit message m , message blocks m_1, m_2, \dots, m_l are produced in a way similar to that of H except for m_0 . Namely, a single ‘1’ bit is appended to m , and then $0^{t'}$ is appended where $t' = n - (\ell + 1 \bmod n)$. The padded message is divided into n -bit message blocks m_i ($i = 1, 2, \dots, l$). The index of the last message block, l , is given by the following formula.²

$$l = \text{idxPRF}(\ell) = \left\lceil \frac{\ell + 1}{n} \right\rceil$$

An output is computed as

$$\begin{aligned} \hat{H}_k(m) &= G \circ F(k \parallel m) \\ &= G(h(\dots h(h(g(k, m_1), m_2), m_3) \dots, m_l)). \end{aligned}$$

3 Security Definition

Section 4 will prove that the security of the proposed hashing mode H is reduced to that of an underlying blockcipher E . This section describes definitions of security on hash functions and blockciphers. Let $\{0, 1\}^n$ be the set of all n -bit binary strings and $\{0, 1\}^{\leq \ell}$ is defined as

$$\{0, 1\}^{\leq \ell} = \cup_{i=1}^{\ell} \{0, 1\}^i.$$

² In this case, the last index l is equal to the number of message blocks.

The concatenation of m_i, m_{i+1}, \dots, m_j is denoted by $m_{[i,j]}$, that is, $m_{[i,j]} = m_i \parallel m_{i+1} \parallel \dots \parallel m_j$. When S is a probabilistic space, $s \leftarrow S$ denotes the operation of selecting s according to the distribution specified by S . Unless otherwise specified, the distribution specified by S is the uniform distribution on S . We often say that s is *uniformly selected from S* . If S is not a set, then $s \leftarrow S$ is an assignment statement.

3.1 Preimage Resistance

Let H be a hash function from $\{0, 1\}^*$ to $\{0, 1\}^\kappa$. Let A be an adversary that finds a preimage of a given digest d for H . The advantage of A against H is defined by

$$\mathbf{Adv}_H^{\text{pre}[\ell, \check{\ell}]}(A) = \Pr [m \leftarrow \{0, 1\}^{\leq \ell}; d \leftarrow H(m); \hat{m} \leftarrow A(d, \ell) : H(\hat{m}) = d] \quad (5)$$

where the length of \hat{m} is at most $\check{\ell}$. The adversary A is called a *pre $[\ell, \check{\ell}]$ -adversary* against H . The definition above is different from that of the preimage resistance in previous works. In the previous works, a digest d is uniformly selected from the digest space $\{0, 1\}^\kappa$ or is arbitrarily selected by an adversary before making any queries [25]. In order to discuss the security in practical use, we define the preimage resistance as Eq. (5), which is similar to the definition of a one-way function given in article [6].

3.2 Pseudorandom Function

Let $\text{Func}(a, b)$ be the set of all the functions from $\{0, 1\}^a$ to $\{0, 1\}^b$. When a domain is $\{0, 1\}^{\leq a}$, the set is denoted by $\text{Func}(\leq a, b)$. A function λ that is uniformly selected from $\text{Func}(a, b)$ is called a *random function in $\text{Func}(a, b)$* . In this paper, a random function always works by lazy evaluation. Namely, on any query in $\{0, 1\}^a$, the random function returns an element selected from $\{0, 1\}^b$ uniformly as a response subject to the restriction that if the same query is given, the response is the same. Let $\text{Func}(\kappa + a, b)$ be the set of all the keyed functions from $\{0, 1\}^\kappa \times \{0, 1\}^a$ to $\{0, 1\}^b$ where $\{0, 1\}^\kappa$ is a key space. A keyed function $f(k, x) \in \text{Func}(\kappa + a, b)$ is often denoted by $f_k(x)$ (particularly, when k is fixed). Suppose that k is uniformly selected from $\{0, 1\}^\kappa$ and an oracle V is either f_k or λ . Let A be an adversary that makes queries to the oracle V and outputs 0 or 1. The advantage of A against f is defined as

$$\mathbf{Adv}_f^{\text{prf}}(A) = |\Pr [k \leftarrow \{0, 1\}^\kappa : A^{f_k} = 1] - \Pr [\lambda \leftarrow \text{Func}(a, b) : A^\lambda = 1]|. \quad (6)$$

The adversary A is called a *prf-adversary* against f . If $\mathbf{Adv}_f^{\text{prf}}(A)$ is negligible for any efficient A , then f is called a *pseudorandom function*. When there is no confusion on k and λ , the right hand side of Eq. (6) is often denoted by $|\Pr [A^{f_k} = 1] - \Pr [A^\lambda = 1]|$.

In this paper, a related-key attack using a multi-oracle will be discussed. For a keyed function $f \in \text{Func}(\kappa + a, b)$, let $\langle f_{k_j} \rangle_{j=1}^{4q} = (f_{k_1}, f_{k_2}, \dots, f_{k_{4q}})$ where each k_j is defined as

$$k_j \leftarrow \begin{cases} \{0, 1\}^\kappa & \text{if } j \bmod 4 = 1 \\ k_{j-1} \oplus c_1 & \text{if } j \bmod 4 = 2 \\ k_{j-2} \oplus c_2 & \text{if } j \bmod 4 = 3 \\ k_{j-3} \oplus c_3 & \text{if } j \bmod 4 = 0, \end{cases} \quad (7)$$

and each c_i is the constant defined in Sect. 2. In general, related-key attacks depend on related-key-deriving functions [2]. The related-key-deriving function we consider in this paper is the identical function and three XORing functions given by Eq. (7). Namely, an adversary knows constants c_i , but the adversary is not allowed to change them. Let ρ be a family of 2^κ independent random functions in $\text{Func}(a, b)$, called a *keyed random function in $\text{Func}(\kappa + a, b)$* . Note that λ_{k_j} and $\lambda_{k_{j'}}$ are independent random functions in $\text{Func}(a, b)$ even if $k_{j'}$ is a related-key of k_j such as Eq. (7). The advantage of A against f under the related-key attack using $4q$ oracles is defined as

$$\text{Adv}_f^{4q\text{-prf-rka}}(A) = \left| \Pr \left[A^{\langle f_{k_j} \rangle_{j=1}^{4q}} = 1 \right] - \Pr \left[A^{\langle \lambda_{k_j} \rangle_{j=1}^{4q}} = 1 \right] \right|,$$

where each k_j is determined as Eq. (7). The adversary A is called a *4q-prf-rka-adversary* against f .

3.3 Pseudorandom Permutation

Let $\text{Perm}(a)$ be the set of all the permutations on $\{0, 1\}^a$. A permutation π that is uniformly selected from $\text{Perm}(a)$ is called a *random permutation in $\text{Perm}(a)$* . In this paper, a random permutation always works by lazy evaluation. Let $\text{Perm}(\kappa, a)$ be the set of all the keyed permutations from $\{0, 1\}^\kappa \times \{0, 1\}^a$ to $\{0, 1\}^a$ where $\{0, 1\}^\kappa$ is a key space. The set $\text{Perm}(\kappa, a)$ is identical to the set of all the blockciphers with key length κ and block length a . A keyed permutation $p(k, x) \in \text{Perm}(\kappa, a)$ is often denoted by $p_k(x)$. Suppose that k is uniformly selected from $\{0, 1\}^\kappa$ and an oracle V is either p_k or π . Let A be an adversary that makes queries to the oracle V and outputs 0 or 1. The advantage of A against p is defined as

$$\text{Adv}_p^{\text{prp}}(A) = |\Pr [k \leftarrow \{0, 1\}^\kappa : A^{p_k} = 1] - \Pr [\pi \leftarrow \text{Perm}(a) : A^\pi = 1]|$$

The adversary A is called a *prp-adversary* against p . If $\text{Adv}_f^{\text{prp}}(A)$ is negligible for any efficient A , then p is called a *pseudorandom permutation*. When there is no confusion on k and π , the right hand side of Eq. (6) is often denoted by $|\Pr [A^{p_k} = 1] - \Pr [A^\pi = 1]|$, for short. A *4q-prp-rka-adversary* is defined in a manner similar to *4q-prf-rka-adversary*. The advantage of the adversary against p is done as

$$\text{Adv}_p^{4q\text{-prp-rka}}(A) = \left| \Pr \left[A^{\langle p_{k_j} \rangle_{j=1}^{4q}} = 1 \right] - \Pr \left[A^{\langle \pi_{k_j} \rangle_{j=1}^{4q}} = 1 \right] \right|,$$

where each k_j is determined as Eq. (7) and π is a keyed random permutation (i.e., an ideal cipher).

In this paper, a key recovery attack in the known-plaintext model will be discussed. Let A be an adversary that finds a key from q pairs of a plaintext x_j and its ciphertext y_j , denoted by $\langle x_j, y_j \rangle_{j=1}^q$. The adversary A cannot choose x_j because this attack is a known-plaintext attack. The advantage of A against p is defined as

$$\text{Adv}_p^{q\text{-kr-kp}}(A) = \Pr \left[\langle x_j, y_j \rangle_{j=1}^q \leftarrow \text{Enc}^p(x_j); \hat{k} \leftarrow A(\langle x_j, y_j \rangle_{j=1}^q) : p(\hat{k}, x_j) = y_j, 1 \leq \forall j \leq q \right],$$

where $\text{Enc}^p(x_j)$ denotes $k \leftarrow \{0, 1\}^\kappa; y_j \leftarrow p(k, x_j), 1 \leq j \leq q$. The recovered key \hat{k} may not be equal to k . The adversary A is called a q -kr-kp-adversary against p . This paper only discusses the following advantage of A .

$$\text{Adv}_p^{2\text{-kr-kp}}(A) = \Pr \left[\langle o_j, y_j \rangle_{j=1}^2 \leftarrow \text{Enc}^p(o_j); \hat{k} \leftarrow A(\langle x_j, y_j \rangle_{j=1}^2) : p(\hat{k}, o_j) = y_j, j = 1, 2 \right],$$

where each o_j is defined in Section 2.

4 Preimage Resistance

4.1 Main Theorem

Let $\langle x_i, y_i \rangle_{i=1}^q$ be a set of q pairs (x_i, y_i) where each x_i is distinct and $y_i = E_k(x_i)$. We say that a key \hat{k} is *consistent* with $\langle x_i, y_i \rangle_{i=1}^q$ if $y_i = E_{\hat{k}}(x_i)$ for $i = 1, 2, \dots, q$. The set of all the keys consistent with $\langle x_i, y_i \rangle_{i=1}^q$ is denoted by

$$\text{Cons}_{E_k}(\langle x_i, y_i \rangle_{i=1}^q).$$

The number of elements in this set, denoted by $\#\text{Cons}_{E_k}(\langle x_i, y_i \rangle_{i=1}^q)$, may not be one. Let o_1 and o_2 be constants defined in Sect. 2. We denote by ϕ_E the average number of elements in $\text{Cons}_{E_k}(\langle o_i, y_i \rangle_{i=1}^2)$, that is,

$$\phi_E = \frac{1}{2^\kappa} \sum_{k \in \{0,1\}^\kappa} \#\text{Cons}_{E_k}(\langle o_i, y_i \rangle_{i=1}^2).$$

Similarly, we define $\phi_{\text{tr} \circ E}$ as

$$\phi_{\text{tr} \circ E} = \frac{1}{2^\kappa} \sum_{k \in \{0,1\}^\kappa} \#\text{Cons}_{\text{tr} \circ E_k}(\langle o_i, d_i \rangle_{i=1}^2),$$

where $d_i = \text{tr}(E_k(o_i)) = f(k, o_i)$.

Theorem 1 means that the preimage resistance of H can be reduced to the security of the underlying blockcipher E .

Theorem 1. *Suppose that $n < \kappa \leq 2n$. Let A be a $\text{pre}[\ell, \check{\ell}]$ -adversary against H that runs in time τ_A . Then, there exists a 4-prp-rka-adversary B against E and a 2-kr-kp-adversary C against E such that*

$$\mathbf{Adv}_H^{\text{pre}[\ell, \check{\ell}]}(A) \leq \text{id}\mathbf{xH}(\ell)\mathbf{Adv}_E^{4\text{-prp-rka}}(B) + \frac{\phi_{\text{tro}E}}{\phi_E}\mathbf{Adv}_E^{2\text{-kr-kp}}(C). \quad (8)$$

Let τ_E be a time for the encryption of E . The adversary B makes two queries and the running time is $\tau_A + O((\text{id}\mathbf{xH}(\ell) + \text{id}\mathbf{xH}(\check{\ell}))\tau_E)$. The adversary C runs in time $\tau_A + O(\text{id}\mathbf{xH}(\check{\ell})\tau_E)$.

As mentioned in Sect. 1, the input length in RFID tag-based applications is usually less than 256 bits. The value of $\text{id}\mathbf{xH}(\ell)$ does not have a major effect on the security bound. The preimage resistance of H also depends on the value of $\phi_{\text{tro}E}/\phi_E$, which is a statistical property of E .

4.2 Proof of Theorem 1 and Lemmas

The proof of Theorem 1 requires Lemma 1 and Lemma 2 that are proved in Appendix. Let $G \circ \$$ be an operation as

$$G \circ \$: \delta \leftarrow \{0, 1\}^\kappa; d \leftarrow G(\delta).$$

Let $\mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \check{\ell}]-H}(A)$ be the advantage of A that is a $\text{pre}[\ell, \check{\ell}]$ -adversary against H when A is given a digest d produced with $G \circ \$$, that is,

$$\mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \check{\ell}]-H}(A) = \Pr[\delta \leftarrow \{0, 1\}^\kappa; d \leftarrow G(\delta); \hat{m} \leftarrow A(d, \ell) : H(\hat{m}) = d]. \quad (9)$$

Let \hat{F} be a keyed function as

$$\begin{aligned} \hat{F}_k(m) &= F(k \parallel m) \\ &= h(\dots h(h(g(k, m_1), m_2), m_3) \dots, m_l)). \end{aligned}$$

for a κ -bit key k and a message m . The algorithm of \hat{F} is the same as that of the prf mode of H excluding G .

Lemma 1. *Let A be a $\text{pre}[\ell, \check{\ell}]$ -adversary against H that runs in time τ_A . Then, there exists a prf-adversary A_1 against \hat{F} such that*

$$\left| \mathbf{Adv}_H^{\text{pre}[\ell, \check{\ell}]}(A) - \mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \check{\ell}]-H}(A) \right| \leq \mathbf{Adv}_{\hat{F}}^{\text{prf}}(A_1). \quad (10)$$

The adversary A_1 makes a single query and runs in time $\tau_A + O(\text{id}\mathbf{xH}(\check{\ell})\tau_E)$. The length of the single query is at most $\ell - \kappa$.

Since $H = G \circ F$, Lemma 1 shows that the difference between $G \circ F$ and $G \circ \$$ is determined by the pseudorandomness of \hat{F} . Lemma 2 shows that the pseudorandomness of \hat{F} is reduced to the security of an underlying blockcipher E against the related-key attack.

Lemma 2. *Let A_1 be a prf-adversary against \hat{F} that makes a single query and runs in time τ_{A_1} . The length of the single query is at most $\ell - \kappa$ bits. Then, there exists a 4-prp-rka-adversary B against E such that*

$$\mathbf{Adv}_{\hat{F}}^{\text{prf}}(A_1) \leq \text{id}\times\text{H}(\ell)\mathbf{Adv}_E^{4\text{-prp-rka}}(B). \tag{11}$$

The adversary B makes two queries and runs in time $\tau_{A_1} + O(\text{id}\times\text{H}(\ell)\tau_E)$.

Proof of Theorem 1. In Theorem 1, the goal of C is to find a key k from $\langle o_i, y_i \rangle_{i=1}^2$ where $y_i = E_k(o_i)$. Note that each o_i is defined in Sect. 2, that is, they are not chosen by C . Consider C that uses the $\text{pre}[\ell, \tilde{\ell}]$ -adversary A as a subroutine. Then, C has to produce a digest d given to A . The algorithm of C is described below.

1. Produce a digest d as $d = d_1 \parallel d_2$ where $d_i = \text{tr}(y_i)$.
2. Run A given d . Let m be the output of A .
3. Compute an inner digest δ as $\delta = F(m)$.
4. If $E(\delta \oplus c_3, o_1) = y_1$ and $E(\delta \oplus c_3, o_2) = y_2$, then output $\delta \oplus c_3$ as a key. Otherwise output a κ -bit random string r .

The running time of C is $\tau_A + O(\text{id}\times\text{H}(\tilde{\ell})\tau_E) + O(\tau_E)$.

We discuss the distribution of the digest d in step 1. Step 1 is equivalently written as

$$\begin{aligned} d &= \text{tr}(E_k(o_1)) \parallel \text{tr}(E_k(o_2)) \\ &= G(k \oplus c_3), \end{aligned}$$

where G is defined by Eq. (4) and k is unknown to C . We cannot say that the distribution of d in step 1 is the same as that of $H(m) = G \circ F(m)$ where m is uniformly selected from $\{0, 1\}^{\leq \ell}$ because the distribution of $k \oplus c_3$ is not necessarily the same as that of $F(m)$ when m is uniformly selected from $\{0, 1\}^{\leq \ell}$. However, since k is uniformly selected from $\{0, 1\}^\kappa$, the distribution of d is the same as that of $G \circ \$$.

Assume that A succeeds in finding m such that $H(m) = d$ in step 2. Equation (9), $\mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \tilde{\ell}]\text{-}H}(A)$, denotes the probability that the assumption holds. Owing to the assumption, $\delta \oplus c_3$ in step 4 is consistent with $\langle o_i, d_i \rangle_{i=1}^2$, but $\delta \oplus c_3$ is not necessarily consistent with $\langle o_i, y_i \rangle_{i=1}^2$ because A does not know the $(n - \kappa/2)$ -bit information of y_i . Consequently, the probability that two equations in step 4 holds is given by $\phi_E / \phi_{\text{tr} \circ E}$. Summarizing the discussion above gives

$$\begin{aligned} \mathbf{Adv}_E^{2\text{-kr-kp}}(C) &= \mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \tilde{\ell}]\text{-}H}(A) \cdot \frac{\phi_E}{\phi_{\text{tr} \circ E}} \\ &+ \left(1 - \mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \tilde{\ell}]\text{-}H}(A) \cdot \frac{\phi_E}{\phi_{\text{tr} \circ E}} \right) \frac{\#\text{Cons}_{E_k}(\langle o_i, y_i \rangle_{i=1}^2)}{2^\kappa} \tag{12} \end{aligned}$$

$$\geq \mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \tilde{\ell}]\text{-}H}(A) \cdot \frac{\phi_E}{\phi_{\text{tr} \circ E}}. \tag{13}$$

Since the second term in Eq. (12) is the probability that r in step 4 is consistent with $\langle o_i, y_i \rangle_{i=1}^2$, we are not interested in the second term. If $\mathbf{Adv}_{G \circ \mathcal{S}}^{\text{pre}[\ell, \tilde{\ell}]-H}(A) \geq \mathbf{Adv}_H^{\text{pre}[\ell, \tilde{\ell}]}(A)$, then substituting it into Eq. (13) gives

$$\mathbf{Adv}_E^{2\text{-kr-kp}}(C) \geq \frac{\phi_E}{\phi_{\text{tr} \circ E}} \mathbf{Adv}_H^{\text{pre}[\ell, \tilde{\ell}]}(A).$$

If $\mathbf{Adv}_{G \circ \mathcal{S}}^{\text{pre}[\ell, \tilde{\ell}]-H}(A) < \mathbf{Adv}_H^{\text{pre}[\ell, \tilde{\ell}]}(A)$, then applying Eq. (10) of Lemma 1 to Eq. (13) yields

$$\mathbf{Adv}_E^{2\text{-kr-kp}}(C) \geq \frac{\phi_E}{\phi_{\text{tr} \circ E}} \left(\mathbf{Adv}_H^{\text{pre}[\ell, \tilde{\ell}]}(A) - \mathbf{Adv}_F^{\text{prf}}(A_1) \right).$$

Since $\mathbf{Adv}_F^{\text{prf}}(A_1) \geq 0$, the following inequality holds for both of cases.

$$\mathbf{Adv}_E^{2\text{-kr-kp}}(C) \geq \frac{\phi_E}{\phi_{\text{tr} \circ E}} \left(\mathbf{Adv}_H^{\text{pre}[\ell, \tilde{\ell}]}(A) - \mathbf{Adv}_F^{\text{prf}}(A_1) \right). \quad (14)$$

Substituting Eq. (11) of Lemma 2 into Eq. (14) yields

$$\mathbf{Adv}_E^{2\text{-kr-kp}}(C) \geq \frac{\phi_E}{\phi_{\text{tr} \circ E}} \left(\mathbf{Adv}_H^{\text{pre}[\ell, \tilde{\ell}]}(A) - \text{id} \times \text{H}(\ell) \mathbf{Adv}_E^{4\text{-prp-rka}}(B) \right).$$

The proof is completed. \square

5 Pseudorandom Function

5.1 Main Theorem

This section analyzes the pseudorandomness of the prf mode of H, \hat{H} , described in Sect. 2. Theorem 2 shows that the pseudorandomness of \hat{H} is reduced to the computational security of E .

Theorem 2. *Suppose that $n < \kappa \leq 2n - 5$. Let A be a prf-adversary against \hat{H} that runs in time τ_A and makes at most q queries. The length of a query by A is at most ℓ bits. Suppose that $q < 2^{\kappa/2}$. Let $l_{\max} = \text{id} \times \text{PRF}(\ell)$. Then, there exists a 4-prp-rka-adversary C against E such that*

$$\mathbf{Adv}_{\hat{H}}^{\text{prf}}(A) \leq (l_{\max} + 1) \left(q \mathbf{Adv}_E^{4\text{-prp-rka}}(C) + \frac{2^{\kappa/2}}{1 - \left(\frac{2e}{\theta}\right)} \left(\frac{2e}{\theta}\right)^\theta \right) \quad (15)$$

where e denotes the base of the natural logarithm and $\theta = 2^{n-\kappa/2} + 1$. The adversary C makes at most $2q$ queries and runs in time $\tau_A + O(q l_{\max} \tau_E)$ where τ_E denotes the time required to compute E .

The assumption that $\kappa \leq 2n - 5$ is required for being $2e/\theta < 1$. When $n = 64$, $\kappa = 80$, and $\kappa = 96$, the second term in the last parenthesis on the right hand side of Eq. (15) is approximated to $2^{-3.6 \times 10^8}$ and $2^{-8.8 \times 10^5}$, respectively. The second term is negligible for typical parameters of lightweight blockciphers. In tag-based applications, the value of l_{max} does not have a major effect on the security bound.

Owing to the assumption of $q < 2^{\kappa/2}$, Theorem 2 does not guarantee the pseudorandomness beyond the birthday bound of the output length κ . The assumption is required only to obtain the second term on the right hand side. Namely, the assumption is not required to obtain the first term. However, it is meaningless to remove the assumption from the second term for the following reason. Consider an adversary C that performs an exhaustive search for q random keys because C is allowed to spend time for computing E q times. The probability that C finds a correct key by the exhaustive search is at least $q/2^\kappa$, that is, $\text{Adv}_E^{4\text{-prp-rka}}(C) \geq q/2^\kappa$. It follows that the first term looks like $q^2/2^\kappa$. Accordingly, the pseudorandomness does not go beyond the birthday bound of the output length κ even if the assumption is removed.

5.2 Lemmas for Theorem 2

Combining Lemma 3 and Lemma 4, which are proved in the full paper, immediately gives Theorem 2

Lemma 3. *Let A be a prf-adversary against \hat{H} that runs in time τ_A and makes at most q queries. The length of a query by A is at most ℓ bits. Let $l_{max} = \text{idXPRF}(\ell)$. Then, there exists a $4q$ -prf-rka-adversary B against f , which is defined by Eq. (1), such that*

$$\text{Adv}_{\hat{H}}^{\text{prf}}(A) = (l_{max} + 1)\text{Adv}_f^{4q\text{-prf-rka}}(B). \quad (16)$$

The adversary B makes at most $2q$ queries and runs in time $\tau_A + O(ql_{max}\tau_E)$.

Lemma 4. *Suppose that $\kappa \leq 2n - 5$. Let B be a $4q$ -prf-rka-adversary against f that runs in time τ_B and makes at most $2q$ queries. Suppose that $q < 2^{\kappa/2}$. Then, there exists a 4 -prp-rka-adversary C against E such that*

$$\text{Adv}_f^{4q\text{-prf-rka}}(B) \leq q\text{Adv}_E^{4\text{-prp-rka}}(C) + \frac{2^{\kappa/2}}{1 - \left(\frac{2e}{\theta}\right)} \left(\frac{2e}{\theta}\right)^\theta, \quad (17)$$

where e denotes the base of the natural logarithm and $\theta = 2^{n-\kappa/2} + 1$. The adversary C makes at most $2q$ queries and runs in time $\tau_B + O(q\tau_E)$.

6 Concluding Remarks

This article has proposed a new hashing mode suitable for a lightweight blockcipher. When a lightweight blockcipher is implemented in the same device, the

hashing mode is expected to be more efficient than a dedicated hash function in terms of the circuit size. When we designed the hashing mode, we were desperately aware of issues pointed out by Bogdanov *et al.* Although the security of previous hash functions has been analyzed in the ideal-primitive model, the preimage resistance and the pseudorandom function of our hashing mode can be reduced to the standard security of an underlying blockcipher. Our analysis is practically more significant than analysis in the ideal-primitive model. In addition, the full paper will show the collision resistance of the hashing mode in the ideal-cipher model.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers 22560376, 25330150.

References

1. Aumasson, J.-P., Henzen, L., Meier, W., Naya-Plasencia, M.: QUARK: a lightweight hash. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 1–15. Springer, Heidelberg (2010)
2. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
3. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Sponge functions. In: Ecrypt Hash Workshop 2007 (2007), <http://sponge.noekeon.org/SpongeFunctions.pdf>
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indistinguishability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
5. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak sponge function family (2009), <http://keccak.noekeon.org/>
6. Black, J.A., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
7. Bogdanov, A.A., Knudsen, L.R., 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)
8. Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., Verbauwhede, I.: SPONGENT: A lightweight hash function. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 312–325. Springer, Heidelberg (2011)
9. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash functions and RFID tags: Mind the gap. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 283–299. Springer, Heidelberg (2008)
10. Borghoff, J., Canteaut, A., Güneş, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE - a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)

11. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
12. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
13. Fleischmann, E., Forler, C., Lucks, S., Wenzel, J.: The collision security of MDC-4. Cryptology ePrint Archive, Report 2012/096 (2012), <http://eprint.iacr.org/>
14. Fleischmann, E., Forler, C., Lucks, S., Wenzel, J.: Weimar-DM: A highly secure double-length compression function. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 152–165. Springer, Heidelberg (2012)
15. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A new family of lightweight block ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
16. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash functions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 222–239. Springer, Heidelberg (2011)
17. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. Cryptology ePrint Archive, Report 2012/600 (2012), <http://eprint.iacr.org/>
18. Hirose, S.: Some plausible constructions of double-block-length hash functions. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
19. Hirose, S., Kuwakado, H., Yoshida, H.: Compression functions using a dedicated blockcipher for lightweight hashing. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 346–364. Springer, Heidelberg (2012)
20. Hong, D., Kwon, D.: New preimage attack on MDC-4. Cryptology ePrint Archive, Report 2012/633 (2012), <http://eprint.iacr.org/>
21. ISO/IEC 10118-2:2010, Information technology – security techniques – hash-functions – part 2: Hash-functions using an n-bit block cipher (2010)
22. Jonsson, J., Robshaw, M.: Securing RSA-KEM via the AES. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 29–46. Springer, Heidelberg (2005)
23. Knudsen, L.R., Mendel, F., Rechberger, C., Thomsen, S.S.: Cryptanalysis of MDC-2. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 106–120. Springer, Heidelberg (2009)
24. Lai, X., Massey, J.L.: Hash functions based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
25. Lee, J., Stam, M., Steinberger, J.: The preimage security of double-block-length compression functions. Cryptology ePrint Archive, Report 2011/210 (2011), <http://eprint.iacr.org/>
26. Mennink, B.: On the collision and preimage security of MDC-4 in the ideal cipher model. Cryptology ePrint Archive, Report 2012/113 (2012), <http://eprint.iacr.org/>
27. Naito, Y.: Blockcipher-based double-length hash functions for pseudorandom oracles. Cryptology ePrint Archive, Report 2010/566 (2010), <http://eprint.iacr.org/>
28. National Institute of Standards and Technology, Advanced encryption standard (AES), Federal Information Processing Standards Publication 197 (2001), <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
29. National Institute of Standards and Technology, Secure hash standard, Federal Information Processing Standards Publication 180-2 (August 2002), <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>

30. National Institute of Standards and Technology, The keyed-hash message authentication code (HMAC), Federal Information Processing Standards Publication, FIPS PUB 198-1 (2008),
http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
31. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011)
32. Steinberger, J.P.: The collision intractability of MDC-2 in the ideal-cipher model. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 34–51. Springer, Heidelberg (2007)
33. Suzuki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: A lightweight, versatile block cipher. In: ECRYPT Workshop on Lightweight Cryptography 2011 (2011)
34. Wu, W., Zhang, L.: LBlock: A lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)

A Proof of Lemma 1

Lemma 1. *Let A be a $\text{pre}[\ell, \check{\ell}]$ -adversary against H that runs in time τ_A . Then, there exists a prf -adversary A_1 against \hat{F} such that*

$$\left| \text{Adv}_H^{\text{pre}[\ell, \check{\ell}]}(A) - \text{Adv}_{G \circ S}^{\text{pre}[\ell, \check{\ell}]-H}(A) \right| \leq \text{Adv}_{\hat{F}}^{\text{prf}}(A_1).$$

The adversary A_1 makes a single query and runs in time $\tau_A + O(\text{id}\check{\text{H}}(\check{\ell})\tau_E)$. The length of the single query is at most $\ell - \kappa$.

Proof. When the pseudorandomness of \hat{F} is discussed, a length-extension attack can be ignored because A_1 is not allowed to make two queries.

Let V be the oracle of A_1 . The oracle V is either \hat{F}_k or η where η is a random function in $\text{Func}(\leq \ell - \kappa, \kappa)$. The goal of A_1 is to identify V . Consider A_1 that uses the $\text{pre}[\ell, \check{\ell}]$ -adversary A against H as a subroutine. The algorithm of A_1 is described below.

1. $m \leftarrow \{0, 1\}^{\leq \ell - \kappa}$.
2. Make a query m to V . The answer is denoted by δ .
3. Compute $d = G(\delta)$. (G : Eq. (4))
4. Run A given d . The output of A is denoted by \hat{m} .
5. If $H(\hat{m}) = d$, then output 1. Otherwise output 0.

The number of queries by A_1 is one (i.e., step 2) and the length of the query is at most $\ell - \kappa$. Step 3 requires two encryptions of E . The time of step 4 is equal to that of A . Step 5 requires at most $2 \cdot \text{id}\check{\text{H}}(\check{\ell}) + 2$ encryptions of E because the length of \hat{m} is at most $\check{\ell}$. Accordingly, the running time of A_1 is $\tau_A + O((2\text{id}\check{\text{H}}(\check{\ell}) + 2)\tau_E)$.

If $V = \hat{F}_k$, then $\delta = \hat{F}_k(m)$ and $d = G(\delta) = G \circ F(k \parallel m) = H(k \parallel m)$ where the length of $k \parallel m$ is at most ℓ bits. If $V = \eta$, then $\delta \leftarrow \eta(m)$ and $d = G(\delta) = G \circ \eta(m)$. Since V is invoked only once, $\delta \leftarrow \eta(m)$ is equivalent to

$\delta \leftarrow \{0, 1\}^\kappa$. Hence, d can be regarded as the string produced with the operation $G \circ \$$. The probability that A_1 outputs 1 is given by

$$\Pr [A_1^{\hat{F}_k} = 1] = \mathbf{Adv}_H^{\text{pre}[\ell, \hat{\ell}]}(A), \quad \Pr [A_1^\eta = 1] = \mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \hat{\ell}]-H}(A).$$

Hence, we have

$$\begin{aligned} \mathbf{Adv}_F^{\text{prf}}(A_1) &= \left| \Pr [A_1^{F_k} = 1] - \Pr [A_1^\eta = 1] \right| \\ &= \left| \mathbf{Adv}_H^{\text{pre}[\ell, \hat{\ell}]}(A) - \mathbf{Adv}_{G \circ \$}^{\text{pre}[\ell, \hat{\ell}]-H}(A) \right|. \end{aligned}$$

Since there may exist a better adversary with the same resource as A_1 we considered here, we obtain the inequality of Eq. (10). \square

B Proof of Lemma 2

Lemma 2. *Let A_1 be a prf-adversary against \hat{F} that makes a single query and runs in time τ_{A_1} . The length of the single query is at most $\ell - \kappa$ bits. Then, there exists a 4-prp-rka-adversary B against E such that*

$$\mathbf{Adv}_F^{\text{prf}}(A_1) \leq \text{id}\times\text{H}(\ell) \mathbf{Adv}_E^{4\text{-prp-rka}}(B)$$

The adversary B makes two queries and runs in time $\tau_{A_1} + O(\text{id}\times\text{H}(\ell)\tau_E)$.

Proof. The oracle given to B is denoted by $\mathcal{V} = \langle V_{k_j} \rangle_{j=1}^4$. The oracle \mathcal{V} is $\mathcal{E} = \langle E_{k_j} \rangle_{j=1}^4$ or $\mathcal{P} = \langle \pi_{k_j} \rangle_{j=1}^4$ where each k_j is determined by Eq. (7) and π is a keyed random permutation in $\text{Perm}(\kappa, a)$. To make consistent with notation, we sometime denote k_1 by k . The goal of B is to identify \mathcal{V} . Consider B that uses the prf-adversary A_1 against \hat{F} as a subroutine. Then, B has to simulate the oracle of A_1 , which accepts an $(\ell - \kappa)$ -bit (or less) string and answers a κ -bit string.

To simulate the oracle of A_1 using \mathcal{V} , B defines a function $T^\mathcal{V}(v, m)$ as Fig. 2. In Fig. 2, η is a random function in $\text{Func}(\leq \ell - \kappa, \kappa)$ and h is the function of Eq. (3). In Line 102, message blocks $m_{[1, l]}$ are produced in the same way as \hat{F}_k . Proposition 1 described later shows that the following equations hold when $T^\mathcal{V}(v, m)$ is invoked only once.

$$T^\mathcal{E}(1, m) = \hat{F}_k(m) \tag{18}$$

$$T^\mathcal{E}(v, m) = T^\mathcal{P}(v - 1, m) \quad \text{for } 2 \leq v \leq l \tag{19}$$

$$T^\mathcal{P}(l, m) = \eta(m) \tag{20}$$

The above equations suggest that $T^\mathcal{V}(v, m)$ changes from $\hat{F}_k(m)$ to $\eta(m)$ as v increases. Note that A_1 is the adversary that distinguishes $\hat{F}_k(m)$ from $\eta(m)$.

The adversary B tries to identify \mathcal{V} using A_1 . The algorithm of B is described below.

```

101: function  $T^{\mathcal{V}}(v, m)$ 
102:   Produce  $m_{[1,l]}$  from  $m$ .
103:   if  $l \leq v - 1$  then
104:      $x_l \leftarrow \eta(m) \triangleright \eta$ : rand. func.
105:   else if  $l \geq v$  then
106:     if  $v = 1$  then
107:        $x_v \leftarrow \text{tr}(V_{k_1}(m_v) \oplus m_v) \parallel \text{tr}(V_{k_2}(m_v) \oplus m_v) \triangleright k_2 = k_1 \oplus c_1$ 
108:     else
109:        $x_v \leftarrow \text{tr}(V_{k_1}(m_v) \oplus m_v) \parallel \text{tr}(V_{k_3}(m_v) \oplus m_v) \triangleright k_3 = k_1 \oplus c_2$ 
110:     end if
111:     for  $i \leftarrow v + 1$  to  $l$  do
112:        $x_i \leftarrow h(x_{i-1}, m_i) \triangleright h$ : Eq. (3)
113:     end for
114:   end if
115:   return  $x_l$ 
116: end function

```

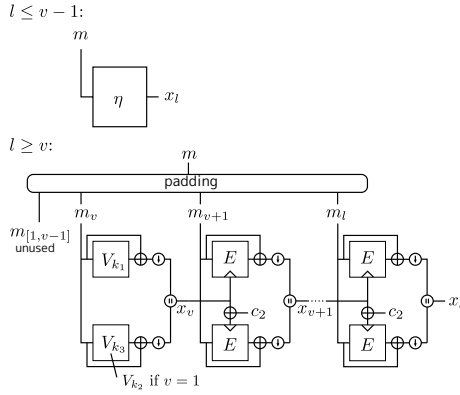


Fig. 2. Pseudocode of T and its diagram

1. Run A_1 that is allowed to make a single query m .
2. Compute $l = \text{idxF}(\text{len}(m))$ where $\text{len}(m)$ denotes the length of m .
3. $v \leftarrow \{1, 2, \dots, l\}$.
4. Compute $x_l = T^{\mathcal{V}}(v, m)$ according to Fig. 2.
5. Answer x_l to A_1 .
6. Output the output of A_1 , which is 0 or 1.

From the above algorithm, the number of queries by B to \mathcal{V} is two (Line 109), precisely one query to each V_i . The running time of B is dominated by the sum

of the running time of A_1 and time for encrypting m_i at most $2(\text{idxF}(\ell - \kappa) - 1)$ times (Line 112). The probability that B outputs 1 is given as follows:

$$\begin{aligned}
\Pr[B^{\mathcal{E}} = 1] &= \sum_{i=1}^l \Pr[v = i] \Pr[A_1^{T^{\mathcal{E}}(v, \cdot)} = 1 | v = i] \\
&= \frac{1}{l} \left(\Pr[A_1^{\hat{F}_k} = 1] + \sum_{v=2}^l \Pr[A_1^{T^{\mathcal{E}}(v, \cdot)} = 1] \right) \quad (\because \text{Eq. (18)}) \\
&= \frac{1}{l} \left(\Pr[A_1^{\hat{F}_k} = 1] + \sum_{v=2}^l \Pr[A_1^{T^{\mathcal{P}}(v-1, \cdot)} = 1] \right) \quad (\because \text{Eq. (19)}) \\
&= \frac{1}{l} \left(\Pr[A_1^{\hat{F}_k} = 1] + \sum_{v=1}^{l-1} \Pr[A_1^{T^{\mathcal{P}}(v, \cdot)} = 1] \right) \quad (21)
\end{aligned}$$

$$\begin{aligned}
\Pr[B^{\mathcal{P}} = 1] &= \sum_{i=1}^l \Pr[v = i] \Pr[A_1^{T^{\mathcal{P}}(v, \cdot)} = 1 | v = i] \\
&= \frac{1}{l} \left(\sum_{v=1}^{l-1} \Pr[A_1^{T^{\mathcal{P}}(v, \cdot)} = 1] + \Pr[A_1^{\eta} = 1] \right) \quad (\because \text{Eq. (20)}) \quad (22)
\end{aligned}$$

Subtracting Eq. (22) from Eq. (21) gives the advantage of B against E .

$$\begin{aligned}
\mathbf{Adv}_E^{4\text{-prp-rka}}(B) &= |\Pr[B^{\mathcal{E}} = 1] - \Pr[B^{\mathcal{P}} = 1]| \\
&= \frac{1}{l} \left| \Pr[A_1^{\hat{F}_k} = 1] - \Pr[A_1^{\eta} = 1] \right| \quad (\because \text{Eq. (21), Eq. (22)}) \\
&= \frac{1}{l} \mathbf{Adv}_{\hat{F}}^{\text{prf}}(A_1) \\
&\geq \frac{1}{\text{idxF}(\ell - \kappa)} \mathbf{Adv}_{\hat{F}}^{\text{prf}}(A_1) \quad (\because \text{len}(m) \leq \ell - \kappa) \\
&\geq \frac{1}{\text{idXH}(\ell)} \mathbf{Adv}_{\hat{F}}^{\text{prf}}(A_1)
\end{aligned}$$

The proof is completed. \square

Proposition 1. *When $T^{\mathcal{V}}(v, m)$ is invoked only once, the following equations hold.*

$$\begin{aligned}
T^{\mathcal{E}}(1, m) &= \hat{F}_k(m) \\
T^{\mathcal{P}}(l, m) &= \eta(m) \\
T^{\mathcal{E}}(v, m) &= T^{\mathcal{P}}(v-1, m) \quad \text{for } 2 \leq v \leq l
\end{aligned}$$

Proof. When $\mathcal{V} = \mathcal{E}$ and $v = 1$, x_1 is computed in Line 107 as follows:

$$\begin{aligned}
x_1 &\leftarrow \text{tr}(E_{k_1}(m_1) \oplus m_1) \parallel \text{tr}(E_{k_2}(m_1) \oplus m_1) \\
&= g(k, m_1),
\end{aligned}$$

where we denoted k_1 by k and $k_2 = k \oplus c_1$. It turns out that

$$T^{\mathcal{E}}(1, m) = h(\dots h(h(g(k, m_1), m_2), m_3) \dots m_l) = \hat{F}_k(m).$$

When $\mathcal{V} = \mathcal{P}$ and $v = l$, x_l is computed in Line 109 as follows:

$$T^{\mathcal{P}}(l, m) = x_l \leftarrow \text{tr}(\pi_{k_1}(m_v) \oplus m_v) \parallel \text{tr}(\pi_{k_3}(m_v) \oplus m_v) \quad (23)$$

Recall that $T^{\mathcal{V}}(v, m)$ is invoked only once. Since each π_{k_i} is the random permutation, the distribution of the first output of π_{k_i} is uniform on $\{0, 1\}^n$. It follows that the distribution of x_l is uniform on $\{0, 1\}^\kappa$. Since the distribution of $\eta(m)$ is also uniform on $\{0, 1\}^\kappa$, Eq. (23) is equivalent to $x_l \leftarrow \eta(m)$.

Lastly, we show $T^{\mathcal{E}}(v, m) = T^{\mathcal{P}}(v - 1, m)$ depending on the last index of message blocks of m , denoted by l .

– $l \leq v - 2$: In both of functions, x_l is computed in Line 104.

– $l = v - 1$:

- $T^{\mathcal{E}}(v, m)$: Since x_l is computed in Line 104, x_l is uniformly distributed on $\{0, 1\}^\kappa$.

- $T^{\mathcal{P}}(v - 1, m)$: x_l is computed in Line 109, (i.e., Eq. (23)). For the reason above, x_l is uniformly distributed on $\{0, 1\}^\kappa$.

– $l = v$:

- $T^{\mathcal{E}}(v, m)$: x_l is computed in Line 109.

$$x_l \leftarrow \text{tr}(E_{k_1}(m_l) \oplus m_l) \parallel \text{tr}(E_{k_3}(m_l) \oplus m_l), \quad (24)$$

where $k_3 = k_1 \oplus c_2$.

- $T^{\mathcal{P}}(v - 1, m)$: x_{l-1} is computed in Line 109.

$$x_{l-1} \leftarrow \text{tr}(\pi_{k_1}(m_{v-1}) \oplus m_{v-1}) \parallel \text{tr}(\pi_{k_3}(m_{v-1}) \oplus m_{v-1})$$

For the reason above, x_{l-1} is uniformly distributed on $\{0, 1\}^\kappa$. After that, x_l is computed in Line 109.

$$x_l \leftarrow \text{tr}(E_{x_{l-1}}(m_l) \oplus m_l) \parallel \text{tr}(E_{x_{l-1} \oplus c_2}(m_l) \oplus m_l) \quad (25)$$

Since the distribution of x_{l-1} is the same as that of k_1 in $T^{\mathcal{E}}(v, m)$, the distribution of x_l in Eq. (25) is the same as that of x_l in Eq. (24).

– $l \geq v + 1$:

- $T^{\mathcal{E}}(v, m)$: x_v is computed in Line 109.

$$x_v \leftarrow \text{tr}(E_{k_1}(m_v) \oplus m_v) \parallel \text{tr}(E_{k_3}(m_v) \oplus m_v) = h(k, m_v) \quad (26)$$

- $T^{\mathcal{P}}(v - 1, m)$: x_{v-1} is computed in Line 109.

$$x_{v-1} \leftarrow \text{tr}(\pi_{k_1}(m_{v-1}) \oplus m_{v-1}) \parallel \text{tr}(\pi_{k_3}(m_{v-1}) \oplus m_{v-1}).$$

For the reason above, x_{v-1} is uniformly distributed on $\{0, 1\}^\kappa$. x_v is computed in Line 112.

$$x_v \leftarrow \text{tr}(E_{x_{v-1}}(m_v) \oplus m_v) \parallel \text{tr}(E_{x_{v-1} \oplus c_2}(m_v) \oplus m_v) = h(x_{v-1}, m_v) \quad (27)$$

Since the distribution of k in Eq. (26) is the same as that of x_{v-1} in Eq. (27), the distribution of x_v in Eq. (26) is the same as that of x_v in Eq. (27). The subsequent steps in the both of functions are identical.

□

Indifferentiability of Double Length Compression Functions

Bart Mennink

Dept. Electrical Engineering, ESAT/COSIC,
KU Leuven, and iMinds, Belgium
`bart.mennink@esat.kuleuven.be`

Abstract. Double block length hashing covers the idea of constructing a compression function on $2n$ bits using an n -bit block cipher. In this work, we present a comprehensive indifferentiability analysis of all relevant double length compression functions. Indifferentiability is a stronger security notion than collision and preimage resistance and ensures that a design has no structural flaws. It is very well suited for composition: using an indifferentiable compression function in a proper mode of operation supplies an indifferentiable hash function. Yet, as we demonstrate compression function indifferentiability is not at all a triviality: almost all double length compression functions, including Tandem-DM and Jetchev et al.'s, appear to be differentiable from a random function in 2 queries. Nevertheless, we also prove that two known functions are indifferentiable: the MDC-4 compression function (up to $2^{n/4}$ queries tight) and Mennink's function (up to $2^{n/2}$ queries tight).

Keywords: double block length, block cipher based, compression function, indifferentiability.



1 Introduction

Double (block) length hashing is a well-established method for constructing a compression function with $2n$ -bit output based only on n -bit block ciphers. The idea dates back to the designs of MDC-2 and MDC-4 in 1988 by Meyer and Schilling [22]. Double length hash functions have an obvious advantage over classical block cipher based functions such as Davies-Meyer and Matyas-Meyer-Oseas, and more generally the PGV class of functions [25, 28]: the same type of underlying primitive allows for a larger compression function. Yet, for double length compression functions it is harder to achieve optimal n -bit collision and $2n$ -bit preimage security.

We focus on the simplest type of double length compression functions, namely those that compress $3n$ to $2n$ bits. Adopting the convention of Mennink [20], we divide the state of the art into two classes: DBL^{2n} , consisting of all functions that internally evaluate a $2n$ -bit keyed block cipher $E : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, and DBL^n , of functions based on an n -bit keyed block cipher $E : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. A classification on the collision and preimage security of the

known functions is given in Table 1 (this classification is accredited to [20]). Regarding these security properties, DBL^{2n} is well-understood. For instance, the most notable Tandem-DM and Abreast-DM [11] and Hirose’s function [7], that all make two underlying block cipher calls, are proven optimally secure with respect to both security notions. Hirose [6] and Özen and Stam [24] presented generalizations of these compression function designs (but for convenience all of these results are handled separately in this work). Stam introduced a single-call compression function [27,28] (reconsidered in [16]) which is proven optimally collision secure. Lucks [17] introduced a compression function that allows for collisions in about $2^{n/2}$ queries—and is therefore not included in the classification—but achieves optimal collision resistance in the iteration. On the other hand, in the DBL^n class the first provably optimally collision secure function was presented only recently by Mennink [20]: his function is proven collision secure up to 2^n queries and preimage secure up to $2^{3n/2}$ queries. Earlier designs in this class are the MDC-2 and MDC-4 compression functions [22] and MJH [13], which are merely constructed to achieve security in the iteration,¹ and Jetchev et al.’s construction (which we will call JOS) [9], a clever design achieving collision security up to $2^{2n/3}$ queries (with preimage security guaranteed up to 2^n queries).

Table 1. Asymptotic ideal cipher model security guarantees of known functions in the classes DBL^{2n} (first) and DBL^n (second). The collision and preimage results are taken from [20]; all indifferentiability results (in **bold**) are derived in this paper.

compression function	E -calls	collision security	preimage security	indifferentiability	underlying cipher
Stam’s	1	2^n [28]	2^n [28]	2 (Sect. 3)	
Tandem-DM	2	2^n [14]	2^{2n} [2, 15]	2 (Sect. 3)	
Abreast-DM	2	2^n [5, 12]	2^{2n} [2, 15]	2 (Sect. 3)	
Hirose’s	2	2^n [7]	2^{2n} [2, 15]	2 (Sect. 3)	
Hirose-class	2	2^n [6]	2^n [6]	2 (Sect. 3)	
Özen-Stam-class	2	2^n [24]	2^n [24]	2 (Sect. 4)	
MDC-2	2	$2^{n/2}$	2^n	2 (Sect. 5)	
MJH	2	$2^{n/2}$	2^n	2 (Sect. 5)	
JOS	2	$2^{2n/3}$ [9]	2^n [9]	2 (Sect. 6)	
Mennink’s	3	2^n [20]	$2^{3n/2}$ [20]	$2^{n/2}$ (Sect. 7)	
MDC-4	4	$2^{5n/8}$ [21]	$2^{5n/4}$ [21]	$2^{n/4}$ (Sect. 8)	

So far, these results only concern the collision and preimage security of the compression functions. If such compression function is used in a proper iteration, these carry over to the hash function design [1]. Beyond these notions,

¹ In the iteration, collision resistance is proven up to $2^{3n/5}$ for MDC-2 [29] and $2^{2n/3}$ for MJH [13].

the *indifferentiability* framework of Maurer et al. [18] has gained recent attention. Indifferentiability is an important security criterion as it guarantees that a construction based on an underlying idealized primitive shows no structural flaws: generic attacks on such a design are impossible up to the proven bound, and weaknesses, if any, come from the underlying primitive. It is well suited for composition: a hash function indifferentiability result (based on an underlying compression function) and a compression function indifferentiability result (based on, say, a block cipher) compose to security of the hash function based on the ideality of the block cipher. Several hash function indifferentiability results exist [3, 4, 8] and compression functions are usually easier to analyze than hash functions, and therefore it is of interest to study the indifferentiability of compression functions.

But, returning to block cipher based compression functions, the state of affairs is entirely topsy-turvy when it comes to indifferentiability. First of all, as for single block length compression functions, the PGV functions are known to be differentiable from random functions [10]. As a first contribution of this work, we show that this problematic situation also applies to double length functions: all functions in the DBL^{2n} class, as well as MDC-2, MJH, and JOS (in the DBL^n class), are trivially differentiable from a random function in 2 queries. The attacks show similarities with the differentiability attacks on the PGV functions. In general, indifferentiability appears to be much harder to achieve than “simply” collision and preimage security.

However, on the positive side, we derive non-trivial indifferentiability results for Mennink’s and the MDC-4 compression function. Starting with Mennink’s compression function class, called $F_{\mathbf{A}}^3$ (see Fig. 2). These functions make three block cipher calls and are indexed by a 4×4 matrix \mathbf{A} that is required to comply with certain simple conditions. We prove that any $F_{\mathbf{A}}^3$ meeting these conditions is indifferentiable from a random function in about $2^{n/2}$ queries (tight). This bound is worse than the collision and preimage bounds, but this is as expected, given the negative indifferentiability results so far. The proof crucially relies on two key characteristics of $F_{\mathbf{A}}^3$: that any two block cipher evaluations of $F_{\mathbf{A}}^3$ define the inputs to the third one, but more importantly, that at least two such calls *are needed* to learn something about an $F_{\mathbf{A}}^3$ evaluation. In general, the proof is made possible by the sequential block cipher evaluation of the design.

Next, for the MDC-4 compression function (see Fig. 5) based on two distinct block ciphers, we prove it indifferentiable from a random function up to $2^{n/4}$ queries (tight).² The proof is very similar to the one of Mennink’s function, and in particular also crucially relies on the sequential block cipher evaluation.

All indifferentiability results are summarized in Table 1, in which we also mention the corresponding section of this paper. The work is concluded in Sect. 9.

² The MDC-4 compression function based on one single block cipher is differentiable in 2 queries.

2 Indifferentiability

The indifferentiability framework, introduced by Maurer et al. [18], is a security notion that formally captures the “distance” between a cryptographic construction and its random equivalent. Informally, it gives a sufficient condition under which an ideal primitive \mathcal{R} can be replaced by some construction $\mathcal{C}^{\mathcal{P}}$ using an ideal subcomponent \mathcal{P} . In this paper, we employ the adaption and simplification by Coron et al. [4]. Recent results by Ristenpart et al. [26] show that indifferentiability does not capture all properties of a random oracle, it applies to single stage games only. Nevertheless, this notion captures pretty many games and remains the best way to prove that a hash or compression function behaves like a random oracle.

Definition 1. *Let \mathcal{C} be a cryptographic primitive with oracle access to an ideal primitive \mathcal{P} . Let \mathcal{R} be an ideal primitive with the same domain and range as \mathcal{C} . Let \mathcal{S} be a simulator with the same domain and range as \mathcal{P} with oracle access to \mathcal{R} and making at most $q_{\mathcal{S}}$ queries, and let \mathcal{D} be a distinguisher making at most $q_{\mathcal{D}}$ queries. The differentiability advantage of \mathcal{D} is defined as*

$$\text{adv}_{\mathcal{C},\mathcal{S}}^{\text{iff}}(\mathcal{D}) = |\Pr(\mathcal{D}^{\mathcal{C},\mathcal{P}} = 1) - \Pr(\mathcal{D}^{\mathcal{R},\mathcal{S}} = 1)|.$$

We refer to $(\mathcal{C}, \mathcal{P})$ as the real world, and to $(\mathcal{R}, \mathcal{S})$ as the simulated world. We denote \mathcal{D} 's left oracle (\mathcal{C} or \mathcal{R}) by L and its right oracle (\mathcal{P} or \mathcal{S}) by R .

For $k, n \geq 1$, we denote by $\text{Bloc}(k, n)$ the set of all block ciphers with a k -bit key and n -bit message space. We simply write $\text{Bloc}(n)$ if $k = n$. Throughout, $\mathcal{C} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ corresponds to a compression function design, and \mathcal{P} represents block ciphers from $\text{Bloc}(k, n)$ (where $k = 2n$ for functions in DBL^{2n} and $k = n$ for functions in DBL^n). We stress that some of the designs analyzed in this work are defined to make use of two distinct block ciphers (e.g., one call to a cipher E_1 and one call to E_2). Except for our indifferentiability result on MDC-4, for all of our results it is not relevant whether the underlying ciphers are distinct or the same. Therefore, we consider all designs simply to be based on one single block cipher, unless stated otherwise.

3 Stam's, Tandem-DM, Abreast-DM, Hirose's, and Hirose-Class

In this section, we consider Tandem-DM and Abreast-DM [11] (cf. Fig. 1), Hirose's compression function [7] (cf. Fig. 1) and its generalized Hirose-class [6],³ as well

³ Hirose's function can be seen as a special case of Hirose-class (using that in the attack it is not relevant whether the underlying block ciphers are distinct or the same), and our attack directly carries over.

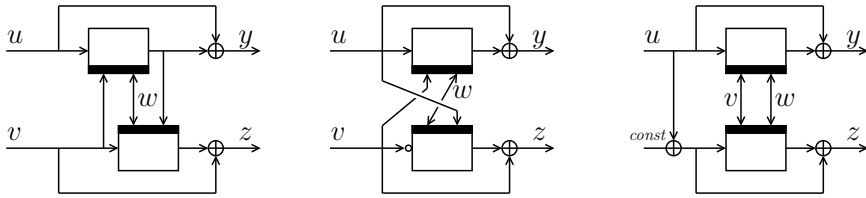


Fig. 1. Tandem-DM (left), Abreast-DM (middle), and Hirose’s compression function (right) [7, 11]. All wires carry n bits. For Abreast-DM, the circle \circ denotes bit complementation. For Hirose’s function, $const$ is any non-zero constant.

as Stam’s supercharged single call Type-I compression function design [27, 28], or more specifically the block cipher based variant considered in [16]:

$$\begin{aligned} \text{Stam}(u, v, w) &= (y, z), \text{ where:} \\ c_1 &\leftarrow E(v\|w, u), \\ y &\leftarrow c_1 + u, \\ z &\leftarrow wy^2 + vy + u. \end{aligned}$$

Here, additions and finite field multiplications are done over the field $GF(2^n)$. The differentiability attacks are identical for all designs, and we only consider Tandem-DM (abbreviated to TDM). The attack is a direct generalization of the fixed-point attack on the Davies-Meyer (DM) compression function.

We note that Özen and Stam presented a generalized double length design [24], and our attack on their class (in Sect. 4) can be seen as a true generalization of the attacks in this section on Abreast-DM and Hirose’s functions (given that in these attacks it is not relevant whether the underlying block ciphers are distinct or the same). Nevertheless, these functions are handled separately for clarity and as an illustration.

Proposition 1. *Let $E \xleftarrow{\$} \text{Bloc}(2n, n)$, and let $\mathcal{R} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ be a random compression function. For any simulator \mathcal{S} that makes at most $q_{\mathcal{S}}$ queries to \mathcal{R} , there exists a distinguisher \mathcal{D} that makes 2 queries to its oracles, such that*

$$\text{adv}_{\text{TDM}, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \geq 1 - \frac{q_{\mathcal{S}} + 1}{2^n}.$$

Proof. Our distinguisher \mathcal{D} aims at finding an evaluation of TDM that satisfies:

$$\text{TDM}(u, v, w) = (u, z), \tag{1}$$

for some values u, v, w, z . \mathcal{D} operates as follows. First, it fixes some values v, w , and queries $u \leftarrow R^{-1}(v\|w, 0)$. Next, it queries its left oracle L on input of (u, v, w) , and outputs 0 if and only if the first half of the response equals u (hence if (1) is satisfied). Clearly, in the real world, (1) holds with certainty, and \mathcal{D} succeeds except if \mathcal{S} or \mathcal{D} obtains a solution to $\mathcal{R}(u, v, w) = (u, z)$. As \mathcal{R} is a random function, any query satisfies this equation with probability $\frac{1}{2^n}$, and \mathcal{R} is consulted at most $q_{\mathcal{S}} + 1$ times. This completes the proof. \square

4 Özen-Stam-Class

Özen and Stam [24] analyzed a wide class of double length compression functions, extending the single-length compression function result of Stam [28].

$$\begin{aligned} \text{OS}(u, v, w) &= (y, z), \text{ where:} \\ (k_1, m_1) &\leftarrow C_1^{\text{pre}}(u, v, w), \\ c_1 &\leftarrow E(k_1, m_1), \\ (k_2, m_2) &\leftarrow C_2^{\text{pre}}(u, v, w), \\ c_2 &\leftarrow E(k_2, m_2), \\ (y, z) &\leftarrow C^{\text{post}}(u, v, w, c_1, c_2). \end{aligned}$$

Here, it is required that C_1^{pre} and C_2^{pre} are bijections, as is $C^{\text{post}}(u, v, w, \cdot, \cdot)$ for fixed (u, v, w) . Additionally, certain requirements are posed on C_1^{aux} and C_2^{aux} (combinations of the three functions), but these are not relevant for our analysis.

We assume the existence of a bijection $M : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$ such that the left half of $M \circ C^{\text{post}}(u, v, w, c_1, c_2)$ is independent of c_2 , and consider the compression function design with M appended. (Note that this does not affect the security result.) For convenience, we simply assume the existence of C_1^{post} and C_2^{post} such that

$$\begin{aligned} y &\leftarrow C_1^{\text{post}}(u, v, w, c_1), \\ z &\leftarrow C_2^{\text{post}}(u, v, w, c_1, c_2). \end{aligned}$$

Proposition 2. *Let $E \stackrel{s}{\leftarrow} \text{Bloc}(2n, n)$, and let $\mathcal{R} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ be a random compression function. For any simulator \mathcal{S} that makes at most $q_{\mathcal{S}}$ queries to \mathcal{R} , there exists a distinguisher \mathcal{D} that makes 2 queries to its oracles, such that*

$$\text{adv}_{\text{OS}, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \geq 1 - \frac{q_{\mathcal{S}} + 1}{2^n}.$$

Proof. The proof is similar to the one of Prop. 1, and we only highlight the differences. Our distinguisher \mathcal{D} aims at finding an evaluation of OS that satisfies:

$$\text{OS}(u, v, w) = (C_1^{\text{post}}(u, v, w, 0), z), \tag{2}$$

for some values u, v, w, z . First, the adversary fixes k_1 , and queries $m_1 \leftarrow R^{-1}(k_1, 0)$. Then, it computes $(u, v, w) \leftarrow C_1^{-\text{pre}}(k_1, m_1)$. Next, it queries its left oracle L on input of (u, v, w) , and outputs 0 if and only if (2) is satisfied. The remainder of the analysis is the same as in the proof of Prop. 1. □

5 MDC-2 and MJH

In this section, we consider the MDC-2 and MJH compression functions. For MDC-2, we leave out the swapping at the end as it is of no influence to the

indifferentiability proof. The functions are defined as follows (for MJH, σ is an involution and θ a constant):

$$\begin{array}{ll}
 \text{MDC-2}(u, v, w) = (y, z), \text{ where:} & \text{MJH}(u, v, w) = (y, z), \text{ where:} \\
 c_1 \leftarrow E(u, w), & c_1 \leftarrow E(v, u + w), \\
 y \leftarrow c_1 + w, & y \leftarrow c_1 + u + w, \\
 c_2 \leftarrow E(v, w), & c_2 \leftarrow E(v, \sigma(u + w)), \\
 z \leftarrow c_2 + w. & z \leftarrow (c_2 + \sigma(u + w)) \cdot \theta + u.
 \end{array}$$

Recall that for our results, it is not relevant whether the underlying ciphers are distinct or the same.

Proposition 3. *Let $E \stackrel{\$}{\leftarrow} \text{Bloc}(n)$, and let $\mathcal{R} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ be a random compression function. For any simulator \mathcal{S} that makes at most $q_{\mathcal{S}}$ queries to \mathcal{R} , there exists a distinguisher \mathcal{D} that makes 2 queries to its oracles, such that*

$$\text{adv}_{\text{MDC-2}, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \geq 1 - \frac{q_{\mathcal{S}} + 1}{2^n}.$$

The same result holds for MJH.

Proof. The proof is similar to the one of Prop. 1. Now, our distinguisher aims at finding an evaluation of MDC-2 that satisfies $\text{MDC-2}(u, v, w) = (w, z)$, and the same for MJH. The remainder of the analysis is almost identical to the proof of Prop. 1, and therefore omitted. \square

6 JOS

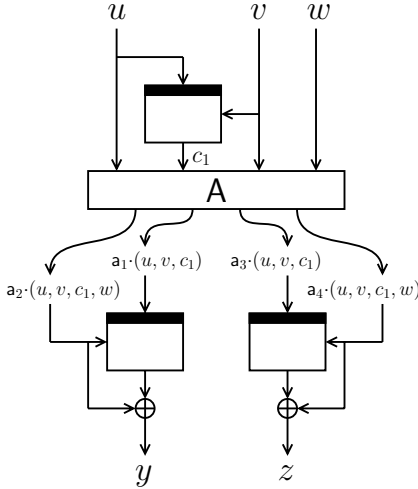
In this section, we consider Jetchev et al.’s compression function (called JOS). The analysis is slightly more complicated but in fact not much different. We consider the block cipher based variant with the underlying matrix A as suggested in [23, Sect. 5.4.2].

$$\begin{array}{l}
 \text{JOS}(u, v, w) = (y, z), \text{ where:} \\
 c_1 \leftarrow E(w, u), \\
 c_2 \leftarrow E(w + uv, v), \\
 y \leftarrow u + v + (u + c_1)(v + c_2), \\
 z \leftarrow u + v + c_1 + c_2.
 \end{array}$$

Here, additions and finite field multiplications are done over the field $GF(2^n)$.

Proposition 4. *Let $E \stackrel{\$}{\leftarrow} \text{Bloc}(n)$, and let $\mathcal{R} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ be a random compression function. For any simulator \mathcal{S} that makes at most $q_{\mathcal{S}}$ queries to \mathcal{R} , there exists a distinguisher \mathcal{D} that makes 2 queries to its oracles, such that*

$$\text{adv}_{\text{JOS}, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \geq 1 - \frac{q_{\mathcal{S}} + 1}{2^n}.$$



$$\begin{aligned}
 F_A^3(u, v, w) &= (y, z), \text{ where:} \\
 c_1 &\leftarrow E(u, v), \\
 k_2 &\leftarrow \mathbf{a}_1 \cdot (u, v, c_1), \\
 m_2 &\leftarrow \mathbf{a}_2 \cdot (u, v, c_1, w), \\
 y &\leftarrow E(k_2, m_2) + m_2, \\
 k_3 &\leftarrow \mathbf{a}_3 \cdot (u, v, c_1), \\
 m_3 &\leftarrow \mathbf{a}_4 \cdot (u, v, c_1, w), \\
 z &\leftarrow E(k_3, m_3) + m_3.
 \end{aligned}$$

Fig. 2. Mennink’s compression function class F_A^3 where A is a 4×4 matrix as in (3)

Proof. The proof is similar to the one of Prop. 1. Now, our distinguisher aims at finding an evaluation of $JOS(u, v, w) = (y, z)$ that satisfies $y + uz = u^2 + u + v$. The remainder of the analysis is almost identical to the proof of Prop. 1, and therefore omitted. \square

7 Mennink’s

Mennink’s double length compression function design, dubbed $F_A^3 : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ (depicted in Fig. 2), makes three calls to a single block cipher, and is indexed by a 4×4 matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{a}_4 \end{pmatrix} = \begin{pmatrix} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} & 0 \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \mathbf{a}_{23} & \mathbf{a}_{24} \\ \mathbf{a}_{31} & \mathbf{a}_{32} & \mathbf{a}_{33} & 0 \\ \mathbf{a}_{41} & \mathbf{a}_{42} & \mathbf{a}_{43} & \mathbf{a}_{44} \end{pmatrix} \tag{3}$$

over the field $GF(2^n)$.⁴

The security of F_A^3 is based on the key principle that any *two* block cipher evaluations define the input to the third one. Indeed, invertibility of A guarantees that evaluations of the second and third block cipher define the values (u, v, c_1) . Also, if $\mathbf{a}_{24} \neq 0$ (resp. $\mathbf{a}_{44} \neq 0$), the first and second (resp. third) block cipher define the inputs to the third (resp. second) one. However, in order to achieve

⁴ Bit strings from $\{0, 1\}^n$ and finite field elements in $GF(2^n)$ are identified to define addition and scalar multiplication over $\{0, 1\}^n$. For two tuples $x = (x_1, \dots, x_l)$ and $y = (y_1, \dots, y_l)$ of elements from $\{0, 1\}^n$, $x \cdot y$ denotes inner product $\sum_{i=1}^l x_i y_i \in \{0, 1\}^n$.

collision and preimage security, Mennink posed a slightly larger set of conditions on \mathbf{A} , which he called **colreq** (for collision security) and **prereq** (for preimage security). For the indistinguishability results, it suffices to pose a much weaker condition on \mathbf{A} . In detail, we require the following from \mathbf{A} (called **indreq**): \mathbf{A} is invertible and $\mathbf{a}_{12}, \mathbf{a}_{13}, \mathbf{a}_{24}, \mathbf{a}_{32}, \mathbf{a}_{33}, \mathbf{a}_{44} \neq 0$. As **prereq** \Rightarrow **colreq** \Rightarrow **indreq**, our results particularly apply to *all* schemes proven secure in [20].

Suiting the analysis, we define a function $\text{get}w$ that, on input of $j \in \{2, 4\}$, $m \in \{0, 1\}^n$, and $(k_1, m_1, c_1) \in \{0, 1\}^{3n}$, outputs w such that $\mathbf{a}_j \cdot (k_1, m_1, c_1, w) = m$. Note that $\mathbf{a}_{24}, \mathbf{a}_{44} \neq 0$ implies uniqueness of w . Differentiability is discussed in Sect. 7.1, and indistinguishability in Sect. 7.2.

7.1 Differentiability

In Prop. 5 we show that $F_{\mathbf{A}}^3$ is differentiable from a random oracle in at most about $2^{n/2}$ queries.

Proposition 5. *Let $E \stackrel{\$}{\leftarrow} \text{Bloc}(n)$, and let $\mathcal{R} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ be a random compression function. For any simulator \mathcal{S} that makes at most $q_{\mathcal{S}}$ queries to \mathcal{R} , there exists a distinguisher \mathcal{D} that makes $2^{n/2} + 2$ queries to its oracles, such that*

$$\text{adv}_{F_{\mathbf{A}}^3, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \geq \frac{1}{2} - \frac{1}{2^{n/2+1}} - \frac{q_{\mathcal{S}} + 1}{2^n - q_{\mathcal{S}}}.$$

Proof. Our distinguisher \mathcal{D} aims at finding two different evaluations of $F_{\mathbf{A}}^3$ with the same key inputs to the second (or third) block cipher call. In more detail, the distinguisher aims at finding two distinct block cipher calls (k_1, m_1, c_1) and (k'_1, m'_1, c'_1) such that for $j \in \{1, 3\}$:

$$\mathbf{a}_j \cdot (k_1, m_1, c_1) = \mathbf{a}_j \cdot (k'_1, m'_1, c'_1). \quad (4)$$

Note that in the real world, for $F_{\mathbf{A}}^3$, such collisions are expected to be found in about $2^{n/2}$ queries to E (here we use that $\mathbf{a}_{12}, \mathbf{a}_{13}, \mathbf{a}_{32}, \mathbf{a}_{33} \neq 0$). If the distinguisher eventually finds a collision as in (4), then for any $m \in \{0, 1\}^n$, the following condition naturally holds in the real world:

$$y = y' \text{ if } j = 1 \text{ and } z = z' \text{ if } j = 3, \quad (5)$$

where

$$\begin{aligned} (y, z) &= F_{\mathbf{A}}^3(k_1, m_1, \text{get}w(j+1, m, k_1, m_1, c_1)), \\ (y', z') &= F_{\mathbf{A}}^3(k'_1, m'_1, \text{get}w(j+1, m, k'_1, m'_1, c'_1)). \end{aligned}$$

In the random world, with $F_{\mathbf{A}}^3$ replaced by \mathcal{R} , this equation only holds with small probability. Note that the simulator never learns the value m , yet, it may simply try to avoid collisions as in (4). However, in this case, the responses from \mathcal{S} are too biased, which allows the distinguisher to succeed.

Formally, the distinguisher \mathcal{D} proceeds as follows.

- (i) \mathcal{D} makes $2^{n/2}$ queries to its right oracle R for different key and different message values, obtaining $2^{n/2}$ distinct tuples (k_1, m_1, c_1) ;
- (ii) If there is no solution to (4), \mathcal{D} returns 1;
- (iii) Let $j \in \{1, 3\}$ and (k_1, m_1, c_1) and (k'_1, m'_1, c'_1) be such that (4) is satisfied;
- (iv) Take $m \xleftarrow{\$} \{0, 1\}^n$. If (5) holds, \mathcal{D} returns 0, and otherwise it returns 1.

Distinguisher \mathcal{D} succeeds except in the following two cases: “ C_1 ” it is conversing with the real world and (4) does not have a solution (which means that his guess in step (ii) is wrong), or “ C_2 ” it is conversing with the simulated world and (5) holds (which means that his guess in step (iv) is wrong). Therefore, $\text{adv}_{F_A^3, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \geq 1 - \Pr(C_1) - \Pr(C_2)$. Regarding C_1 : note that all queries are made with different key inputs, and E is a random cipher. Therefore, all responses are randomly drawn from a set of size 2^n , and a collision (4) occurs with probability at least $\binom{2^{n/2}}{2} \frac{1}{2^n}$ (as $a_{12}, a_{13}, a_{32}, a_{33} \neq 0$). Thus,

$$\Pr(C_1) \leq 1 - \binom{2^{n/2}}{2} \frac{1}{2^n} = \frac{1}{2} + \frac{1}{2^{n/2+1}}.$$

Regarding C_2 , denote by E the event that \mathcal{S} ever queries $\mathcal{R}(k_1, m_1, \text{getw}(j + 1, m, k_1, m_1, c_1))$. Then,

$$\Pr(C_2) \leq \Pr(C_2 \mid \neg E) + \Pr(E) \leq \frac{1}{2^n - q_S} + \frac{q_S}{2^n - (q_S - 1)} = \frac{q_S + 1}{2^n - q_S},$$

where we use that $a_{24}, a_{44} \neq 0$. This completes the proof. □

7.2 Indifferentiability

We prove that F_A^3 is indifferentiable from a random function.

Theorem 1. *Let $E \xleftarrow{\$} \text{Bloc}(n)$, and let $\mathcal{R} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ be a random function. There exists a simulator \mathcal{S} such that for any distinguisher \mathcal{D} that makes at most q_L left queries and q_R right queries,*

$$\text{adv}_{F_A^3, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \leq \frac{7(3q_L + q_R)^2}{2^n},$$

where \mathcal{S} makes $q_S \leq q_R$ queries to \mathcal{R} .

The simulator \mathcal{S} used in the proof mimics the behavior of random cipher E such that queries to \mathcal{S} and queries to \mathcal{R} are consistent, which means that relations among the query outputs in the real world hold in the simulated world as well. In the remainder of the section, we first introduce our simulator and accommodate it with an intuition, and next present the formal proof.

Simulator Intuition

For $k \in \{0, 1\}^n$, the simulator maintains an initially empty list $\mathcal{L}E[k]$. In this list, it stores tuples (m, c) such that $\mathcal{S}(k, m) = c$. We write $\mathcal{L}E^+[k]$ for all input values m and $\mathcal{L}E^-[k]$ for all output values c . Sometimes, we abuse notation and write $(k, m, c) \in \mathcal{L}E$ to denote that $(m, c) \in \mathcal{L}E[k]$.

Mennink's F_A^3 class of functions is based on the key principle that any two block ciphers define the inputs to the third one. The simulator we use for the proof of Thm. 1 enormously benefits from some of these characteristics. In more detail, the simulator is given in Fig. 3.

Apart from the **if**-clause of lines 02-06, the simulator identically mimics an ideal cipher. In this particular clause, the simulator checks whether a query (k, m) may appear in an F_A^3 evaluation (see Fig. 2) as a bottom query (left or right) for some other query appearing in the top. In more detail, this happens if $(k, m) = (\mathbf{a}_j \cdot (k_1, m_1, c_1), \mathbf{a}_{j+1} \cdot (k_1, m_1, c_1, w))$ for some $j \in \{1, 3\}$ and some earlier query $(k_1, m_1, c_1) \in \mathcal{L}E$. In this case, the simulator should consult \mathcal{R} to derive the query response. At a higher level, the simulator is based on the idea that, with high probability, a distinguisher can only compare (F_A^3, E) and $(\mathcal{R}, \mathcal{S})$ if it makes the queries to E/\mathcal{S} “in correct order”: for any evaluation of F_A^3 that can be derived from $\mathcal{L}E$, the top query is made prior to the two bottom queries.

Forward Query $\mathcal{S}(k, m)$	Inverse Query $\mathcal{S}^{-1}(k, c)$
<pre> 00 if $\mathcal{L}E^+[k](m) \neq \perp$ return $c = \mathcal{L}E^+[k](m)$ 01 $c \stackrel{\\$}{\leftarrow} \{0, 1\}^n \setminus \mathcal{L}E^+[k]$ 02 if $\exists j \in \{1, 3\}, (k_1, m_1, c_1) \in \mathcal{L}E : k = \mathbf{a}_j \cdot (k_1, m_1, c_1)$ 03 $w \leftarrow \text{getw}(j + 1, m, k_1, m_1, c_1)$ 04 $(y, z) \leftarrow \mathcal{R}(k_1, m_1, w)$ 05 $c \leftarrow m + (y[j] = 1) + z[j] = 3)$ 06 end if 07 return $\mathcal{L}E^+[k](m) \leftarrow c$ </pre>	<pre> 10 if $\mathcal{L}E^-[k](c) \neq \perp$ return $m = \mathcal{L}E^-[k](c)$ 11 $m \stackrel{\\$}{\leftarrow} \{0, 1\}^n \setminus \mathcal{L}E^-[k]$ 12 return $\mathcal{L}E^-[k](c) \leftarrow m$ </pre>

Fig. 3. The simulator \mathcal{S} for E used in the proof of Thm. 1

Proof of Theorem 1

We formally proof Thm. 1. Let \mathcal{S} be the simulator of Fig. 3, and let \mathcal{D} be any distinguisher that makes at most q_L left queries and q_R right queries. Note that \mathcal{S} makes $q_S \leq q_R$ queries. By Def. 1, the goal is to bound:

$$\text{adv}_{F_A^3, \mathcal{S}}^{\text{iff}}(\mathcal{D}) = \left| \Pr\left(\mathcal{D}^{F_A^3, E} = 1\right) - \Pr\left(\mathcal{D}^{\mathcal{R}, \mathcal{S}} = 1\right) \right|. \quad (6)$$

As a first step, we apply a PRP-PRF switch to both worlds. More formally, we define \tilde{E} as E with the difference that all responses are randomly drawn from $\{0, 1\}^n$. Similarly, $\tilde{\mathcal{S}}$ is defined as \mathcal{S} of Fig. 3 with the difference that random sampling from $\{0, 1\}^n$ is done in lines 01 and 11. Now,

$$\left| \Pr\left(\mathcal{D}^{F_A^3, E} = 1\right) - \Pr\left(\mathcal{D}^{F_A^3, \tilde{E}} = 1\right) \right| \leq \frac{(3q_L + q_R)^2}{2^{n+1}},$$

and

$$\left| \Pr \left(\mathcal{D}^{\mathcal{R}, \tilde{\mathcal{S}}} = 1 \right) - \Pr \left(\mathcal{D}^{\mathcal{R}, \mathcal{S}} = 1 \right) \right| \leq \frac{q_R^2}{2^{n+1}},$$

and we obtain for (6):⁵

$$\text{adv}_{F_A^3, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \leq \left| \Pr \left(\mathcal{D}^{F_A^3, \tilde{E}} = 1 \right) - \Pr \left(\mathcal{D}^{\mathcal{R}, \tilde{\mathcal{S}}} = 1 \right) \right| + \frac{(3q_L + q_R)^2}{2^n}. \quad (7)$$

It remains to analyze the probability of \mathcal{D} to distinguish (F_A^3, \tilde{E}) from $(\mathcal{R}, \tilde{\mathcal{S}})$. Abusing notation, we remain calling these worlds the real and simulated world. These worlds are described in Fig. 4. Here, in both worlds, $\mathcal{L}E$ represents an initially empty list of all right oracle queries, and in the simulated world only we furthermore use $\mathcal{L}\mathcal{R}$ as an initially empty list of all left oracle queries.

Let event $\text{cond}(\mathcal{L}E)$ be defined as follows:

$$\text{cond}(\mathcal{L}E) = \left(\begin{array}{l} \exists j, j' \in \{1, 3\}, (k, m, c), (k', m', c') \in \mathcal{L}E : \\ (k, m, c) \text{ newer than } (k', m', c') \text{ and} \\ \mathbf{a}_j \cdot (k, m, c) \in \{k, k', \mathbf{a}_{j'} \cdot (k', m', c')\} \end{array} \right). \quad (8)$$

Event $\text{cond}(\mathcal{L}E)$ covers the case of two distinct top queries that result to the same key input to two bottom queries, as well as the case of a top query accidentally hitting the key k' of a bottom query (which may be the equal to the top query). Particularly, as long as $\neg \text{cond}(\mathcal{L}E)$, the condition in line 42 of Fig. 4 is always satisfied by at most one $(j, (k_1, m_1, c_1))$. In the remainder, we prove in Lem. 1 that (F_A^3, \tilde{E}) and $(\mathcal{R}, \tilde{\mathcal{S}})$ are perfectly indistinguishable as long as $\text{cond}(\mathcal{L}E)$ does not occur in both worlds. Then, in Lem. 2 we prove that $\text{cond}(\mathcal{L}E)$ occurs in the real world with probability at most $\frac{3(3q_L + q_R)^2}{2^n}$ and in the simulated world with probability at most $\frac{3q_R^2}{2^n}$. Together with (7), this completes the proof.

Lemma 1. *As long as $\neg \text{cond}(\mathcal{L}E)$, (F_A^3, \tilde{E}) from $(\mathcal{R}, \tilde{\mathcal{S}})$ are perfectly indistinguishable.*

Proof. We consider any query made by the distinguisher, either to the left oracle L (either F_A^3 or \mathcal{R}) and the right oracle R/R^{-1} (either \tilde{E}/\tilde{E}^{-1} or $\tilde{\mathcal{S}}/\tilde{\mathcal{S}}^{-1}$), and show that the query responses are equally distributed in both worlds (irrespectively of the query history). Without loss of generality, we consider new queries only: if the distinguisher makes a repetitive query, the answer is known and identically distributed in both worlds.

⁵ Technically, we could have taken $\tilde{\mathcal{S}}$ as our simulator, therewith obtaining an improved indifferentiability bound for Thm. 1. However, for clarity and ease of presentation, we opted for simulator \mathcal{S} .

Query $F_A^3(u, v, w)$ 00 $c_1 \leftarrow \tilde{E}(u, v)$ 01 $k_2 \leftarrow \mathbf{a}_1 \cdot (u, v, c_1)$ 02 $m_2 \leftarrow \mathbf{a}_2 \cdot (u, v, c_1, w)$ 03 $y \leftarrow \tilde{E}(k_2, m_2) + m_2$ 04 $k_3 \leftarrow \mathbf{a}_3 \cdot (u, v, c_1)$ 05 $m_3 \leftarrow \mathbf{a}_4 \cdot (u, v, c_1, w)$ 06 $z \leftarrow \tilde{E}(k_3, m_3) + m_3$ 07 return (y, z)	Query $\mathcal{R}(u, v, w)$ 30 if $\mathcal{LR}(u, v, w) \neq \perp$ return $(y, z) = \mathcal{LR}(u, v, w)$ 31 $(y, z) \stackrel{\$}{\leftarrow} \{0, 1\}^{2n}$ 32 return $\mathcal{LR}(u, v, w) \leftarrow (y, z)$
Query $\tilde{E}(k, m)$ 10 if $\mathcal{LE}^+[k](m) \neq \perp$ return $c = \mathcal{LE}^+[k](m)$ 11 $c \stackrel{\$}{\leftarrow} \{0, 1\}^n$ 12 return $\mathcal{LE}^+[k](m) \leftarrow c$	Query $\tilde{\mathcal{S}}(k, m)$ 40 if $\mathcal{LE}^+[k](m) \neq \perp$ return $c = \mathcal{LE}^+[k](m)$ 41 $c \stackrel{\$}{\leftarrow} \{0, 1\}^n$ 42 if $\exists j \in \{1, 3\}, (k_1, m_1, c_1) \in \mathcal{LE} : k = \mathbf{a}_j \cdot (k_1, m_1, c_1)$ 43 $w \leftarrow \text{getw}(j+1, m, k_1, m_1, c_1)$ 44 $(y, z) \leftarrow \mathcal{R}(k_1, m_1, w)$ 45 $c \leftarrow m + (y[j=1] + z[j=3])$ 46 end if 47 return $\mathcal{LE}^+[k](m) \leftarrow c$
Query $\tilde{E}^{-1}(k, c)$ 20 if $\mathcal{LE}^-[k](c) \neq \perp$ return $m = \mathcal{LE}^-[k](c)$ 21 $m \stackrel{\$}{\leftarrow} \{0, 1\}^n$ 22 return $\mathcal{LE}^-[k](c) \leftarrow m$	Query $\tilde{\mathcal{S}}^{-1}(k, c)$ 50 if $\mathcal{LE}^-[k](c) \neq \perp$ return $m = \mathcal{LE}^-[k](c)$ 51 $m \stackrel{\$}{\leftarrow} \{0, 1\}^n$ 52 return $\mathcal{LE}^-[k](c) \leftarrow m$

Fig. 4. The worlds (F_A^3, \tilde{E}) (left) and $(\mathcal{R}, \tilde{\mathcal{S}})$ (right)

L -query (u, v, w) . We make the following distinction:

1. $\mathcal{LE}^+[u](v) = \perp$. In the real world, this means that the first cipher call $\tilde{E}(u, v)$ is new, and answered with a fresh value. As $\text{cond}(\mathcal{LE})$ does not occur, also the second and third call, $\tilde{E}(k_2, m_2)$ and $\tilde{E}(k_3, m_3)$, are fresh, and both their responses are drawn from $\{0, 1\}^n$. Regarding the simulated world, by the condition “ $\mathcal{LE}^+[u](v) = \perp$,” $\tilde{\mathcal{S}}$ has never queried \mathcal{R} on input of (u, v, w) . Indeed, it had only queried \mathcal{R} if the condition of line 42 was satisfied for some $j \in \{1, 3\}$ and existing $(u, v, c_1) \in \mathcal{LE}$. Thus, also in this world the response is randomly generated from $\{0, 1\}^{2n}$;
2. $\mathcal{LE}^+[u](v) \neq \perp$. Note that in the real world, this element could have been added to \mathcal{LE} via \mathcal{D} or via F_A^3 . Let $c_1 = \mathcal{LE}^+[u](v)$, and write $(k_2, m_2) = (\mathbf{a}_1 \cdot (u, v, c_1), \mathbf{a}_2 \cdot (u, v, c_1, w))$ and $(k_3, m_3) = (\mathbf{a}_3 \cdot (u, v, c_1), \mathbf{a}_4 \cdot (u, v, c_1, w))$. We make the following distinction:
 - $\mathcal{LE}^+[k_2](m_2) = \perp$ and $\mathcal{LE}^+[k_3](m_3) = \perp$. In the real world, the answers to the queries $\tilde{E}(k_2, m_2)$ and $\tilde{E}(k_3, m_3)$ are both fresh and randomly drawn from $\{0, 1\}^n$. Regarding the simulated world, by contradiction we prove that $\mathcal{R}(u, v, w)$ has never been queried before by $\tilde{\mathcal{S}}$. Indeed, suppose it has been queried before. This necessarily means that there exist $j \in \{1, 3\}$ and $(u, v, c_1) \in \mathcal{LE}$ such that $\mathbf{a}_j \cdot (u, v, c_1) = k'$ and $w = \text{getw}(j+1, m', u, v, c_1)$ for some $(k', m', c') \in \mathcal{LE}$. The former implies $k' = k_2[j=1] + k_3[j=3]$, and the latter implies $m' = \mathbf{a}_{j+1} \cdot (u, v, c_1, w)$ and thus $m' = m_2[j=1] + m_3[j=3]$. This contradicts the condition that (k_2, m_2) and (k_3, m_3) are not in \mathcal{LE} . Therefore, the query (u, v, w) to \mathcal{R} is new, and the response is randomly drawn from $\{0, 1\}^{2n}$;

- $\mathcal{L}E^+[k_2](m_2) \neq \perp$ and/or $\mathcal{L}E^+[k_3](m_3) \neq \perp$. Without loss of generality, assume the former and write $c_2 = \mathcal{L}E^+[k_2](m_2)$. In the real world, this query could not have been made in an earlier evaluation of F_A^3 (by virtue of $\text{cond}(\mathcal{L}E)$). Therefore, the distinguisher must have made this query, and particular knows $y = c_2 + m_2$, which is the left half of the query response. In the simulated world, a similar story applies: by $\neg\text{cond}(\mathcal{L}E)$, this query to $\tilde{\mathcal{S}}$ must have been made after (u, v, c_1) , and thus, the response value c_2 equals $m + y$ by line 45, where y equals the left half of $\mathcal{R}(u, v, w)$. Thus also in this case, the distinguisher knows the left half of the query response.
 If also $\mathcal{L}E^+[k_3](m_3) \neq \perp$, the same reasoning applies to z , the second half of the query response. On the other hand, in case $\mathcal{L}E^+[k_3](m_3) = \perp$, the previous bullet carries over to the z -part.

R-query (k, m) . We make the following distinction:

1. $\neg \exists j \in \{1, 3\}, (k_1, m_1, c_1) \in \mathcal{L}E : k = a_j \cdot (k_1, m_1, c_1)$. In the simulated world, the response is randomly drawn from $\{0, 1\}^n$ by construction. Regarding the real world, first assume (k, m) has never been queried to \tilde{E} via a query to F_A^3 . Then, the response is clearly fresh and randomly drawn from $\{0, 1\}^n$. However, it may be the case that the \tilde{E} -query could have been triggered by an earlier F_A^3 -query. However, by the condition, it could have possibly appeared in such evaluation as a bottom left/right query. It may have appeared as a top query in an F_A^3 evaluation, which means that (k, m, w) has been queried to F_A^3 for some w . However, in this setting, the adversary never learnt c_1 , and thus the response to the R -query appears completely randomly drawn from $\{0, 1\}^n$;
2. $\exists j \in \{1, 3\}, (k_1, m_1, c_1) \in \mathcal{L}E : k = a_j \cdot (k_1, m_1, c_1)$. By $\neg\text{cond}(\mathcal{L}E)$, these values are unique. Let $w = \text{getw}(j+1, m, k_1, m_1, c_1)$. In the simulated world, the response c is defined as $m + y$ (if $j = 1$) or $m + z$ (if $j = 3$), where $(y, z) = \mathcal{R}(k_1, m_1, w)$. Clearly, if the distinguisher has queried $\mathcal{R}(k_1, m_1, w)$ before, it knows the response in advance. Otherwise, it is randomly drawn from $\{0, 1\}^n$ by construction. Regarding the real world, the same reasoning applies: either the query is new, or it must have appeared as a bottom query (left if $j = 1$, right if $j = 3$) of an earlier F_A^3 evaluation (by $\neg\text{cond}(\mathcal{L}E)$), in which case the distinguisher knows the response.

R^{-1} -query (k, c) . In the simulated world, queries are always answered with a random answer from $\{0, 1\}^n$. In the real world, this is also the case, except if a certain query (k, m) with $\mathcal{L}E^+[k](m) = c$ has ever been triggered via a call to F_A^3 . However, in this case, the response will still appear completely random to the distinguisher, similar to the first item of forward queries to R . □

Lemma 2. $\Pr\left(\text{cond}(\mathcal{L}E) \text{ for } (F_A^3, \tilde{E})\right) \leq \frac{3(3q_L+q_R)^2}{2^n}$ and

$\Pr\left(\text{cond}(\mathcal{L}E) \text{ for } (\mathcal{R}, \tilde{\mathcal{S}})\right) \leq \frac{3q_R^2}{2^n}$.

Proof. We start with the real world (F_A^3, \tilde{E}) . At the end of the proof, we highlight the differences that give rise to the bound for the simulated world $(\mathcal{R}, \tilde{\mathcal{S}})$.

Let $1 \leq i \leq 3q_L + q_R$, and denote by $\mathcal{L}E_i$ the set $\mathcal{L}E$ after the i th query. We assume $\neg\text{cond}(\mathcal{L}E_{i-1})$ and consider the probability $\text{cond}(\mathcal{L}E_i)$ gets satisfied. More detailed, we consider the probability that the i th query makes the condition satisfied for some $j, j' \in \{1, 3\}$ and some earlier query $(k', m', c') \in \mathcal{L}E$. Note that $\text{cond}(\mathcal{L}E_i)$ can only be triggered by the values derived in lines 11 and 21. In fact, these values are always randomly generated from $\{0, 1\}^n$.

Decomposing $\text{cond}(\mathcal{L}E_i)$, the i th query satisfies the condition if it satisfies any of the following three:

$$\begin{aligned} a_{j \cdot}(k, m, c) &= k && \text{for } j \in \{1, 3\}, \\ a_{j \cdot}(k, m, c) &= k' && \text{for } j \in \{1, 3\} \text{ and } (k', m', c') \in \mathcal{L}E_{i-1}, \\ a_{j \cdot}(k, m, c) &= a_{j' \cdot}(k', m', c') && \text{for } j, j' \in \{1, 3\} \text{ and } (k', m', c') \in \mathcal{L}E_{i-1}. \end{aligned}$$

Therefore, $\text{cond}(\mathcal{L}E_i)$ gets satisfied with probability at most $\frac{6(i-1)+2}{2^n}$ (as $a_{12}, a_{13}, a_{32}, a_{33} \neq 0$). We thus find:

$$\begin{aligned} \Pr(\text{cond}(\mathcal{L}E)) &\leq \sum_{i=1}^{3q_L+q_R} \Pr(\text{cond}(\mathcal{L}E_i) \mid \neg\text{cond}(\mathcal{L}E_{i-1})) \\ &\leq \sum_{i=1}^{3q_L+q_R} \frac{6(i-1)+2}{2^n} \leq \frac{3(3q_L+q_R)^2}{2^n}. \end{aligned}$$

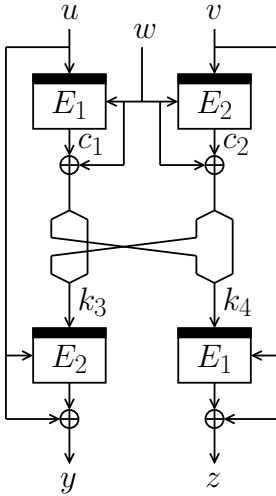
Now, for the simulated world, first note that $1 \leq i \leq q_R$. In this setting, $\text{cond}(\mathcal{L}E_i)$ can only be triggered by the values derived in lines 41, 45, and 51. We remark that in line 45, the value c is indeed always a random n -bit value by $\neg\text{cond}(\mathcal{L}E_{i-1})$. □

8 MDC-4

For MDC-4, we leave out the swapping at the end as it is of no influence to the indistinguishability proof. The function is given in Fig. 5. Here, for a bit string x , we write x^l and x^r to denote its left and right halves where $|x^l| = |x^r|$. MDC-4 achieves a higher level of indistinguishability security than MDC-2, mainly due to the two sequential rounds. Differentiability is discussed in Sect. 8.1, and indistinguishability in Sect. 8.2.

8.1 Differentiability

In Prop. 6 we show that MDC-4 is differentiable from a random oracle in at most about $2^{n/4}$ queries. The attack is very similar to the attack of Prop. 5, but is included for convenience. We briefly note that if $E_1 = E_2$, MDC-4 is clearly differentiable in 2 queries, exploiting that $\text{MDC-4}(u, u, w)$ has the same left and right half for any $u, w \in \{0, 1\}^n$.



MDC-4(u, v, w) = (y, z), where:

$$\begin{aligned}
 c_1 &\leftarrow E_1(u, w), \\
 c_2 &\leftarrow E_2(v, w), \\
 k_3 &\leftarrow c_2^l \parallel c_1^r + w, \\
 y &\leftarrow E_2(k_3, u) + u, \\
 k_4 &\leftarrow c_1^l \parallel c_2^r + w, \\
 z &\leftarrow E_1(k_4, v) + v.
 \end{aligned}$$

Fig. 5. The MDC-4 compression function. For convenience, the swapping at the end is omitted.

Proposition 6. Let $E_1, E_2 \stackrel{\$}{\leftarrow} \text{Bloc}(n)$, and let $\mathcal{R} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ be a random compression function. For any simulator \mathcal{S} that makes at most q_S queries to \mathcal{R} , there exists a distinguisher \mathcal{D} that makes $2^{n/4} + 2$ queries to its oracles, such that

$$\text{adv}_{\text{MDC-4}, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \geq \frac{1}{2} - \frac{1}{2^{n/4+1}} - \frac{q_S + 1}{2^n - q_S}.$$

Proof. Our distinguisher \mathcal{D} aims at finding two different evaluations of MDC-4 with the same key inputs to the bottom left block cipher call. In more detail, the distinguisher fixes u and w and aims at finding two distinct block cipher calls (v, w, c_2) and (v', w, c'_2) such that:

$$c_2^l = c'_2{}^l. \tag{9}$$

Note that in the real world, for MDC-4, such collisions are expected to be found in about $2^{n/4}$ queries to E . If the distinguisher eventually finds a collision as in (9), then the following condition naturally holds in the real world:

$$y := \text{MDC-4}(u, v, w)^l = \text{MDC-4}(u, v', w)^l =: y'. \tag{10}$$

In the random world, with MDC-4 replaced by \mathcal{R} , this equation only holds with small probability. Note that the simulator never learns the value u , yet, it may simply try to avoid collisions as in (9). However, in this case, the responses from \mathcal{S} are too biased, which allows the distinguisher to succeed.

Formally, the distinguisher \mathcal{D} proceeds as follows.

- (i) \mathcal{D} makes $2^{n/4}$ queries to its right oracle R for different key values and for a fixed message value w , obtaining $2^{n/4}$ distinct tuples (v, w, c_2) ;

- (ii) If there is no solution to (9), \mathcal{D} returns 1;
- (iii) Let (v, w, c_2) and (v', w, c'_2) be such that (9) is satisfied;
- (iv) Take $u \xleftarrow{\$} \{0, 1\}^n$. If (10) holds, \mathcal{D} returns 0, and otherwise it returns 1.

Distinguisher \mathcal{D} succeeds except in the following two cases: “ C_1 ” it is conversing with the real world and (9) does not have a solution (which means that his guess in step (ii) is wrong), or “ C_2 ” it is conversing with the simulated world and (10) holds (which means that his guess in step (iv) is wrong). Therefore, $\text{adv}_{\text{MDC-4}, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \geq 1 - \Pr(C_1) - \Pr(C_2)$. Regarding C_1 : note that all queries are made with different key inputs, and E_2 is a random cipher. Therefore, all responses are randomly drawn from a set of size 2^n , and a collision (4) occurs with probability at least $\binom{2^{n/4}}{2} \frac{2^{n/2}}{2^n}$. Thus,

$$\Pr(C_1) \leq 1 - \binom{2^{n/4}}{2} \frac{2^{n/2}}{2^n} = \frac{1}{2} + \frac{1}{2^{n/4+1}}.$$

Regarding C_2 , the proof of Prop. 5 carries over and we find $\Pr(C_2) \leq \frac{q_S+1}{2^n-q_S}$. This completes the proof. \square

8.2 Indifferentiability

We prove that MDC-4 is indifferentiable from a random function.

Theorem 2. *Let $E_1, E_2 \xleftarrow{\$} \text{Bloc}(n)$, and let $\mathcal{R} : \{0, 1\}^{3n} \rightarrow \{0, 1\}^{2n}$ be a random function. There exists a simulator \mathcal{S} such that for any distinguisher \mathcal{D} that makes at most q_L left queries and q_R right queries,*

$$\text{adv}_{\text{MDC-4}, \mathcal{S}}^{\text{iff}}(\mathcal{D}) \leq \frac{6(4q_L + q_R)^2}{2^{n/2}},$$

where \mathcal{S} makes $q_S \leq q_R$ queries to \mathcal{R} .

The proof is truly similar to the proof of Thm. 1. We now only present the simulator; the formal proof is presented in the full version of this paper.

Simulator Intuition

Similar to Sect. 7.2, the simulator maintains an initially empty lists $\mathcal{L}E_1[k]$ (corresponding to E_1) and $\mathcal{L}E_2[k]$ (corresponding to E_2) for $k \in \{0, 1\}^n$. Abusing notation, we also write $\mathcal{L}E = \mathcal{L}E_1 \cup \mathcal{L}E_2$. The simulator is given in Fig. 6. It consists of four interfaces: $\mathcal{S}_1/\mathcal{S}_1^{-1}$ corresponding to E_1/E_1^{-1} , and $\mathcal{S}_2/\mathcal{S}_2^{-1}$ corresponding to E_2/E_2^{-1} .

Again, apart from the **if**-clause of lines 02-06, the simulator identically mimics an ideal cipher. In this particular clause, the simulator checks whether a query (k, m) may appear in an MDC-4 evaluation (see Fig. 5) as a bottom query (left or right) for some other pair of queries appearing in the top. In this case, the simulator should consult \mathcal{R} to derive the query response.

<p>Forward Query $\mathcal{S}_j(k, m)$ ($j \in \{1, 2\}$)</p> <pre> 00 if $\mathcal{L}E_j^+[k](m) \neq \perp$ return $c = \mathcal{L}E_j^+[k](m)$ 01 $c \xleftarrow{\\$} \{0, 1\}^n \setminus \mathcal{L}E_j^+[k]$ 02 if $\exists (u, w, c_1) \in \mathcal{L}E_1, (v, w, c_2) \in \mathcal{L}E_2 : \dots$ 03 $\dots m = u[j = 2] + v[j = 1]$ and $k = c_j^1 \ c_j^2 + w$ 04 $(y, z) \leftarrow \mathcal{R}(u, v, w)$ 05 $c \leftarrow m + (y[j = 2] + z[j = 1])$ 06 end if 07 return $\mathcal{L}E_j^+[k](m) \leftarrow c$ </pre>	<p>Inverse Query $\mathcal{S}_j^{-1}(k, c)$ ($j \in \{1, 2\}$)</p> <pre> 10 if $\mathcal{L}E_j^-[k](c) \neq \perp$ return $m = \mathcal{L}E_j^-[k](c)$ 11 $m \xleftarrow{\\$} \{0, 1\}^n \setminus \mathcal{L}E_j^-[k]$ 12 return $\mathcal{L}E_j^-[k](c) \leftarrow m$ </pre>
--	---

Fig. 6. The simulator \mathcal{S} for E used in the proof of Thm. 2. Here, $\bar{j} \in \{1, 2\}$ is the complement of $j \in \{1, 2\}$.

9 Conclusions

Being the only known double length compression function that achieves optimal collision security and a non-trivial indifferentiability bound, Mennink’s compression function class appears to be stronger than its alternatives. Yet, this additional level of security does not come for free: the function makes three block cipher calls, rather than “the usual” two, which are moreover not parallelizable. It would be of both theoretical and practical interest to derive a two-call compression function (for either choice of k) with the same or even better security guarantees.⁶ We note, however, that the indifferentiability proof in this work relies on the presence of the third block cipher call, and all attacks on functions with $k = 2n$ rely on the fact that these make only two primitive calls.

Acknowledgments. This work has been funded in part by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, and in part by the Research Council K.U.Leuven: GOA TENSE. The author is supported by a Ph.D. Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). The author would like to thank Atul Luykx for his valuable help and feedback.

References

1. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
2. Armknecht, F., Fleischmann, E., Krause, M., Lee, J., Stam, M., Steinberger, J.: The preimage security of double-block-length compression functions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 233–251. Springer, Heidelberg (2011)

⁶ Without going into detail, we refer to a slightly related work of Maurer and Tessaro [19] on indiffereniable domain extenders from random functions.

3. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
4. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
5. Fleischmann, E., Gorski, M., Lucks, S.: Security of cyclic double block length hash functions. In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 153–175. Springer, Heidelberg (2009)
6. Hirose, S.: Provably secure double-block-length hash functions in a black-box model. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 330–342. Springer, Heidelberg (2005)
7. Hirose, S.: Some plausible constructions of double-block-length hash functions. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
8. Hirose, S., Park, J.H., Yun, A.: A simple variant of the merkle-damgård scheme with a permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
9. Jetchev, D., Özen, O., Stam, M.: Collisions are not incidental: A compression function exploiting discrete geometry. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 303–320. Springer, Heidelberg (2012)
10. Kuwakado, H., Morii, M.: Indifferentiability of single-block-length and rate-1 compression functions. IEICE Transactions 90-A(10), 2301–2308 (2007)
11. Lai, X., Massey, J.L.: Hash functions based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
12. Lee, J., Kwon, D.: The security of Abreast-DM in the ideal cipher model. Cryptology ePrint Archive, Report 2009/225 (2009)
13. Lee, J., Stam, M.: MJH: A faster alternative to MDC-2. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 213–236. Springer, Heidelberg (2011)
14. Lee, J., Stam, M., Steinberger, J.: The collision security of tandem-DM in the ideal cipher model. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 561–577. Springer, Heidelberg (2011)
15. Lee, J., Stam, M., Steinberger, J.: The preimage security of double-block-length compression functions. Cryptology ePrint Archive, Report 2011/210 (2011)
16. Lee, J., Steinberger, J.: Multi-property-preserving domain extension using polynomial-based modes of operation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 573–596. Springer, Heidelberg (2010)
17. Lucks, S.: A collision-resistant rate-1 double-block-length hash function (Symmetric Cryptography, Dagstuhl Seminar Proceedings 07021) (2007)
18. Maurer, U.M., Renner, R.S., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
19. Maurer, U.M., Tessaro, S.: Domain extension of public random functions: Beyond the birthday barrier. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 187–204. Springer, Heidelberg (2007)
20. Mennink, B.: Optimal collision security in double block length hashing with single length key. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 526–543. Springer, Heidelberg (2012)

21. Mennink, B.: On the collision and preimage security of MDC-4 in the ideal cipher model. In: *Designs, Codes and Cryptography* (to appear, 2013)
22. Meyer, C., Schilling, M.: Secure program load with manipulation detection code. In: *Proc. Securicom.*, pp. 111–130 (1988)
23. Özen, O.: *Design and Analysis of Multi-Block-Length Hash Functions*. PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne (2012)
24. Özen, O., Stam, M.: Another glance at double-length hashing. In: Parker, M.G. (ed.) *Cryptography and Coding 2009*. LNCS, vol. 5921, pp. 176–201. Springer, Heidelberg (2009)
25. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) *CRYPTO 1993*. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
26. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011)
27. Stam, M.: Beyond uniformity: Better security/Efficiency tradeoffs for compression functions. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 397–412. Springer, Heidelberg (2008)
28. Stam, M.: Blockcipher-based hashing revisited. In: Dunkelman, O. (ed.) *FSE 2009*. LNCS, vol. 5665, pp. 67–83. Springer, Heidelberg (2009)
29. Steinberger, J.P.: The collision intractability of MDC-2 in the ideal-cipher model. In: Naor, M. (ed.) *EUROCRYPT 2007*. LNCS, vol. 4515, pp. 34–51. Springer, Heidelberg (2007)

Security Amplification against Meet-in-the-Middle Attacks Using Whitening

Pierre-Alain Fouque^{1,2} and Pierre Karpman³

¹ Université de Rennes 1, France

² Institut universitaire de France, France

`pierre-alain.fouque@irisa.fr`

³ École normale supérieure de Cachan, antenne de Bretagne, France

`pierre.karpman@gmail.com`

Abstract. In this paper we introduce a model for studying meet-in-the-middle attacks on block ciphers, and a simple block cipher construction provably resistant to such attacks in this model. A side-result of this is a proper formalization for an unproven alternative to DESX proposed by Kilian and Rogaway; this construction can now be shown to be sound in our model. Meet-in-the-middle attacks exploit weaknesses in key schedule algorithms, and building constructions resistant to such attacks is an important issue for improving the security of block ciphers. Our construction is generic so that it can be used on top of any block cipher, and it does not require to increase the key-length. We use an *exposure resilient function* (or ERF) as a building block and we propose a concrete and efficient instantiation strategy based on compression functions.

Keywords: Block cipher, meet-in-the-middle attack, provable security, exposure resilient function.

1 Introduction

In the area of block cipher design, much work up to now has been devoted to proving resistance to classical statistical attacks like standard linear and differential cryptanalysis (see *e.g.* [33,38,13]). However, resistance to attacks that exploit weaknesses of the key schedule has remained mainly unaddressed. These attacks consist principally in meet-in-the-middle (MiTM) [15] and related-key attacks [4].

A typical good design criterion for key schedules is to have a high minimal distance between expanded keys; performance is also often another issue, and key schedules are expected to be fast, so as not to impact too much the encryption of small messages. An additional criterion could be for the key schedule to be non-linear, although many (good) key schedules are in fact linear. These design principles, however, do not really amount to a theory comparable to the one devoted to resistance to statistical attacks. Nonetheless, a few works study the security of key schedules with respect to related-key attacks from a more theoretical perspective [31,30,12].

Meet-in-the-middle attacks are an important technique available to the cryptanalysts studying symmetric primitives. It is important to avoid such attacks since unlike statistical attacks they usually have low data requirements. Beyond the classical result on double-encryption (see *e.g.* [15]), MiTM attacks are effective at exploiting weaknesses in the key schedule algorithms of block ciphers or the message expansion of hash functions. In the context of block ciphers, MiTM attacks are the most efficient attacks on many ciphers: round-reduced version of IDEA [5], round-reduced version of AES [14,17,9], or the full GOST [21]. Furthermore, a MiTM phase is usually used to extend the number of rounds reached by a statistical attack, as seen for instance in the attacks of Biham and Shamir [6] and of Matsui [29] on the DES. Finally, the recent biclique attack is a MiTM-related technique useful to speed up exhaustive search. Bicliques were found to be successful against AES and IDEA [7,23]. In the context of hash functions, MiTM techniques may be used to find preimages; this was for instance the case for attacks on MD4 [26,2], MD5 [37,2], or AES in a hash function setting [36].

Our Contributions. In this paper, we develop a simple model for meet-in-the-middle attacks and propose a generic block cipher construction that is provably resistant to such attacks in this model. The idea behind our model is simple and based on the fact that many MiTM attacks on block ciphers can be seen as decomposing the cipher into two sub-ciphers, and then applying the classical MiTM attack on double-encryption (in more recent variations, the cryptanalyst may also guess part of the key or part of the intermediate state [21,16]). Hence we argue that studying a construction in the sole context of double-encryption is actually meaningful for studying many types of MiTM attacks on a single cipher vulnerable to such attacks. However our goal is not to study constructions *actually based on double-encryption* (such as for instance the double XOR-cascade [20], *cf.* below). This is because such constructions already lend themselves to meet-in-the-middle attacks even when the underlying cipher(s) does not; our objective is different and consists in obtaining a *construction resistant to MiTM when the underlying cipher is not*. Studying the construction with a composition of two ciphers as an underlying primitive is therefore only a *mean of simulating the construction applied to a single cipher that is vulnerable to meet-in-the-middle attacks*.

Our construction relies on a core (or internal) cipher and on a form of whitening. Let \mathcal{E}_k be the core cipher of key k , and f a function with good enough properties, then define a new cipher

$$\text{EF}_k(p) \triangleq \mathcal{E}_k(p \oplus f(k)) \oplus f(k).$$

The main idea behind this construction is to force an attacker to commit to a value for the whole key before being able to exploit any data he may have access to, thereby making it impossible to work separately on parts of the key¹.

¹ Or alternatively to force the attacker to guess the value of the whitening independently of the key.

A similar idea can be found for instance in the operation mode of the SHA-3 candidate SIMD [27]: the goal in this context was to make message modification techniques *à la* Wang impossible by forcing the attacker to commit to a specific value of the message, before the message expansion phase. We prove that meet-in-the-middle attacks are not effective against the EF cipher; this is achieved by upper-bounding the maximum advantage of an attacker of the above construction in a double-encryption setting (when \mathcal{E} is a cascade of two ciphers), and showing that it is less than the advantage of a meet-in-the-middle adversary. We do this with a method adapted from Kilian and Rogaway’s proof on DESX, and justify formally how this is relevant when the construction is applied to a single cipher on which MiTM attacks may be performed. We also discuss how the construction can be instantiated efficiently in practice.

Related Work. We can distinguish two kinds of works on provable security for block ciphers: proving a property for some high-level and generic construction, or proving the resistance of an actual cipher to more specific attacks. Our work clearly belongs to the first category, whereas from the second we can cite *e.g.* the provable resistance of block ciphers to classical linear and differential cryptanalysis. Our approach is generic so that it can be used on top of any cipher; it is for instance similar to some proposals for building tweakable block ciphers [28].

Similar-looking constructions have already been proposed in the literature, but with a distinct motivation of extending the *equivalent key length* of the core primitive. One such construction is the DESX (or its more generic name ‘FX’) construction, proposed by Rivest and formally proved by Kilian and Rogaway [25]. It can be described as taking a cipher \mathcal{E}_k under the key k , and defining a new cipher

$$\text{EX}_{k,k_1,k_2}(p) \triangleq \mathcal{E}_k(p \oplus k_1) \oplus k_2.$$

A more recent development is the aforementioned double XOR-cascade of Gaži and Tessaro [20]. This construction is based on a cipher \mathcal{E} and defines a new cipher 2XOR as:

$$\text{2XOR}_{k,k_1}(p) \triangleq \mathcal{E}_{\tilde{k}}(\mathcal{E}_k(p \oplus k_1) \oplus k_1),$$

where \tilde{k} is a key related to k (the authors suggest flipping one bit of k). However, this requires two calls to the cipher \mathcal{E} , and therefore cannot readily be used in our context. The main difference between the above constructions and ours is that we do not aim for a bigger equivalent key length, and derive all the whitening keys from the original key to \mathcal{E} . We also study our construction specifically in the context of resistance to meet-in-the-middle attacks.

Interestingly, Kilian and Rogaway briefly mention a construction that can be seen as an instantiation of ours. Their purpose was to define an alternative to the FX construction that gives more flexibility in the choice of the key length to the user. Instead of using independent keys k, k_1, k_2 , they suggest deriving them from an arbitrarily-long key \hat{k} , as the (truncated) output of $f(\hat{k}), f_1(\hat{k})$, and $f_2(\hat{k})$ respectively, where f, f_1, f_2 are defined as SHA-1 prefixed with three different constants. Once again the motivation is different from ours, and no proof nor

formalization is given for this construction. Note that as a consequence of our results, it is possible to prove that this construction is sound.

Outline of the Paper. We present our model for studying meet-in-the-middle attacks in §2 and our construction in §3. We prove the resistance of the construction to the attacks captured by our model in §4. We discuss our result and its implications on advanced meet-in-the-middle techniques in §5, and instantiation issues in §6.

2 A Model for Meet-in-the-Middle Attacks

2.1 Generic Constructions

The aim of our work is to define constructions resistant to MiTM attacks. We define here what we mean by *construction* and what kinds of constructions we specifically consider. We first recall the definition of a block cipher.

Definition 1. *A block cipher is a mapping $\mathcal{E} : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that $\forall k \in \{0, 1\}^\kappa$, $\mathcal{E}(k, \cdot)$ (also noted $\mathcal{E}_k(\cdot)$) is a permutation. The first and second arguments of \mathcal{E} are called the key and message (block) respectively. The variables κ and n denote the key size (or length) and block size (idem) of \mathcal{E} .*

Definition 2. *A single-cipher construction is a block cipher $E : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ that can be described as the composition $\text{Post} \circ \mathcal{E} \circ \text{Pre}$ of functions Pre , \mathcal{E} , and Post , where $\mathcal{E} : \{0, 1\}^{\kappa'} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a block cipher. The functions Pre and Post may take both of \mathcal{E} 's inputs as arguments, that is the key and the message. A ν -cipher construction is a block cipher that can be described as the composition $\text{Post}^{(\nu)} \circ \mathcal{E}^{(\nu)} \circ \text{Pre}^{(\nu)} \circ \text{Post}^{(\nu-1)} \circ \dots \circ \mathcal{E}^{(1)} \circ \text{Pre}^{(1)}$ where the ciphers $\mathcal{E}^{(i)}$ use independent keys, and where the $\text{Pre}^{(i)}$ and $\text{Post}^{(i)}$ functions may take any of these keys as inputs. A ν -cipher construction of the specific form $\text{Post} \circ \mathcal{E}^{(\nu)} \circ \mathcal{E}^{(\nu-1)} \circ \dots \circ \mathcal{E}^{(1)} \circ \text{Pre}$ is called a ν^* -cipher construction. Any single-cipher construction can be extended to a ν^* -cipher construction in a straightforward way.*

In this paper, we consider the EF construction, defined in §3, which is a single-cipher construction. It is thus generic, and can be used both with already-existing algorithms, and as a basis to design a cipher *ex nihilo*.

2.2 The Model

Our goal in this section is to give a formal model for MiTM attacks that allows to prove security properties. We later use this model to prove the resistance of the EF construction to such attacks. This model does not capture the concept of *any* MiTM attack, but it does nonetheless take into account a significant class.

The idea behind our model is that a MiTM attack on, say, cipher \mathcal{E} often performs a conceptual decomposition of \mathcal{E} into two sub-ciphers, with separate key

bits. We can thus model such attacks as being performed on a double-encryption construction with two independent ciphers, seen as black-boxes. This allows us to consider MiTM attacks against generic ciphers. We detail this argument in the remainder of this section.

The Classical Meet-in-the-Middle Attack on Double Encryption. Let us consider the cipher \mathcal{E} , defined as the composition of the two independent ciphers \mathcal{F} and \mathcal{G} , operating on independent keys k_1 and k_2 respectively. We denote by $\mathcal{E}_k(p)$ the action of encrypting the plaintext p with \mathcal{E} and key k and producing the ciphertext c . By definition of \mathcal{E} , we have $\mathcal{E}_k(p) \triangleq \mathcal{G}_{k_2}(\mathcal{F}_{k_1}(p))$, with k_1 and k_2 uniquely defined by k .

The MiTM attack on double-encryption exploits the fact that k_1 and k_2 are used independently in their respective ciphers; in its simplest form, it can be described as follows:

```

Get a known plaintext  $p$  and its corresponding ciphertext  $c$ .
for every possible candidate  $k_1^i$  for key  $k_1$  do
    Compute  $y^i \triangleq \mathcal{F}_{k_1^i}(p)$  and store the result in memory.
end for
for every possible candidate  $k_2^j$  for key  $k_2$  do
    Compute  $y'^j \triangleq \mathcal{G}_{k_2^j}^{-1}(c)$  and store the result in memory.
end for
for every  $y^i = y'^j$  do
    Output  $(k_1^i, k_2^j)$  as a candidate for  $(k_1, k_2)$ .
end for

```

This procedure may be repeated until only one candidate for k_1 and k_2 remains, using many plaintext/ciphertext pairs. If we call κ the size of the keys k_1 and k_2 in bits, the cost of the attack is then of the order 2^κ in time and memory, which is much lower than the $2^{2\kappa}$ time that brute-force search on k would cost. If k_1 and k_2 are of different size κ_1 and κ_2 , one needs only to store the candidates for the smaller size, *e.g.* in a hash table, and the candidates for the bigger key size can be computed on the fly. The general cost of this attack is thus of the order of $\max(2^{\kappa_1}, 2^{\kappa_2})$ in time, and $\min(2^{\kappa_1}, 2^{\kappa_2})$ in memory.

This attack can still be applied when k_1 and k_2 are not independent but have only some of their bits in common. In that case, one just needs to guess the common bits and repeat the above procedure for every guess.

A Model for Meet-in-the-Middle Attacks on a Single Cipher. MiTM attacks are in no way limited to double-encryption; in fact they are well-suited to iterated ciphers with weak key schedules. However, the ideas involved in a MiTM attack on a single cipher are essentially the same as for attacking double-encryption.

Let us consider an iterated cipher \mathcal{E} with round function ρ^i for round i : we define $\mathcal{E}_k(p)$ as the composition $\rho_{k_r}^r \circ \dots \circ \rho_{k_1}^1(p)$, where r is the number of iterations of the round function, and the k_i 's are round keys generated from k

by a key schedule. The idea behind a MiTM attack on \mathcal{E} is to find two sets k_α and k_β of consecutive round keys such that they involve strictly different bits of k . In other words we have $k_\alpha \triangleq \{k_i, \dots, k_{i+j}\}$, $k_\beta \triangleq \{k_{i+j+1}, \dots, k_{i+j+1+k}\}$, with $k_\alpha \cap k_\beta = \emptyset$ (when the intersection is taken over the bits of k found in k_α and k_β). Once these sets are found, it is possible to guess independently the bits of k present in k_α and the ones present in k_β , in a way exactly similar as for double-encryption. That is, finding the sets is equivalent to conceptually decompose (a part of) \mathcal{E} in two sub-ciphers with independent keys, on which double-encryption is performed: we have $\mathcal{E}_k = (\rho^{i+j+1+k} \circ \dots \circ \rho^{i+j+1})_{k_\beta} \circ (\rho^{i+j} \circ \dots \circ \rho^i)_{k_\alpha}$ (although this equality is true only if $i = 1$ and $i + j + 1 + k = r$. This constraint can easily be waved, however, if we restrict ourselves to finding sub-ciphers for a round-reduced version of \mathcal{E}).

We are now ready to formalize our model. We start by stating the security of double-encryption with ideal ciphers, thanks to a theorem of Aiello *et al.* [1]².

Theorem 1 ([1]). *Let $\mathcal{F} : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be an ideal block cipher. For any $\kappa, n, t, q \geq 1$, where t and q denote the number of oracle access to \mathcal{F} and \mathcal{F}^{-1} , and $\mathcal{F} \circ \mathcal{F}$ respectively, then the maximum advantage of any adversary $A_{t,q}$ trying to distinguish $\mathcal{F} \circ \mathcal{F}$ from a random permutation with resources t and q is upper-bounded by $t^2/2^{2\kappa}$. This bound is tight up to a constant factor as long as q is not too small.*

We allow ourselves to use a more general expression of this result when considering double-encryption of not necessarily equal ciphers \mathcal{F} and \mathcal{G} , of not necessarily equal key lengths κ_1 and κ_2 , where t_1 and t_2 denote the number of oracle access to \mathcal{F} and \mathcal{G}^{-1} respectively. In this case, we use the upper-bound $t_1 \cdot t_2/2^{\kappa_1+\kappa_2}$ ³. We now define the notion of constructions resistant to MiTM attacks.

Definition 3. *Let $\mathcal{F} : \{0, 1\}^{\kappa_1} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ and $\mathcal{G} : \{0, 1\}^{\kappa_2} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be two block ciphers. A two-cipher construction $E(\mathcal{F}, \mathcal{G})$ is said to be resistant to the meet-in-the-middle attack if the maximum advantage of any adversary A trying to distinguish $E(\mathcal{F}, \mathcal{G})$ from a random permutation is:*

$$\max_A Adv_{E(\mathcal{F}, \mathcal{G})}(A_{t_1+t_2, q}) < t_1 \cdot t_2/2^{\kappa_1+\kappa_2},$$

up to constant factors.

This definition is made meaningful by the tightness of the bound of theorem 1. Essentially, it says that a two-cipher construction is resistant to meet-in-the-middle attacks if no adversary can distinguish it with an advantage that is *at least as good as the best one it could get if only composition of the two ciphers were used instead.*

² The result is stated in the specific case where the two ciphers are equal.

³ Although in practice, we actually study our construction in the case of $\kappa_1 = \kappa_2$, and therefore we will really be using exactly the same bound as in the main result of [1].

Definition 4. Let $\mathcal{E} : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a block cipher. A single-cipher construction $E(\mathcal{E})$ is said to be resistant to the meet-in-the-middle attack if for any two block ciphers $\mathcal{F} : \{0, 1\}^{\kappa_1} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ and $\mathcal{G} : \{0, 1\}^{\kappa_2} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that $\mathcal{E} = \mathcal{G} \circ \mathcal{F}$ and $\kappa = \kappa_1 + \kappa_2$, the maximum advantage of any adversary A trying to distinguish $E(\mathcal{E})$ from a random permutation is:

$$\max_A \text{Adv}_{E(\mathcal{E})}(A_{t_1+t_2, q}) < t_1 \cdot t_2 / 2^{\kappa_1 + \kappa_2},$$

up to constant factors.

In other words, this means that the best attack on the construction $E(\mathcal{E})$ is strictly worse than the best meet-in-the-middle attack on \mathcal{E} . Our model is justified by the next proposition:

Proposition 1. Let E be a two*-ciphers construction resistant to the meet-in-the-middle attack. Then the restriction of E to a single cipher is a single-cipher construction resistant to the meet-in-the-middle attack.

Proof. This is a direct consequence of definitions 3 and 4. □

Hence, the resistance of a single-cipher construction to MiTM attacks can be studied by analyzing its two*-ciphers variant. In practice, we perform this analysis in the ideal block cipher model [3,25].

3 A Construction Resistant to Meet-in-the-Middle Attacks

We now formally define our construction. We start by introducing the notion of *Exposure-Resilient Functions* (or ERF), as defined by Canetti et al. [11]. ERFs are similar to *all or nothing transforms*, which were introduced by Rivest [35].

Definition 5. An ℓ -ERF is a mapping $f : \{0, 1\}^\alpha \rightarrow \{0, 1\}^\beta$ such that for random values r, R in $\{0, 1\}^\alpha, \{0, 1\}^\beta$; for any $L \in \binom{[\alpha]}{\ell}$, the distributions $\langle [r]_{\bar{L}}, f(r) \rangle$ and $\langle [r]_{\bar{L}}, R \rangle$ are indistinguishable one from another; where $\binom{[\alpha]}{\ell}$ denotes the set of subsets of $\{1 \dots \alpha\}$ of size ℓ , and for $x \in \{0, 1\}^\alpha$, $[x]_{\bar{L}}$ denotes x restricted to its bits not in L .

Here, we will consider particularly weak ERFs, in the sense that when less than ℓ bits of r are unknown to the adversary, he is then supposed to be able to predict the value of $f(r')$ for any r' that fixes the unknown bits of r to some value. However, we will mostly consider degenerate cases where ℓ is zero. That is, the output of the ERF is indistinguishable from random until all of its input is revealed. Constructions of ERFs are known to exist in the standard model [11], and it is trivial to see that a random oracle meets the definition of a zero-ERF. Hence, from a practical point of view, a zero-ERF can be instantiated by a hash function, in the random oracle model.

We now define our construction, which we will note EF for short.

Definition 6. Let $\mathcal{E} : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a block cipher. We write $\mathcal{E}_k(p) \triangleq c$ the action of encrypting the plaintext p with \mathcal{E} under the key k , to produce the ciphertext c . Let $f : \{0, 1\}^\kappa \rightarrow \{0, 1\}^n$ be an ℓ -ERF. Then we define the EF construction with core cipher \mathcal{E} as $\text{EF}_k(p) \triangleq \mathcal{E}_k(p \oplus f(k)) \oplus f(k)$, where ‘ \oplus ’ denotes bitwise exclusive or.

Our goal is to prove the resistance of this construction to MiTM attacks. According to our model, we will study this construction as a two-ciphers construction. That is, we instantiate \mathcal{E}_k by $\mathcal{G}_{k_2} \circ \mathcal{F}_{k_1}$. In this case, the EF construction applied to \mathcal{E} can be written as:

$$\text{EF}_{(k_1, k_2)}(p) \triangleq \mathcal{G}_{k_2}(\mathcal{F}_{k_1}(p \oplus f(k_1, k_2))) \oplus f(k_1, k_2).$$

4 Resistance of the EF Construction to Meet-in-the-Middle Attacks

In this section, we prove an upper-bound on the advantage of an adversary trying to distinguish the EF two-ciphers construction from a random permutation, in function of the number of queries made to different oracles. The bound we obtain shows that our construction significantly improves the resistance of double-encryption to generic attacks such as the classical MiTM, and hence is resistant to the MiTM attack, in the terminology of definitions 3 and 4.

4.1 Security Model

We consider the EF two-ciphers construction applied to $\mathcal{G} \circ \mathcal{F}$, where $\mathcal{F} : \{0, 1\}^{\kappa_1} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ and $\mathcal{G} : \{0, 1\}^{\kappa_2} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ are ideal block ciphers [3,25]: for each key k_1 (resp. k_2), the map \mathcal{F}_{k_1} (resp. \mathcal{G}_{k_2}) is a permutation randomly chosen from the set Π_n of all $(2^n)!$ permutations operating on words of size n . For ease of presentation, and without loss of generality, we assume that $\kappa_1 = \kappa_2 \triangleq \kappa$. The ‘ f ’ function used in EF is an ℓ -ERF with ℓ small. We consider an adversary who is given access to four oracles:

- Two of them are \mathcal{F} and \mathcal{G} ; when provided with a key k'_1 (resp. k'_2) of size κ and an input x (resp. y) of size n , they return the result y (resp. z) of encrypting x (resp. y) with \mathcal{F} (resp. \mathcal{G}) with key k'_1 (resp. k'_2). Queries to the inverse oracles \mathcal{F}^{-1} and \mathcal{G}^{-1} are permitted, and are not distinguished from regular queries. That is, if $\mathcal{F}(x) \triangleq y$ has been queried, we consider that $\mathcal{F}^{-1}(y) \triangleq x$ has also been queried, and conversely.
- One oracle gives an access to f , and when provided with an input of size 2κ bits, returns the result of size n of the evaluation of f on this input.
- The last oracle, which we call \mathcal{U} , takes as input a plaintext p and returns either $\text{EF}(p)$, with EF instantiated with \mathcal{F} , \mathcal{G} , and f with fixed, randomly chosen keys k_1 and k_2 ; or the image of p from a fixed permutation π randomly selected from the set Π_n . Again, queries to \mathcal{U}^{-1} are permitted.

Each access to an oracle will be counted and expressed by the following variables:

- The number of accesses to \mathcal{U} is denoted by D . They represent the amount of data available to the adversary.
- The number of accesses to \mathcal{F} and \mathcal{F}^{-1} (resp. \mathcal{G} and \mathcal{G}^{-1}) is denoted by q_1 (resp. q_2).
- The number of accesses to f is denoted by q_f .

The goal of the adversary is to distinguish between \mathcal{U} being π or the EF construction. We define the *advantage* of an adversary as its probability of successfully distinguishing the two instantiations of \mathcal{U} . More formally:

Definition 7. Let Π_n be the set of permutations on words of size n ; let us note $x \stackrel{R}{\leftarrow} \mathcal{S}$ the action of randomly choosing an object x from the set \mathcal{S} ; let us denote by A^{EF} and A^π the answer, 0 or 1, of an adversary A with access to the aforementioned oracles, when \mathcal{U} is the EF construction and a randomly chosen permutation respectively. Then the advantage Adv_A of the adversary A is defined as:

$$\begin{aligned} \text{Adv}_A &\triangleq \\ &\Pr [\forall k'_1 \in \{0, 1\}^\kappa, \mathcal{F}_{k'_1} \stackrel{R}{\leftarrow} \Pi_n; \forall k'_2 \in \{0, 1\}^\kappa, \mathcal{G}_{k'_2} \stackrel{R}{\leftarrow} \Pi_n; \pi \stackrel{R}{\leftarrow} \Pi_n : A^\pi = 1] - \\ &\Pr [\forall k'_1 \in \{0, 1\}^\kappa, \mathcal{F}_{k'_1} \stackrel{R}{\leftarrow} \Pi_n; \forall k'_2 \in \{0, 1\}^\kappa, \mathcal{G}_{k'_2} \stackrel{R}{\leftarrow} \Pi_n; k_1 \stackrel{R}{\leftarrow} \{0, 1\}^\kappa; \\ &\quad k_2 \stackrel{R}{\leftarrow} \{0, 1\}^\kappa : A^{\text{EF}} = 1]. \end{aligned}$$

Our objective is to bound this advantage for any adversary A in function of the number of queries he has made to the oracles. We note $\text{Adv}(\ell, D, q_1, q_2, q_f)$ the advantage of any adversary having made less than D , q_1 , q_2 , and q_f queries to the oracles, when f is an ℓ -ERF.

Notations.

- A *key candidate* for \mathcal{F} (resp. \mathcal{G}) is denoted by k'_1 (resp. k'_2). Those are keys for which the adversary makes queries to the oracles awaiting a key as part of their inputs.
 Moreover, we may count the accesses to those oracles when queried with a specific key. The number of accesses to \mathcal{F} (resp. \mathcal{G} , f) with key k'_1 (resp. k'_2 , (k'_1, k'_2)) is written $q_1(k'_1)$ (resp. $q_2(k'_2)$, $q_f(k'_1, k'_2)$).
 Furthermore, the number of queries to the \mathcal{F} (resp. \mathcal{G}) oracle with key k'_1 (resp. k'_2) and for a specific message p is noted $q_1(k'_1, p)$ (resp. $q_2(k'_2, p)$).
- If \mathcal{U} has been instantiated with EF, the key used in the instantiation for \mathcal{F} (resp. \mathcal{G}) is denoted by k_1 (resp. k_2). For ease of presentation, we will consider it valid to talk about those keys even when it is not clear if \mathcal{U} is instantiated with EF.
- We denote by x, y, z , the intermediate values $p \oplus f(k_1, k_2)$, $\mathcal{F}_{k_1}(p \oplus f(k_1, k_2))$, and $\mathcal{G}_{k_2}(\mathcal{F}_{k_1}(p \oplus f(k_1, k_2)))$ respectively.
- We make use of an *indicator function*, written $\mathbb{1}$. We have $\mathbb{1}(x) = 0$ if and only if x is zero, and it is one otherwise. This may be extended to sets, where $\mathbb{1}(x) = 0$ if and only if x is the empty set, and is one otherwise.

- The concatenation of words x and y is noted $x||y$, the Hamming weight of a word x is denoted by $\text{hw}(x)$.

4.2 The Result

Our main result about the security of the EF construction is summarized by the following theorem.

Theorem 2. *The advantage $\text{Adv}(\ell, D, q_1, q_2, q_f)$ of an adversary trying to distinguish EF from a random permutation is upper-bounded by:*

$$2^{-2\kappa} \max \left(2^\ell \binom{n}{\ell} \cdot q_f, \quad 2^{-n} \cdot D \sum_{k'_1, k'_2} \min(q_1(k'_1), q_2(k'_2)) \right). \quad (1)$$

One can see that to gain an advantage of one, an adversary with D available data needs at least $\frac{2^{2\kappa}}{2^\ell \binom{n}{\ell}}$ or $\frac{2^{\kappa+n}}{D}$ queries to the oracles, whichever is the smallest. For $\ell = 0$, $n = \kappa$, and $D = 1$, these two terms are equal to $2^{2\kappa}$; in the case of double-encryption without the EF construction, one would only need of the order of 2^κ queries. This result immediately implies the following corollary:

Corollary 1. *The two*-ciphers and single-cipher EF construction is resistant to meet-in-the-middle attacks, in the terminology of definitions 3 and 4, as long as $D < 2^n$ (i.e. the adversary does not have access to the whole codebook).*

Proof. For small values of ℓ , and when $D < 2^n$, $\text{Adv}(\ell, D, q_1, q_2, q_f)$ is strictly smaller than the bound of definitions 3 and 4. \square

The restriction that $D < 2^n$ is important, and comes from more general properties of FX-like constructions. In particular, such a construction cannot be used in all generality in order to increase the equivalent key length of a block cipher, as the key length cannot be shown to be more than the one of the core cipher when all the codebook is available to the adversary. This is the case for the original construction of DESX, and remains true in our modified setting when the core cipher is a 2-cascade (i.e. a composition of two ciphers). In the latter case, the double XOR-cascade of Gaži and Tessaro *does* increase the effective key length of the (modified) 2-cascade. However, as it has already been noted, we want our construction to be applicable to a single cipher as well, and therefore cannot use one similar to theirs. The above restriction notwithstanding, we believe that our construction is still interesting in practice. The first reason is that attacks where the adversary uses the whole codebook would not only be of limited interest to the attacker, they can also be made asymptotically as expensive as the designer of the cipher wishes to; so big an amount of data is also fairly unrealistic for many ciphers with big block sizes (e.g. 128 bits). The second reason is that raising the data complexity of a MiTM attack from the information-theoretically lower-bound to the whole codebook, for an adversary to get the same time complexity, is in itself a huge improvement of the resistance to MiTM attacks.

4.3 Proof Sketch

We outline here the strategy used for proving theorem 2. The complete proof is given in the full version of the paper [19].

Given the similarities between the EF and FX constructions and their security models, our proof has a structure close to the one of Kilian and Rogaway [25]. In particular, we use games for each oracle to define the situations where an adversary may distinguish the instantiation of the \mathcal{U} oracle (we do not fully redefine the games in the proof, though, and refer to [25] for a more detailed description). For each such situation, we then bound the probability of the distinction being possible in function of the past queries made by the adversary. We bound separately the advantage of an adversary trying to find distinguishing situations for the three oracles, \mathcal{F} , \mathcal{G} , and \mathcal{U} , and then combine these bounds together with the bound of the advantage over f to produce a general result.

It should be noticed that we do not need to consider situations for the “inverse” oracles (such as *e.g.* \mathcal{F}^{-1}), as the constraints possibly put on the input/output pairs of queries to those oracles are simply swapped when compared to the ones for the “forward” oracles (an input of an inverse oracle being an output for the corresponding forward oracle). Therefore, the advantage when distinguishing inverse oracles is not different from when considering forward oracles.

The main difference between our proof and the one of [25] is that, from the structure of the construction, an adversary has essentially the choice of guessing the output of the f function directly (thereby seeing $f(k_1, k_2)$ as a third independent key), or via its inputs and the properties of f . This shows in the bound of theorem 2 as the two arguments of the maximum function. Because our construction is more complex than FX, we also have more oracles to consider.

5 Discussion

In the previous section, we have shown that the EF construction increases the resistance of a cipher to “classical” MiTM attacks. We now argue that this improved resistance carries on to more advanced MiTM techniques, which further increases the relevance and interest of the construction. We also address the relevance of our ideal-cipher-based model.

5.1 About Ideal Ciphers

As pointed above, our proof uses the ideal cipher model. This could be seen as limiting the relevance of the applications we claim —building constructions resistant to MiTM attacks— as we use a setting where the best attack on the sub-ciphers is basically brute-force.

We claim that this is not a limitation. Using ideal ciphers allows one to express bounds in terms of number of queries to the relevant oracles: the cost of each query is of little importance in so far as it is bounded by a constant. In other words, our results show that the EF construction increases the security of double-encryption by ensuring that an adversary needs *to perform more queries to the*

oracles to gain an advantage comparable to the one he would get when the construction is *not* used. Whether the queries are expensive or not (*i.e.* whether the best attacks on the sub-ciphers is brute-force or not) does not change the asymptotic increase of security that one can expect by choosing a bigger key or block size. This is as much valid for the sub-ciphers decomposition of a MiTM attack on single cipher as for double-encryption.

Another concern might be that actual MiTM attacks do not typically perform a full decomposition of the attacked cipher in two sub-ciphers with independent keys. This is indeed an ideal case for the attacker which is seldom met in practice. However, real attacks that do not conform to this ideal case are typically less powerful, while still needing a decomposition in two sub-ciphers at some point; hence our construction increases the security against these real attacks as well.

5.2 The Splice-and-Cut Exception

Before going on to the expected advantages of the EF construction, we should mention one situation where it does not seem to be useful, *i.e.* when protecting against splice-and-cut MiTM attacks.

A splice-and-cut MiTM attack uses a conceptual decomposition of a cipher with sub-ciphers that may be defined by considering the first and last round of the attacked cipher as consecutive. That is, we consider decompositions of say, \mathcal{E} , that can be written as, say $\mathcal{G}_2 \circ \mathcal{F} \circ \mathcal{G}_1$. In order to perform this variant of the MiTM attack, the attacker typically guesses one intermediate value at the boundary between the two sub-ciphers using different key subsets, and then queries the plaintext or ciphertext corresponding to the encryption or decryption of this value for a given sub-key candidate. With data obtained this way, it is then easy to perform a MiTM attack.

It is possible to adapt our model to the conceptual decomposition used in a splice-and-cut attack. However, because this attack typically requires the whole code-book to be performed, the bound that would be obtained by a generalization of theorem 2 would not show any improved resistance. It might be that improved resistance to the splice-and-cut MiTM could be shown by leaving the information-theoretic view of the ideal cipher model, but this does not seem to be an easy task.

In the end, because of their huge data requirements, splice-and-cut MiTM are more suited to attacking hash functions, and are seldom used against block ciphers. Consequently, we think that the impact of the EF construction not being efficient against them is somewhat limited in its targeted applications.

5.3 Taking Advanced Attack Techniques into Account

We now discuss the issue of including some more advanced MiTM techniques used by cryptanalysts in our proof that the EF construction increases the resistance of a single cipher to MiTM attacks. We discuss this for two important techniques: the *initial structure* of Sasaki and Aoki [37], and its later generalization to *bicliques* by Khovratovich *et al.* [24].

Initial Structure. This is an advanced techniques that may be used in order to increase the number of rounds reached by an existing MiTM attack. It consists in finding an *initial structure* between two sub-ciphers of the MiTM attack that use, say, key subsets k_1 and k_2 respectively. The structure consists itself in a sub-cipher that can be computed thanks to the key subsets $k'_1 \subseteq k_1$ and $k'_2 \subseteq k_2$, but with the key bits from k'_2 being used *before* the bits of k'_1 in the structure (otherwise the relevant parts of the initial structure can be included in the two sub-ciphers of the MiTM). Finding initial structures gives more flexibility to the attacker in the matching phase, leading to more powerful attacks. For instance, they were key in finding the first preimage attack on the full MD5 [37].

In order to be applicable, the initial structure technique still needs the cipher under attack to be decomposed in two sub-ciphers. The added sophistication of the matching phase typically allows the attacker to define a decomposition that covers more rounds of the full cipher than what he would obtain without using the initial structure. Yet this does not change the fact that the attacker needs to test the key candidates for both sub-ciphers. What the bounds on the EF construction say is that it is impossible to use a MiTM technique to do this efficiently, because of the increased number of queries that have to be made to the sub-ciphers (and possibly to the f function) in order to test all possible keys and get an advantage of one; this is completely independent of the number of rounds covered by the sub-ciphers.

In essence, the initial structure technique allows an attacker to find better decompositions, but it does not improve the key-testing phase *per se*. Therefore we claim that the EF construction still improves the security of a cipher against adversaries using initial structures: however good the decomposition of the initial cipher they may get, the construction layer will prevent efficient use of it.

Bicliques. This is a generalization of the previous technique that allows for a more systematic way to construct initial structures, instead of searching them manually as was done originally. In its simplest form, the biclique technique can be seen as a way to extend an existing (splice-and-cut) MiTM attack by constructing *bicliques* between intermediate states in order to cover the rounds not included in the existing attack. Again, this leads to more powerful attacks, and this technique has successfully been used to analyze several hash functions and block ciphers such as **Skein** [24], **AES** [7], or **IDEA** [23].

Again, for this technique to be applicable, finding a decomposition in sub-ciphers is necessary. Hence, even if the biclique parts themselves are not something that is captured in our analysis, it is still the case for the decomposition: for a given decomposition of given parameters size, the bound on the number of queries to the sub-ciphers necessary to gain a given advantage is not changed by the presence of bicliques. The interest of the EF construction in this case is again to ensure that no decomposition can be efficiently used, either alone or as part of a wider attack.

This analysis and its arguments is similar to the one that was performed on the recent **PRINCE** cipher in order to assess its resistance to bicliques [8].

Summary. We did not formalize the arguments presented in this section, and it does not seem to be easy to do so. In addition, although we think that an increased resistance to the MiTM phase is an important step towards some sort of provable security against these techniques, the interest of that in the case of actual designs might be somewhat more limited. The reason is that if these techniques allow to efficiently use a decomposition with small parameters size as part of a bigger attack, the increased resistance to the MiTM phase might not impact the overall complexity significantly. This might be a concern when resistance to *e.g.* bicliques is considered.

In the end, much work still needs to be done in order to better understand how to resist advanced MiTM techniques, and this is far beyond the scope of this paper. Yet we believe that an increased resistance to even just the basic MiTM attack should be an important part of this work, much as resisting standard statistical attacks is an important part of modern cipher design.

5.4 Alternatives for the Construction

We end this section by outlining three alternatives for the EF construction that differ with the main proposal in the way the output of the f function is combined with the input of the core cipher.

A first obvious variation is to use modular addition instead of XOR.

Definition 8. Let $f : \{0, 1\}^\kappa \rightarrow \{0, 1\}^n$ be an ℓ -ERF. Then we define the EF^\boxplus construction with core cipher \mathcal{E} as $\text{EF}_k^\boxplus(p) \triangleq \mathcal{E}_k(p \boxplus f(k)) \boxplus f(k)$, where \boxplus denotes addition in $\mathbb{Z}/2^n\mathbb{Z}$.

This variant may be useful in practice, when the core cipher of the construction is no longer seen as a black box. Because a block cipher typically performs key whitening with sub-keys derived from k , if it is performed with an XOR operation, it may be better to combine the output of f with modular addition, in order to make it non-linear with respect to this whitening⁴. Conversely, the original construction is maybe to be preferred when the whitening is done with modular addition. Note that adding whitening keys with modular addition instead of XOR was already proposed by Dunkelman *et al.* as a generalization of the Even-Mansour construction [18].

A second variation exploiting a similar idea is to replace XOR with multiplication in a finite field.

Definition 9. Let $f : \{0, 1\}^\kappa \rightarrow \{0, 1\}^n$ be an ℓ -ERF. Then we define the EF^\boxtimes construction with core cipher \mathcal{E} as $\text{EF}_k^\boxtimes(p) \triangleq \mathcal{E}_k(p \boxtimes f(k)) \boxtimes f(k)$, where \boxtimes denotes multiplication in \mathbb{F}_{2^n} .

Although quite slower than XOR or modular addition, multiplication in a finite field mixes its inputs very thoroughly, which makes it attractive when performance is not critical.

A last variation we suggest is to use even stronger mixing with a decorrelation module [38].

⁴ Alternatively, a concrete instantiation could use $f(k)$ as the unique whitening key.

Definition 10. Let $f, g : \{0, 1\}^\kappa \rightarrow \{0, 1\}^n$ be two ℓ -ERFs. Then we define the EFG^{DC} construction with core cipher \mathcal{E} as $\text{EFG}_k^{\text{DC}}(p) \triangleq \mathcal{E}_k(p \boxtimes f(k) \oplus g(k)) \boxtimes f(k) \oplus g(k)$.

Finally, we can also propose an obvious variation orthogonal to the three above, consisting in using two (four in the case of EFG^{DC}) different functions to derive the two whitening keys.

It is worthwhile noting that all these variants directly benefit from the proofs for the EF construction, as these did not rely on any specific property of XOR not shared by the alternative operations used here (in particular they are all invertible). However, the different constructions are likely to give different levels of security when used in practice, especially when other attacks than MiTM are considered. For instance, decorrelation modules are expected to provide some additional protection against classical differential attacks, which XOR or modular addition do not by themselves.

6 Practical Instantiation

We conclude this work by discussing how to efficiently instantiate the EF construction in practice. We start by showing how it is possible to use a hash function h as the f function. This is justified by the fact that such a function is a zero-ERF in the random-oracle model. We note EH the resulting construction.

Corollary 2. With notations adapted from §§3 and 4, the advantage $\text{Adv}(D, q_1, q_2, q_h)$ of an adversary trying to distinguish $\text{EH}_k(p) \triangleq \mathcal{E}_k(p \oplus h(k)) \oplus h(k)$ from a random permutation, where h is a hash function, is upper-bounded in the random oracle model by:

$$2^{-\kappa} \max \left(q_h, 2^{-n} \cdot D \sum_{k'_1, k'_2} \min(q_1(k'_1), q_2(k'_2)) \right) \tag{2}$$

for queries q_1 and q_2 to any two ciphers \mathcal{F}_{k_1} and \mathcal{G}_{k_2} such that $\mathcal{E}_k = \mathcal{G}_{k_1} \circ \mathcal{F}_{k_2}$.

This is very similar to an alternative to the FX construction proposed by Kilian and Rogaway, and already mentioned in §1.

From an efficiency point of view, using a call to a (small) hash function as part of the encryption process could be expensive. Therefore, the EH construction might be of little interest when computational power is limited or when the key has to be regularly changed (for instance because the cipher is itself used in a hashing mode). However, we believe that there are meaningful applications for block ciphers where none of these restrictions apply, making this type of instantiation still of interest. It is also worth noting that the input to h/f is of fixed size, and thus only a “one-shot” compression function with a fixed IV $\hat{h} : \{0, 1\}^\kappa \rightarrow \{0, 1\}^n$ is actually needed, and not a full-fledged hash function $h : \{0, 1\}^* \rightarrow \{0, 1\}^n$. Finally, it is likely that a smart instantiation would

use synergy between the core of the cipher and the h/\hat{h} function; using two completely unrelated functions for both components would probably not be the simplest way to proceed. In particular, it seems an interesting option to build the hash function by using the block cipher \mathcal{E} itself in a hashing mode⁵. If we can make it so that only one call to the compression function is needed (this is the case *e.g.* when the key and block sizes of \mathcal{E} are equal), then it is even possible to build the compression function from \mathcal{E} used as a *fixed permutation* \mathcal{E}' , with all its round keys independently set to a constant. This can be achieved by *e.g.* using \mathcal{E} in Matyas-Meyer-Oseas mode, with $\hat{h}(x)$ then defined as $\mathcal{E}'(x) \oplus x$. Note that such a construction can be performed with many current block ciphers, including AES-128. Even though care should be taken before using any cipher in a hashing mode (this is the case for AES too [36]), the fact that in this case it may be used with independent round keys may significantly improve its security in that setting (not least because it rules out meet-in-the-middle attacks such as [36]). Therefore, we believe that this instantiation strategy is sound, and that it can be applied to many existing ciphers, as well as being usable for future designs.

References

1. Aiello, W., Bellare, M., Di Crescenzo, G., Venkatesan, R.: Security Amplification by Composition: The Case of Doubly-Iterated, Ideal Ciphers. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 390–407. Springer, Heidelberg (1998)
2. Aoki, K., Sasaki, Y.: Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009)
3. Bellare, M., Krovetz, T., Rogaway, P.: Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-invertible. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 266–280. Springer, Heidelberg (1998)
4. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptology* 7(4), 229–246 (1994)
5. Biham, E., Dunkelman, O., Keller, N., Shamir, A.: New Data-Efficient Attacks on Reduced-Round IDEA. IACR Cryptology ePrint Archive 2011, 417 (2011)
6. Biham, E., Shamir, A.: Differential Cryptanalysis of the Full 16-Round DES. [10], 487–496
7. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)
8. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)

⁵ Obviously, one may also consider reduced-round variants of the same cipher in order to make this step faster. It is a designer’s role to find a good tradeoff between efficiency and security, and this instantiation strategy makes no exceptions.

9. Boullaguet, C., Derbez, P., Fouque, P.A.: Automatic Search of Attacks on Round-Reduced AES and Applications. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 169–187. Springer, Heidelberg (2011)
10. Brickell, E.F. (ed.): CRYPTO 1992. LNCS, vol. 740. Springer, Heidelberg (1993)
11. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-Resilient Functions and All-or-Nothing Transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
12. Choy, J., Zhang, A., Khoo, K., Henricksen, M., Poschmann, A.: AES Variants Secure against Related-Key Differential and Boomerang Attacks. In: Ardagna, C.A., Zhou, J. (eds.) WISTP 2011. LNCS, vol. 6633, pp. 191–207. Springer, Heidelberg (2011)
13. Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 222–238. Springer, Heidelberg (2001)
14. Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round AES. In: [32], pp. 116–126
15. Diffie, W., Hellman, M.: Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer* 10, 74–84 (1977)
16. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 719–740. Springer, Heidelberg (2012)
17. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 158–176. Springer, Heidelberg (2010)
18. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in Cryptography: The Even-Mansour Scheme Revisited. In: [34], pp. 336–354
19. Fouque, P.A., Karpman, P.: Security Amplification against Meet-in-the-Middle Attacks Using Whitening. *IACR Cryptology ePrint Archive* 2013, 618 (2013)
20. Gazi, P., Tessaro, S.: Efficient and Optimally Secure Key-Length Extension for Block Ciphers via Randomized Cascading. In: [34], pp. 63–80
21. Isobe, T.: A Single-Key Attack on the Full GOST Block Cipher. In: [22], pp. 290–305
22. Joux, A. (ed.): FSE 2011. LNCS, vol. 6733. Springer, Heidelberg (2011)
23. Khovratovich, D., Leurent, G., Rechberger, C.: Narrow-Bicliques: Cryptanalysis of Full IDEA. In: [34], pp. 392–410
24. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: Attacks on skein-512 and the SHA-2 family. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 244–263. Springer, Heidelberg (2012)
25. Kilian, J., Rogaway, P.: How to Protect DES against Exhaustive Key Search. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 252–267. Springer, Heidelberg (1996)
26. Leurent, G.: MD4 is Not One-Way. In: [32], pp. 412–428
27. Leurent, G.: Design and Analysis of Hash Functions. PhD thesis, Université Paris 7 (2010)
28. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. *J. Cryptology* 24(3), 588–613 (2011)
29. Matsui, M.: The First Experimental Cryptanalysis of the Data Encryption Standard. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 1–11. Springer, Heidelberg (1994)

30. May, L., Henricksen, M., Millan, W.L., Carter, G., Dawson, E.: Strengthening the Key Schedule of the AES. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 226–240. Springer, Heidelberg (2002)
31. Nikolić, I.: Tweaking AES. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 198–210. Springer, Heidelberg (2011)
32. Nyberg, K. (ed.): FSE 2008. LNCS, vol. 5086. Springer, Heidelberg (2008) (revised selected papers)
33. Nyberg, K., Knudsen, L.R.: Provable Security Against Differential Cryptanalysis. In: [10], pp. 566–574
34. Pointcheval, D., Johansson, T. (eds.): EUROCRYPT 2012. LNCS, vol. 7237. Springer, Heidelberg (2012)
35. Rivest, R.L.: All-or-Nothing Encryption and the Package Transform. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 210–218. Springer, Heidelberg (1997)
36. Sasaki, Y.: Meet-in-the-Middle Preimage Attacks on AES Hashing Modes and an Application to Whirlpool. In: [22], pp. 378–396
37. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
38. Vaudenay, S.: Provable Security for Block Ciphers by Decorrelation. In: Meinel, C., Morvan, M. (eds.) STACS 1998. LNCS, vol. 1373, pp. 249–275. Springer, Heidelberg (1998)

Secure Key Management in the Cloud

Ivan Damgård¹, Thomas P. Jakobsen^{1,*},
Jesper Buus Nielsen^{1,*}, and Jakob I. Pagter²

¹ Aarhus University**

{ivan,tpj,jbn}@cs.au.dk

² The Alexandra Institute A/S

jakob.i.pagter@alexandra.dk

Abstract. We consider applications involving a number of servers in the cloud that go through a sequence of online periods where the servers communicate, separated by offline periods where the servers are idle. During the offline periods, we assume that the servers need to securely store sensitive information such as cryptographic keys. Applications like this include many cases where secure multiparty computation is outsourced to the cloud, and in particular a number of online auctions and benchmark computations with confidential inputs. We consider *fully autonomous* servers that switch between online and offline periods without communicating with anyone from outside the cloud, and *semi-autonomous* servers that need a limited kind of assistance from outside the cloud when doing the transition. We study the levels of security one can – and cannot – obtain in this model, propose light-weight protocols achieving maximal security, and report on their practical performance.

Keywords: Information security, cloud computing, cloud cryptography, secure key management, secure distributed storage, secure multiparty computation.

1 Introduction

Cloud computing is a disruptive technology, changing the way computing resources are deployed and consumed. The benefits of cloud computing are many, ranging from cost-efficiency to business agility. The main drawback, however, is security and in particular data confidentiality: Users of cloud technology essentially have to trust that the cloud providers do not misuse their data. The recent disclosure of the PRISM surveillance program¹ in which NSA directly monitors

* Supported by the Danish Council for Independent Research via DFF Starting Grant 10-081612.

** The authors acknowledge support from the CFEM research center (supported by the Danish Strategic Research Council), the Danish National Research Foundation, and the National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed.

¹ [http://en.wikipedia.org/wiki/PRISM_\(surveillance_program\)](http://en.wikipedia.org/wiki/PRISM_(surveillance_program))

all communication going through most world-wide cloud providers such as Yahoo, Google, and Microsoft, is just one out of several incidents emphasizing that this concern about security is real.

In the simple cloud computing case where a user outside the cloud wants to store some data in the cloud for later retrieval, data confidentiality and integrity can relatively easily be ensured. This is typically done using standard cryptography, by encrypting the user's data before it is stored in the cloud, keeping the encryption key secret from the cloud provider. Several products such as CrashPlan² and CloudFogger³ already offer this kind of security.

But the cloud is more than just a storage medium: In particular, computation itself is often outsourced to the cloud. In some cases the computation outsourced is even distributed among several cloud servers and may involve data from many clients. Sometimes the cloud servers may even be controlled by different organizations. Also, the cloud servers may exist in different parts of the cloud, spread across different cloud providers such as Microsoft, Amazon, etc.

An example of this is the Danish site `energiauktion.dk` where electricity for companies is traded at daily online auction. This works by each day starting up a number of auction computations in the cloud. In order to protect the confidentiality of the submitted bids, even against collusions involving the operator of the auction site itself, the bids are encrypted at the clients (the companies), and the auction computations are done using MPC where each MPC server is running in the cloud, controlled by its own organization. Another relevant example is that of the Danish sugarbeet auctions [4]. Here, similar auctions take place, but running on a yearly basis and computing the optimal way to trade Danish sugarbeet contracts instead of electricity. As for `energiauktion.dk`, the confidentiality of bids for the sugarbeet auctions are also ensured via MPC.⁴

Strong notions of security in such more involved cloud applications are generally not as easily obtained as in the simpler case of cloud storage. Promising technologies such as fully homomorphic encryption (FHE) [22] and secure multiparty computation (MPC) [42,23] definitely have the potential to raise the security for these applications. But despite recent advances [32,12,24] they are still quite demanding in terms of performance. While the functions to compute securely in the above examples are simple enough to allow for MPC, securing applications via MPC or FHE in general would still be too resource demanding. More light-weight solutions are therefore needed.

This paper is a study of certain subsets of these cloud computing scenarios. In Section 2 through Section 5 we define these subsets, analyze which levels of security can be obtained and provide concrete protocols achieving this security. In Section 6 we report on a prototype implementation of the proposed protocols. For lack of space, the presentation in these sections has been kept at an informal

² www.crashplan.com

³ www.cloudfogger.com

⁴ The sugarbeet auctions are actually computed by laptop computers on a standard LAN network, but would fit nicely to a cloud setup.

level. A formal model of the protocols and proof of their security within the UC framework [6] are provided in the full version of this paper [11].

2 The Model

In this paper we are interested in the following scenario:

- **Distributed computation.** A number of n servers S_1, S_2, \dots, S_n are engaged in some distributed computation taking place in the cloud.
- **Online/offline periods.** The computation does not proceed in a continuous fashion. Rather, in some periods there is no need for computation and the servers are therefore idle. We call the first periods for online periods and the latter for offline periods. In this way the application goes through a number of rounds, each round consisting of an online period followed by an offline period, and we assume that the servers receive signals from the application that allows them to agree on the times to switch between online and offline.
- **Sensitive state.** During the application’s lifetime some or all of the servers possess sensitive data that is needed by these servers in the online periods and that must be stored securely during the offline periods. This could for example be data used in the computation itself or keys needed to authenticate against other servers. We will refer to the sensitive data that S_i must store securely during the offline phase of round r as that server’s secret file and we denote it by σ_i^r .

This model of course does not cover all kinds of cloud computing. Regarding the online/offline property we note, however, that many applications naturally only require computation at certain well-defined points in time. For example, online auctions and benchmarks are often designed to be repeated at regular time intervals. Furthermore, most cloud providers operate on a pay-per-use basis (pay per CPU cycle spent, pay per byte sent, etc.) that in itself motivates the design of applications in which computation is “batched” together in time as much as possible such that the cloud servers can be shut down in between these periods of computation in order to save money.

Examples fitting particularly well into our model include those where MPC is done in the cloud. Using MPC is for instance relevant in order to let a client securely outsource a computation to the cloud: By computing via MPC and by making sure that servers are hosted at different cloud providers, strong security is guaranteed since a large number of different cloud providers must collaborate maliciously in order to violate security. Other examples involving MPC in the cloud are the electricity auctions and the Danish sugarbeet auctions mentioned above. Both of these applications involve the regular running of auctions with bids containing confidential client information, and to guarantee confidentiality of the bids, the auction computations are done using MPC where the servers are controlled by different organizations. These applications therefore consist of a distributed system of servers going through a number of online and offline periods (daily offline periods for the electricity auctions, yearly offline periods in

the sugarbeet case). During the offline periods, the servers need to store secret data, namely the keys used for doing MPC.

In this paper we do not aim at providing any extra security in the online periods, other than what can already be obtained by other means such as MPC. Rather, we are concerned with the question: *To which extend is it possible to guarantee the confidentiality, integrity, and availability of the servers' secret files in the offline periods, given various attacks on the servers and the network over which they communicate.* By confidentiality and integrity we mean that a secret file stored by a server at shutdown during an offline period is guaranteed not to be read by others and that the server can be assured that it reads the same unmodified file at wakeup as it stored at the previous shutdown. By availability we refer to the guarantee that a file stored by a server can later be retrieved again.

In the first part of our paper we take into account the following additional requirement:

- **Autonomous servers.** The transition between online and offline periods must proceed without anyone from outside the cloud taking action. That is, the servers must be able to switch between online and offline periods communicating only with each other.

This may be essential to some applications. In particular, there simply may not be any relevant parties outside the cloud, such as system administrators or non-cloud servers within an organization, with the right levels of trust, at the times when the cloud servers shut down or wake up.

It turns out that within this model, where the only players are the servers themselves, there is a limit to the level of availability and confidentiality we can hope to get at the same time: Any protocol that guarantees that some subset of the servers can restore a secret file at wake up (availability) of course also allows the same subset of servers to learn the contents of this file, meaning that the file leaks if the servers in that subset are malicious (lack of privacy). Let t_{conf} be the confidentiality threshold, that is, the maximal number of servers an adversary can corrupt without learning anything about the contents of the secret file. Similarly, let t_{avail} be the availability threshold, meaning the maximal number of servers the adversary can corrupt without being able to prevent the reconstruction of the secret file. We can then express this trade-off as follows.

Fact 1. *(Informal) With autonomous servers, the thresholds t_{conf} and t_{avail} must satisfy the equation $t_{\text{conf}} + t_{\text{avail}} < n$. In particular, no protocol for fully autonomous servers can guarantee at the same time both confidentiality and availability of a secret file in the presence of more than $n/2$ malicious servers.*

In Section 4 we present a protocol for fully autonomous servers that achieves very strong privacy. Because of Fact 1, this means that we have to give up on availability.

The limitation expressed by Fact 1 is a consequence of the requirement that the servers are only allowed to communicate with each other during shutdown

and wakeup. We therefore continue our study in the second part of the paper, Section 5, by considering how to most meaningfully relax the requirement of autonomous servers in order to gain more security, while at the same time minimizing the involvement from outside. We end up with the following slightly relaxed requirement:

- **Semi-autonomous servers.** Under normal conditions the transition between online and offline periods must proceed without anyone from outside the cloud taking action. However, in case the system has been attacked, we allow the transitions to involve actions from someone from outside the cloud.

We model this more concretely by assuming the existence of certain parties outside the cloud that can fetch data from the cloud servers. For convenience we call these parties **administrators**, though it could also for example be automated scripts running on trusted (non-cloud) servers within the organizations operating the cloud servers.

The model with semi-autonomous servers covers many applications where the cloud servers are operated by organizations that have their own trusted people or servers elsewhere, outside of the cloud, that can assist the cloud servers in the transitions between online and offline periods. In particular, it models well the scenario where human system administrators are actually willing to log in to the cloud servers in certain situations.

In Section 5 we present a protocol with semi-autonomous servers, providing both very strong confidentiality and availability while at the same time relying only on minimal involvement from the administrators. The protocol essentially guarantees that an administrator can always restore a secret file stored on his server unless *all* the cloud servers have been corrupted.

While definitely suited for increasing security of applications like the Danish sugarbeet auctions [4] and **energiauktion.dk**, we believe that these two models capture many other interesting classes of cloud computing applications and that the protocols provided here therefore will be useful for enhancing security for such applications.

3 Related Work

Our work is based on secret sharing. Several secret sharing schemes exists, including schemes allowing various thresholds [3,38] and schemes with proactive security [25]. However, for the same reason as for Fact 1, secret sharing considered in isolation can never give both availability and confidentiality in case of a dishonest majority of parties. We show how to combine secret sharing with other techniques in a specific context and thereby achieve a level of security that one cannot get with secret sharing alone. In particular, we show how to get both confidentiality and availability in the presence of a dishonest majority in the model with semi-autonomous servers.

Secure multiparty computation (MPC) [42,23] allows a set of servers to jointly compute on encrypted data. Security, including data confidentiality, is then guaranteed even though some of the servers are malicious and may pool their data

together. While still a very resource demanding technology, MPC has reached a level where it has become practical, at least for a limited class of applications [4,17,33,39]. By letting the cloud servers compute using one of the MPC protocols designed to give security against dishonest majority [12,32] one can achieve a level of security somehow similar to ours in the offline periods, namely that an adversary must break into the offline storage of *all* servers, stealing or modifying *all* MPC keys, to do any harm. However, as mentioned, MPC is still often too heavy and contrary to our protocol for semi-autonomous servers, protocols designed for dishonest majority MPC do not provide strong termination (meaning availability of files in our case).

In any case, for all non-MPC computations as well as computations based on more light-weight MPC protocols that assume less powerful adversaries (that is, honest majority protocols, honest-but-curious or covert adversaries, etc.) our protocols can be used to enhance security during the offline periods at a low cost. In this way, our protocols can be seen not as a substitute, but more as a *complement* to the use of MPC.

Fully homomorphic encryption [22,21] allows general computations on encrypted data and is in many respects considered the “holy grail” of cloud computing security. FHE allows a user to outsource computation to one or more cloud servers without violating confidentiality even if *all* servers are malicious. Combining FHE with other techniques allows to also guarantee the correctness of the outsourced computation [20,10,9]. Furthermore, recent results consider outsourcing computations involving input from several parties [27]. As such FHE can be used to secure essentially any cloud computing scenario, including those we consider, to a very high degree. Despite recent improvements [5,24], though, the performance of FHE is still a long way from being efficient enough for practical purposes. Therefore our protocols, perhaps combined with MPC in the online periods, pose a more realistic way to secure cloud computing, at least for the foreseeable future.

Our work essentially consist of protocols for secure, distributed storage of keys and is as such related to the broader field of secure storage and secure distributed file systems. Lots of work has been done in these areas [28], but in many cases such as NFS [40], AFS [26], and SFS [18], security only means that unauthorized clients cannot access or modify files; the storage servers themselves are trusted. Due to increased security demands, a new generation of so-called *cryptographic* storage systems has evolved, exemplified by Windows EFS [16], NCryptfs [41], and many others. Using various kinds of cryptography, these systems provide the stronger notion of *end-to-end* security, meaning that clients no longer need to trust the storage servers. However, all of these systems require that clients themselves securely store keys and/or remember passwords and are therefore not suitable in our setting where servers must operate autonomously (or at least semi-autonomously) in the cloud.

Some results [19,2,30] already consider how data can securely be dispersed among a number of servers *without* the need for storing secret keys on any client. These results combine secret sharing, error correcting codes, variants of Rabin’s

information dispersal algorithm [34,29] and other cryptographic techniques in order to guarantee both confidentiality, integrity and availability of the stored data. Forward-secure threshold encryption [31] could also be used to encrypt files at shutdown. As such, these protocols could indeed be used to secure data during the offline phases in our model with autonomous servers.

Common to these results, however, are that they only provide security in the presence of up to $n/2$ malicious servers. In contrast, the constructions provided in this paper are designed to guarantee confidentiality and integrity of the stored data in the presence of up to $n - 1$ servers. In Section 4 we achieve this for fully autonomous servers by giving up on availability. In Section 5 we show how to take advantage of the model with semi-autonomous servers in order to also guarantee availability with up to $n - 1$ malicious servers.

We are, to the best of our knowledge, the first to consider protocols specifically designed for securing the offline periods in cloud computing environments as described above. In particular, we are not aware of any existing protocols suitable for such cases providing the same combination of high security and good performance as those we present in this paper.

4 Fully Autonomous Servers

We here describe a protocol that increases the offline security in the model with fully autonomous servers and we discuss the limits of the possible security we can in this model. Due to space restrictions the description is kept at an informal level while a formal modelling of the protocol in the UC framework [6] and a rigorous proof of its claimed security are postponed to the full version of this paper [11].

Suppose that the overlying application has a security threshold of T_{app} , meaning that an adversary breaking into T_{app} or less servers does not violate security of the overlying application. For many applications $T_{\text{app}} = 0$, but T_{app} may also be higher, say $T_{\text{app}} = n/2$, if for example the overlying application is MPC.

4.1 What Cannot Be Done

As discussed earlier, there is a limit as to how much confidentiality and availability we can achieve at the same time with fully autonomous servers. In addition we observe that it is clearly impossible to protect against the following kind of attack: If the adversary manages to passively break into server S_i during an offline period, he learns whatever that server knows. If he then also attacks S_i 's network channels in the following wakeup phase, he can cut off S_i , that is, silence S_i and pretend to be that server towards the remaining servers, using the keys for authentication stolen from S_i during the offline period. By doing this, the adversary has essentially carried out what corresponds to an active (Byzantine) corruption of S_i only by means of a combination of a passive break-in during the offline period followed by a network attack – two attacks normally considered less difficult than a full active attack on the server. We will denote such attacks

as cut-off attacks, and not being able to avoid these can be seen as the price we pay for not involving any external parties in the protocol.

4.2 What Can Be Done

We start out with the simplest possible solution and gradually show, in a number of steps informally discussed below, how to extend the solution in order to increase security. The resulting protocol is presented in its entirety at the end of this section.

Secret sharing. In the most naïve protocol each server simply stores its own secret file locally during offline periods. This of course does not add any extra security. In particular, an adversary can spoil security by breaking into the offline storage of $T_{\text{app}} + 1$ servers. The servers could encrypt their secret files before storage, but not much is gained if the encryption key is also stored locally.

We therefore let each server S_i encrypt its file σ_i using a randomly chosen encryption key L_i and then secret share this key among the full set of servers, each server keeping one share $s_{i,i}$ for itself and sending the remaining shares $s_{i,j}$ to each of the other servers S_j before the offline periods. When the servers wake up for the next online period, each server collects its missing shares from the other servers, reconstructs the encryption key, and decrypts its secret file.

With this approach the trade-off between confidentiality and availability discussed earlier can be adjusted by using secret sharing schemes with different thresholds. For example, using Shamir's secret sharing scheme [38] with threshold $t = n/2$ ensures availability of secret files unless $t + 1$ servers are malicious, but also only guarantees confidentiality of the files for up to t malicious servers. For now we aim at optimal confidentiality and therefore instead use a sharing scheme with full threshold ($t = n - 1$), such as additive sharings over a finite field. (Better availability is considered later, in Section 5).

This first solution ensures optimal confidentiality of the secret files against adversaries performing only offline attacks, but we also have to consider network attacks. Consider first the case where we just send the shares in cleartext. Here we can observe that S_i never sends its own share $s_{i,i}$ to anyone. Therefore any attack that only uses the network will miss at least one share for every server and so cannot get any useful information. On the other hand, it is also clear that passive eavesdropping combined with an offline attack on S_i will allow you to get σ_i , and so passive eavesdropping plus offline attacks on $T_{\text{app}} + 1$ servers will break the system.

Diffie-Hellmann key exchange. To improve this, we can encrypt the communication. However, securing the communication channels using standard encryption requires servers to store private keys and therefore does not add extra security: This solution can be still broken by offline attacks on $t + 1$ servers and passive eavesdropping, because the adversary then knows the keys he needs for decryption.

Instead, we use Diffie-Hellman (DH) key exchange [13] to set up secure pairwise channels on the fly when the systems starts up. DH lets each pair of servers S_i and S_j agree on a secret session key $K_{i,j}$ that can be used to encrypt the channel. This way no private keys for encrypting the channels need to be stored during the offline periods. This means that we are secure against offline attacks on up to $n - 1$ servers combined with passive eavesdropping. This is an improvement for any application with $T_{\text{app}} < n - 1$.

The above solution does not authenticate the messages in the DH key exchange, since it is only designed to cope with passive eavesdropping. Using an active network attack, an adversary could therefore impersonate any agent during wakeup, and could therefore get the same information as one would get if everything was sent in the clear. However, such an attack alone will not give him any useful information, for the same reason that we described above (S_i never sends its own share $s_{i,i}$ to anyone). As before, if this is combined with an offline attack on T_{app} servers, one gets the online information for these and nothing more. We are therefore still secure against offline attacks on up to T_{app} servers combined with active network attacks. This is optimal because – as discussed in connection with cut-off attacks above – the same attack on $T_{\text{app}} + 1$ servers is equivalent to $T_{\text{app}} + 1$ full corruptions which is always fatal. In particular, this shows that we do not get any benefit from authenticating the messages in the DH key exchanges.

In conclusion, the solution sketched so far has optimal security against both offline plus passive network attacks as well as offline plus active network attacks, namely security against attacks on $n - 1$, respectively T_{app} servers.

Detecting attacks. It turns out that authenticating the DH key exchanges, for instance using digital signatures as in the STS protocol [14], or more generally, using any scheme for authenticated key exchange (AKE), is not useless, however. As discussed earlier, cut-off attacks cannot be prevented. But using AKE, in case a cut-off attack do in fact occur, the cut-off server S itself will always notice that something is wrong, as long as it is not actively corrupted when it wakes up. The reason is the following: Since the adversary broke passively into S during offline it knows S 's private AKE key and can thus pretend to be S towards the remaining servers in the online phase. But the real, but impersonated, S will still try to do AKEs with the remaining servers. Unless the adversary passively breaks into *all* the servers, there will be at least one private AKE key that he does not know. This means that S will experience that at least one of the AKEs he tries to complete will fail and can therefore abort the protocol and try to warn the other servers. In other words, the adversary can cut off S , but cannot prevent S from *detecting* the cut-off attack, and unless the adversary can carry out a denial-of-service attack on S forever (something that is often considered practically impossible), this fact will become known to the rest of the system. For these reasons we will use AKE instead of unauthenticated DH in our solution.

Integrity. In the above discussion we have focused on confidentiality. The solution does not, however, protect against for example a corrupted server modifying a

share before sending it back to another server at wakeup. We can protect against this by replacing the basic secret sharing scheme with an extended scheme, that we will denote as a **robust secret sharing scheme** (RSS). Such a scheme produces along with the shares, s_1, s_2, \dots, s_n a public verification key V . The key V reveals no information about the secret, and can be kept by the server during the offline period and used at wakeup to verify that the shares reconstruct to the original secret. Details on this kind of secret sharing is provided in the full version of this paper [11].

Proactive security. Proactive security is a powerful notion of security put forward by Canetti *et al.* [7]. In short, a protocol is proactively secure if it can tolerate *any* number of corruptions during its lifetime as long as only a certain number of corruptions take place within a given time frame. Having proactive security is important for protocols such as ours that are supposed to run for a long time.

Our current protocol already is proactively secure in a limited sense: Due to the fact that fresh session keys are generated in each round, we can tolerate any number of *passively* corrupted servers in the offline phases, as long as at most $n - 1$ of the corruptions happen in the same offline phase.

There is no proactiveness for the detection of cut-off attacks discussed above, though, since the servers use the same keys for authenticating the DH throughout all rounds. This means that if one manages to steal the private signing key belonging to a server in one round, this key can be used to cut off that server in a later round. We can remedy this by letting the servers in each round refresh the digital signature keys for authenticating the DH. The refreshment is done by letting each server generate a new key pair, replacing its old private signing key with the fresh signing key while sending the new public verification key to the other servers where similar replacements take place. To prevent an attacker from modifying these new public verification keys while they are in transit, each server attaches a message authentication code (mac) to the key, using the current session key that the sending and receiving server share.⁵

With these extra steps we have now obtained a protocol that is proactively secure with respect to passive corruptions and detection cut-off attacks, with each round being one refreshment period. However, obtaining proactive security against *active* offline attacks, that is, where someone not only gets read access to the servers' offline storage, but who can also modify this state during the offline period, turns out to be impossible, at least without any further assumptions. This stems from the fact that once the server gets actively corrupted, during offline as well as online periods, the adversary can change all state, including

⁵ Our way of securing the network resembles to some extent the way in which SSL/TLS works: SSL/TLS can be configured to use symmetric encryption and macs with a session key established using authenticated Diffie-Hellmann, and also provides a mechanism for renegotiating the keys used to authenticate the DH on a regular basis. We choose however, to embed encryption, etc., directly in our protocol, not relying on SSL/TLS. We do this because we want to be able to reason formally about the security of our protocol which would not be easy with SSL/TLS that consists of over 100 combinations of encryption modes, handshakes, etc.

the protocol code that specifies how the server behaves. By modifying the code offline, the adversary can in effect control the behaviour of the server for the following online period. In this sense, an active attack on a server during the offline period is equivalent to a full active attack on that server during the following online period.

Making the additional assumption that the *code* of each server cannot be changed during offline periods, we can do better. This assumption is a variant of the Read-Only Memory (ROM) model discussed further by Canetti *et al.* [8].⁶ In the ROM model, we can strengthen our protocol by letting each server compute a hash of its secret file plus some random salt at shutdown and distribute this hash to all servers (including keeping a copy itself). At wakeup we let the server collect again the hashes and abort if these are not all equal. In the ROM model this implies that an adversary will have to actively corrupt the offline storage of *all* servers in the same offline in order to break integrity.

The Protocol. We denote the protocol resulting from this discussion the *Cloud Key Management* protocol, or just P_{CKM} . It is illustrated in Fig. 1 and consists of two procedures to be carried out by each server, one before entering an offline period (shutdown) and another before returning to the next online period (wakeup). The entire protocol consists of several rounds, each round r consisting of four phases: An online phase where the application is running, a shutdown phase where the servers run the P_{CKM} shutdown procedure, an offline phase with no computation, and finally a wakeup phase where the servers run the P_{CKM} wakeup procedure to restore the secret files.

When a server S_i receives a file from the environment at shutdown, it is encrypted under a key L using symmetric encryption (Enc). That key is then split into shares $\{s_{i,j}\}_{j \in [n]}$ using a robust secret sharing scheme (RSS). The server keeps one of the shares, $s_{i,i}$ and distributes the remaining shares among the other servers, using a session key for encryption and message authentication codes (macs) to protect against leakage and modification during network transmission. At the end of the shutdown procedure the server erases most values, including the file itself, from its memory. The only values remaining in the following offline phase are the encrypted file, the keys needed for AKEs in the the following wakeup phase, the server's own share and the shares received from the other servers (that follow the same shutdown procedure). On wakeup, a procedure reverse to the shutdown procedure takes place: The server receives its shares from the other servers, reconstructs the key, verifies its integrity, decrypts the file and returns it to the environment. At the beginning of each wakeup and shutdown phase, a server S_i agrees on a fresh secret session key with each of the other servers using AKE. The private and public keys used for AKEs are refreshed once in each round at shutdown.

⁶ The assumption can sometimes be justified by the use of ROM or other special hardware such as TPM modules. Also, one can perhaps argue that this models well a cloud environment with all servers booting up from the same uncorrupted virtual image on every wakeup.

Shutdown Each server S_i holds from the previous round a private key sk_i^{r-1} and public keys vk_j^{r-1} for each of the other servers S_j . When receiving the secret file σ from the application, S_i does the following.

1. *Session key refreshment*
 - (a) For each of the other servers S_j , invoke (in parallel) the AKE protocol, using sk_i^{r-1} and vk_j^{r-1} . This results in S_i and S_j sharing a fresh secret session key $K_{i,j}^{\text{down}}$.
 - (b) Generate a new AKE key pair (sk_i^r, vk_i^r) .
2. *Encrypt file and distribute shares of the encryption key.*
 - (a) Choose a random encryption key L and compute $C \leftarrow \text{Enc}_L(\sigma)$.
 - (b) Compute $V, \{s_{i,j}\}_{j \in [n]} \leftarrow \text{RSS}(L)$.
 - (c) For each of the other servers S_j , the server S_i compute $c_{i,j} \leftarrow \text{Enc}_{K_{i,j}^{\text{down}}}(s_{i,j})$ and $d_{i,j} \leftarrow \text{Mac}_{K_{i,j}^{\text{down}}}(vk_i^r)$.
 - (d) Sends the message $M_{i,j} = c_{i,j} \parallel vk_i^r \parallel d_{i,j}$ to S_j (keeping $s_{i,i}$).
 - (e) Wait to receive messages $M_{j,i} = c_{j,i} \parallel vk_j^r \parallel d_{j,i}$ from the other servers S_j . Abort if the mac $d_{j,i}$ is invalid, otherwise compute $s_{j,i} \leftarrow \text{Dec}_{K_{i,j}^{\text{down}}}(c_{j,i})$. This step is repeated until valid shares and public keys have been received from all other servers.
3. *Offline state hashing.* Let \mathcal{O} be the concatenation of $(sk_i^r, C, V, s_{i,i})$ with the shares $s_{j,i}$ and public AKE keys vk_j^r received from the other servers S_j . Compute $\gamma_i \leftarrow \text{H}(\mathcal{O})$ and send γ_i to all other servers along with a mac using $K_{i,j}^{\text{down}}$. Wait until valid hash values γ_j have been received from all other servers.
4. Erase all data except \mathcal{O} and the hashes $\{\gamma_j\}_{j \in [n]}$.

Wakeup On wakeup S_i does the following.

1. *Session key refreshment.* Invoke the AKE protocol, this time using sk_i^r and vk_j^r , resulting in S_i and S_j sharing a fresh secret session key $K_{i,j}^{\text{up}}$.
2. *Offline state verification.* Send γ_j to S_j , along with a mac of it using $K_{i,j}^{\text{up}}$. Wait to receive γ_j from the other servers. Verify that all macs are valid and that $\text{H}(\mathcal{O}) = \gamma_j$ for all $j = 1, 2, \dots, n$, and abort otherwise.
3. *Reestablishing the secret file.*
 - (a) Compute $c_{j,i} \leftarrow \text{Enc}_{K_{i,j}^{\text{up}}}(s_{j,i})$ and send $c_{j,i}$ to S_j . Wait until $c_{i,j}$ is received from all other S_j and compute $s_{i,j} \leftarrow \text{Dec}_{K_{i,j}^{\text{up}}}(c_{i,j})$.
 - (b) Reconstruct L from $\{s_{i,j}\}_{j \in [n]}$ and verify integrity of the sharing using V . Abort if invalid, otherwise compute and return to the application $\sigma \leftarrow \text{Dec}_L(C)$.
4. Erase all values except sk_i^r and vk_j^r for the other servers S_j .

Fig. 1. The P_{CKM} (Cloud Key Management) protocol

A few notes about the protocol are in place: The refreshment of the AKE keys is done once every round, but the session key is refreshed twice each round, using the same AKE keys. The reason for doing two session key refreshments is to avoid any shared session key to reside in memory not only during offline, but also during online periods, as doing so would reduce the number of corrupted servers we can tolerate. Also, for simplicity of presentation, the same session key is used for both encryption and macs in Fig. 1. A secure implementation

would require separate keys for macs and encryption, see the full version [11] for details.

Security. In order to summarize the security of P_{CKM} we first define cut-off attacks more precisely as follows:

Definition 1 (Cut-Off Attack). *A cut-off attack on S_i in round r is a passive corruption of S_i in round $r - 1$ or r (stealing the server's private AKE key sk_i^{r-1}) combined with active network attack on all of S_i 's channels during the shutdown phase of round r (impersonating S_i in the AKEs done there), or a passive corruption of S_i at some point during round r (stealing sk_i^r) combined with active attacks on all of S_i 's channels during the wakeup phase of round r (impersonating S_i in the AKEs done in that phase).*

We also note that the security of P_{CKM} builds on a number of assumptions:

- *Trusted setup* A once-and-for-all setup must be in place, consisting of the initial AKE keys for the first round. This can for instance be established in practice by a PKI.
- *Cryptographic assumptions* Various cryptographic assumptions due to the primitives used in the protocol. For example, the STS protocol for authenticated key exchange [14] builds on the DDH assumption. More details on this can be found in the full version [11].
- *Erasure* That a server can erase part of its state on shutdown such that it is not accessible to an adversary that gets access to the server's offline storage.
- *Randomness* That each server has access to a source of close-to-true randomness.
- *Static adversary* We assume that the adversary decides before each round which servers and channels to corrupt in the following round.
- *Code in Read-Only Memory (ROM)* We assume that at least the code of the protocol itself is stored in ROM and cannot be altered by an active offline attack.

In [11] we show how to model these assumptions precisely in the UC framework. The ROM assumption is perhaps the most questionable of these assumptions, so we first summarize what security we have obtained without the ROM assumption.

Theorem 1. *(Informal) Given the assumptions above (except the ROM assumption), the confidentiality and integrity of a file σ_i^r stored and retrieved by S_i in round r using the P_{CKM} protocol in Fig. 1 is guaranteed as long as*

1. S_i has not been actively corrupted (during offline or during online periods) up to and including round r .
2. S_i is neither passively corrupted in the shutdown or in the wakeup phases of round r .

3. No cut-off attack on S_i takes place up to and including round r .
4. No more than $n - 1$ servers are passively corrupted in each offline phase up to and including round r .

Furthermore, if σ_i^r leaked due to S_i being exposed to a cut-off attack at any point up to or including round r , this will be detected by S_i .

In particular, without the ROM assumption, a server being actively corrupted at any point, including during offline periods, will stay actively corrupted throughout the protocol. On the other hand, as stated in Theorem 2, the ROM assumption allows us to achieve full proactive security with regards active corruptions.

Theorem 2. (Informal) *Given the assumptions above (including the ROM assumption), the confidentiality and integrity of a file σ_i^r stored and retrieved by S_i in round r using the P_{CKM} protocol in Fig. 1 is guaranteed as long as*

1. S_i is neither passively or actively corrupted during the shutdown or wakeup phases of round r .
2. No cut-off attack on S_i takes place in round r .
3. A maximum of $n - 1$ servers are corrupted, actively or passively, in each round up to and including round r .

Furthermore, if σ_i^r leaked due to S_i being exposed to a cut-off attack in round r , this will be detected by S_i .

This section has been kept at an informal level, including the two theorems above. For lack of space, more precise definitions of the primitives used, such as the AKE scheme and the robust secret sharing, a formal model of the protocol itself, and a proof of its security in the UC framework have been deferred to the full version of this paper [11].

5 Semi-autonomous Servers

In the previous section, dealing with fully autonomous servers, we had to choose between guaranteeing either confidentiality or availability in case of dishonest majority as expressed by Fact 1, and we aimed at a protocol with maximal confidentiality. Here we show how to construct a protocol with the same strong confidentiality as before, but with improved availability. This is possible because semi-autonomous servers are allowed to interact with someone from outside the cloud in case of an attack.

As discussed earlier, this is done by providing a special recovery mechanism by which an administrator for a server is guaranteed to be able to recover a file, even if the normal wakeup procedure fails to terminate.

The protocol, which we will denote as P_{CKM}^* , is given below in Fig. 2 and is an extension to the P_{CKM} protocol described earlier, that in addition relies on a threshold signature scheme. Such a scheme allows the servers to collectively

sign data without any single server being able to sign. In fact, to fit in P_{CKM}^* the threshold scheme must have full threshold and be proactive. It turns out that the threshold signature scheme of Almansa *et al.* [1] is easily modified, giving up on termination, to satisfy our needs. Details are provided in the full version [11].

The protocol works as follows: As part of the trusted setup, we also require a threshold signature scheme to have been initialized with the signing key distributed among the servers and such that the administrator and all servers hold the public verification key. In addition we require each administrator to hold a private decryption key for which his server holds the corresponding public encryption key. At shutdown, along with the procedure already specified by P_{CKM} , the server S_i computes an encryption F_i under the administrator's public encryption key, and the servers then collectively sign F_i . The signature is distributed to all servers, using the session key to authenticate the channels. If normal operation fails during wakeup, the administrator requests the copies of the encrypted file held by the servers. When obtaining one or more of these, he verifies integrity and decrypts the secret file using his private decryption key.

Some additional comments on the protocol: The receipts ensure that if S_i goes offline without aborting, all honest servers have marked the encrypted file of S_i as accessible to the administrator. The operation of "making data accessible" typically involves that this information is stored in a dedicated location on the server's disk, but one could also imagine that on shutdown, this public information is collected, say on a trusted mail server. If normal file recovery by the servers fails, the administrator can, with his verification key, log in to this email server and access the information needed to restore the file.

The security of P_{CKM}^* is summarized in the following theorem.

Theorem 3 (File Availability). *(Informal) The protocol P_{CKM}^* has the same guarantees as P_{CKM} regarding confidentiality and integrity of stored files. Furthermore, once a server that has not been actively corrupted up to and including round r , finishes the shutdown procedure, the file σ_i stored at that server is guaranteed to be recoverable by the corresponding administrator, unless all servers are actively corrupted during the following offline and wakeup phase.*

Again, for lack of space, a more precise modelling of the protocol in the UC framework, including the modelling of the administrator, is deferred to the full version of this paper [11]. The intuitive reason for the strong availability is that because of the threshold signature scheme, the adversary must corrupt *all* servers during the offline period in order to forge the signature or delete all copies of the encrypted file, F_i . If not, the administrator will be able to restore the correct file by fetching F_i from just one honest server, verify the threshold signature, and decrypt it using his private encryption key.

This is a considerably stronger availability guarantee than what was achieved by the P_{CKM} protocol. We stress that the extended protocol P_{CKM}^* works in the semi-autonomous model and therefore requires the involvement of administrators, but only if retrieving secret files in the normal, autonomous, way fails due to an attack on the system.

The trusted setup works as in P_{CKM} , but also includes that the administrator gets a private decryption key dk while each server gets a copy of the corresponding public encryption key ek . Also, the threshold signature scheme is setup, meaning that the administrator and all servers gets the public verification key \mathcal{W} while the shares $\{w_j\}_{j \in [n]}$ of the corresponding signing key is distributed among the servers.

Shutdown As P_{CKM} , but with the addition that also the proactive refreshment method of the threshold signature scheme is invoked. Also, the following additional steps performed by server S_i the erasing of values in Step 4 of P_{CKM} :

1. Compute $F_i \leftarrow \text{Enc}_{ek}(\sigma)$.
2. Compute a threshold signature f_i of F_i by invoking F_{THSIG} .
3. Verify f_i using the public verification key \mathcal{W} and abort if invalid.
4. Place F_i and f_i somewhere that is accessible by the administrator.
5. Send (F_i, f_i) to all other servers.
6. When a pair (F_j, f_j) is received from another server S_j , verify the signature f_j and abort if invalid. Otherwise, make the pair accessible to the administrator and return an OK message to S_j with channel integrity ensured by the session key $K_{i,j}^{\text{down}}$.
7. Abort unless valid signatures have been received from all other servers, and valid OK messages from all servers have been received for the (F_i, f_i) that was sent out from this server.

Wakeup As P_{CKM} .

File Recovery When the administrator wants to recover the file σ during the wakeup phase he does the following:

1. Fetch messages (F_i, f_i) from the servers (can be done in parallel).
2. When a message (F_i, f_i) is fetched from S_j , verify the signature f_i . If valid, output $\sigma \leftarrow \text{Dec}_{dk}(F_i)$. If invalid, fetch a message from another server.

Fig. 2. The protocol P_{CKM}^* for semi-autonomous servers

6 A Prototype Implementation

A prototype of the basic P_{CKM} protocol (without the mechanism for recovery of files by administrators) has been implemented and benchmarked in the Amazon Web Services (AWS) cloud environment. We here report on these benchmarks and discuss a few practical aspects related to the implementation.

For the benchmarks, each server was running on its own EC2 instance with an Elastic Block Store (EBS) volume as permanent storage. Before each offline period, the shutdown procedure of P_{CKM} was executed following by disposing each EC2 instance such that during the offline phase only the EBS storage volumes remained. On wakeup, new EC2 instances were started up, the EBS volumes re-associated to the EC2 instances, and the wakeup procedure of P_{CKM} subsequently executed in order to restore the secret files of the servers.

Table 1. Performance of the CKM protocol, P_{CKM} , in seconds (with 95% confidence intervals). Timings do not include EC2 disposal and start-up times. Each server runs on a small EC2 instance corresponding roughly to 1.7 GiB RAM and a 1.0-1.2 GHz 2007 Xeon processor [15]. Each server stores a 1 Kb file using 1024 bit asymmetric keys and 128 bit symmetric keys.

	P_{CKM} Shutdown	P_{CKM} Wakeup
2 servers	5.6 ± 0.5	4.4 ± 1.1
5 servers	9.2 ± 1.2	7.4 ± 0.9
10 servers	16.7 ± 2.8	15.7 ± 1.0
20 servers	33.3 ± 18.8	30.4 ± 18.8

Table 6 shows the performance of the CKM protocol itself, that is, excluding the 10-30 seconds it typically takes to start up or dispose the EC2 instances. From these results we conclude that the protocol indeed is practical.

Most applications will only require storage of small files such as cryptographic keys. To reflect this, the servers in the benchmark all store and retrieve secret files of size 1 Kb. Storing larger secrets of course increases the execution time, but the size of secrets was found to have relatively little impact: For example, storing 100 Mb instead of 1 Kb secrets roughly costs 2 seconds extra. The reason for this is that that encryption and decryption of secrets take place locally and only the encryption keys are shared.

Also, the results in Table 6 are benchmarks with all servers located in the same Amazon region (with network latency time being roughly 5-10 ms). Other benchmarks have been carried out with servers located worldwide, again with only little impact on the performance: As an example, five servers located across Europe, US, and Singapore were found to decrease performance by roughly 10 percent compared to a single-region setup.

Detecting cut-off attacks. As already discussed, cut-off attacks cannot be prevented, but in case a cut-off attack do in fact occur, the cut-off server S itself will always notice that something is wrong. In order to make this detection as likely as possible in practice, the servers should listen for (authentic) abort messages from the other servers and if such an abort message is received, a server should immediately forward the message to all other servers and to the application. Also, letting the servers wait some time after completing the AKEs, but before sending their shares over the network, will in practice make the task of breaking security by cutting-off servers considerably harder, because the adversary must then silence the cut-off server for at least an amount of time corresponding to this delay before being able to collect shares. Inserting such delays comes, of course, at the price of decreased protocol performance (and are not included in the benchmarks above).

Entropy in the cloud. The servers in the P_{CKM} protocol require sources of good randomness in order to generate keys, shares, etc. In the full version of this paper

[11] this is modelled by letting the servers be *probabilistic* Turing machines. In practice, however, this randomness has to come from somewhere. Perhaps the most straightforward solution is to require a random seed to be passed to the P_{CKM} protocol from the application and then expand the seed using a secure pseudo-random generator. If done correctly, a polynomial-time adversary will not be able to distinguish the expanded randomness from true randomness if the initial seed is truly random.

However, this just pushes the problem of finding good randomness to the application layer. Another approach is to let the P_{CKM} obtain its randomness from the operating system, for example by using the `SecureRandom` Java class which as the default on Linux obtains a random seed from the OS entropy pool though the `\dev\random` interface that blocks until enough entropy has been gathered from the internal clock, network traffic, etc. A somewhat surprising finding from the implementation was that this seriously affects the performance of P_{CKM} . For example, in the case of five servers, this approach was found to cause a slowdown of 5-10 times for wakeup and 15-20 times for wakeup compared to the benchmark results in Table 6 that use the non-blocking, but potentially less secure, `\dev\urandom` that never blocks, but instead falls back to generating pseudo-random numbers using SHA1 when the OS entropy pool is empty: It takes a considerable time for the entropy pool to acquire enough entropy in newly started virtual instances in the Amazon cloud environment.

Acknowledgements. The authors would like to thank Tim Rasmussen for providing the implementation of the protocol as part of his Master's thesis [36].

References

1. Almansa, J.F., Damgård, I., Nielsen, J.B.: Simplified threshold RSA with adaptive and proactive security. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 593–611. Springer, Heidelberg (2006)
2. Alon, N., Kaplan, H., Krivelevich, M., Malkhi, D., Stern, J.P.: Scalable secure storage when half the system is faulty. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 576–587. Springer, Heidelberg (2000)
3. Blakely, G.R.: Safeguarding cryptographic keys. National Computer Conference Proceedings A.F.I.P.S 48, 313–317 (1979)
4. Bogetoft, P., et al.: Secure multiparty computation goes live. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS, pp. 309–325. ACM (2012)
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE Computer Society (2001)
7. Canetti, R., Gennaro, R., Herzberg, A.: Proactive security: Long-term protection against break-ins. *Crypto Bytes* 3, 1–8 (1997)
8. Canetti, R., Halevi, S., Herzberg, A.: Maintaining authenticated communication in the presence of break-ins. *J. Cryptology* 13(1), 61–105 (2000)
9. Canetti, R., Riva, B., Rothblum, G.N.: Refereed delegation of computation. *Inf. Comput.* 226, 16–36 (2013)

10. Chung, K.-M., Kalai, Y.T., Vadhan, S.P.: Improved delegation of computation using fully homomorphic encryption. In: Rabin (ed.) [35], pp. 483–501
11. Damgård, I., Jakobsen, T.P., Nielsen, J.B., Pagter, J.I.: Secure key management in the cloud. Cryptology ePrint Archive, Report 2013/626 (2013), <http://eprint.iacr.org/>
12. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, Canetti (eds.) [37], pp. 643–662
13. Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Transactions on Information Theory* 22(6), 644–654 (1976)
14. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. *Des. Codes Cryptography* 2(2), 107–125 (1992)
15. Amazon EC2 instance types, <http://aws.amazon.com/ec2/instance-types>
16. The Encrypting File System (EFS). A white paper from Microsoft Corporation, <http://technet.microsoft.com/en-us/library/cc700811.aspx>
17. Danish Energy Auctions, <http://energiauktion.dk>
18. Fu, K., Frans Kaashoek, M., Mazières, D.: Fast and secure distributed read-only file system. *ACM Trans. Comput. Syst.* 20(1), 1–24 (2002)
19. Garay, J.A., Gennaro, R., Jutla, C.S., Rabin, T.: Secure distributed storage and retrieval. *Theor. Comput. Sci.* 243(1-2), 363–389 (2000)
20. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin (ed.) [35], pp. 465–482
21. Gentry, C.: Computing on encrypted data. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 477–477. Springer, Heidelberg (2009)
22. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), <http://crypto.stanford.edu/craig>
23. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) STOC, pp. 218–229. ACM (1987)
24. HELib, a software library implementing fully homomorphic encryption (copyrighted by IBM) (2012), <https://github.com/shaih/HELib>
25. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)
26. Howard, J.H.: An overview of the Andrew File System. In: Winter 1988 USENIX Conference Proceedings, pp. 23–26 (1988)
27. Kamara, S., Mohassel, P., Raykova, M.: Outsourcing multi-party computation. IACR Cryptology ePrint Archive, 2011:272 (2011)
28. Kher, V., Kim, Y.: Securing distributed storage: Challenges, techniques, and systems. In: Atluri, V., Samarati, P., Yurcik, W., Brumbaugh, L., Zhou, Y. (eds.) StorageSS, pp. 9–25. ACM (2005)
29. Krawczyk, H.: Distributed fingerprints and secure information dispersal. In: Anderson, J., Toueg, S. (eds.) PODC, pp. 207–218. ACM (1993)
30. Lakshmanan, S., Ahamad, M., Venkateswaran, H.: Responsive security for stored data. In: Proceedings of the 23rd International Conference on Distributed Computing Systems, ICDCS 2003, p. 146. IEEE Computer Society, Washington, DC (2003)
31. Libert, B., Yung, M.: Adaptively secure forward-secure non-interactive threshold cryptosystems. In: Wu, C.-K., Yung, M., Lin, D. (eds.) *InsCrypt 2011*. LNCS, vol. 7537, pp. 1–21. Springer, Heidelberg (2012)

32. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, Canetti [37], pp. 681–700
33. Partisia, <http://partisia.com>
34. Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM* 36(2), 335–348 (1989)
35. Rabin, T. (ed.): CRYPTO 2010. LNCS, vol. 6223. Springer, Heidelberg (2010)
36. Rasmussen, T.: Key Management in the Cloud. Master's thesis, Aarhus University, Aabogade 34, DK-8200 Aarhus N, Denmark. Master's Thesis (2012)
37. Safavi-Naini, R., Canetti, R. (eds.): CRYPTO 2012. LNCS, vol. 7417. Springer, Heidelberg (2012)
38. Shamir, A.: How to share a secret. *Commun. ACM* 22(11), 612–613 (1979)
39. ShareMind, <http://sharemind.cyber.ee>
40. Spencer, B.P., Noveck, D., Robinson, D., Thurlow, R.: The NFS version 4 protocol. In: Proceedings of the 2nd International System Administration and Networking Conference, SANE (2000)
41. Wright, C.P., Martino, M.C., Zadok, E.: NCryptfs: A secure and convenient cryptographic file system. In: Proceedings of the Annual USENIX Technical Conference, pp. 197–210. USENIX Association (2003)
42. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167. IEEE Computer Society (1986)

Estimating Key Sizes for High Dimensional Lattice-Based Systems

Joop van de Pol and Nigel P. Smart

Dept. Computer Science,
University of Bristol,
United Kingdom

joop.vandepol@bristol.ac.uk, nigel@cs.bris.ac.uk

Abstract. We revisit the estimation of parameters for use in applications of the BGV homomorphic encryption system, which generally require high dimensional lattices. In particular, we utilize the BKZ-2.0 simulator of Chen and Nguyen to identify the best lattice attack that can be mounted using BKZ in a given dimension at a given security level. Using this technique, we show that it should be possible to work with lattices of smaller dimensions than previous methods have recommended, while still maintaining reasonable levels of security. As example applications we look at the evaluation of AES via FHE operations presented at Crypto 2012, and the parameters for the SHE variant of BGV used in the SPDZ protocol from Crypto 2012.

1 Introduction

Estimating parameters for lattice-based cryptographic systems is a major problem. Such systems are becoming increasingly of interest since, to the best of our knowledge, they offer resistance to attacks that arise from the future development of a quantum computer; and in addition can offer functionality not found in traditional public key systems. This problem of parameter estimation becomes more pronounced when one considers the lattice-based schemes underlying Fully Homomorphic Encryption (FHE) [7]. This is particularly tricky as the lattice dimension in such schemes needs to be very large, so large in fact that it is unclear whether our existing methods for parameter estimation even apply. It is to this task that the current paper is focused.

The traditional measure of security of a lattice is the estimated root Hermite value δ_B (see later for a definition), for a lattice basis B output by a lattice basis reduction algorithm. In the literature one sees statements such as a δ_B of 1.05 as being “not secure”, but a value of δ_B of 1.005 as being “secure”. These values are given, and evidence is presented for the correctness of such statements, when in the context of relatively low lattice dimension. It is then assumed that such statements also hold when applied to large dimensional lattices, since the overall lattice dimension is not assumed to affect the difficulty of lattice reduction too much. However, such an extrapolation is clearly not valid; lattice basis reduction

will be harder in higher dimension. Hence, it is not realistic to expect the same value of δ_B to be achievable in high dimension as it is in low dimension.

In various works on FHE, for example [8], a method to produce parameter estimates which extrapolates the run time of existing lattice basis reduction implementations is used. This extrapolation is needed so as to obtain security estimates for high dimensional lattices, which are out of the reach of existing software. In particular this line of approach follows from the analysis of Lindner and Peikert [10], where an extrapolation of the performance of the Block Korkine Zolotarev (BKZ) [14] algorithm in NTL is performed. This itself poses some problems as the implementation of BKZ within NTL is very old (dating from the 1990's in some respects) and does not take into account the various optimizations and improvements which have been introduced over the years.

It turns out that on one hand the analysis in Lindner and Peikert extrapolates the run times of an implementation which does not use modern techniques, whilst on the other hand we show that the parameter estimates are too conservative. This could be explained by the fact that Lindner and Peikert look at a decoding attack, as opposed to our examination of a distinguishing attack. The decoding attack is slightly more powerful than the distinguishing attack. The decoding attack could benefit from the application of extreme pruning techniques, and the type of analysis conducted here, but it is unclear how one could analytically analyse the application of extreme pruning to decoding.

The BKZ algorithm, as one would implement it today, has a number of parameters which one can set to obtain different run-times and output qualities. Such parameters include the block size β , the number of rounds R of BKZ one runs (where each round consists of $d - \beta$ applications of finding short vectors in β dimensional projected lattices), and so-called *pruning parameters* for the search in the projected lattices. Fortunately, in [2], Chen and Nguyen present a simulation algorithm for their improved variant of BKZ. This simulation algorithm allows one to estimate the output quality of a lattice produced by the BKZ algorithm when performing R rounds with block size β . They also provide an estimate for the number of basic operations needed to perform each search, for varying values of the block size β . The term basic operation is deliberately fuzzy, but in this paper we shall take it to mean the number of nodes visited in all of the searches in the projected lattices.

Using the simulation algorithm in [2] one obtains the following “standard” method of determining the hardness of a *given* set of lattice security parameters. One first estimates the value of δ_B one would need to obtain so as to break the system, one then uses the BKZ simulator to determine how many operations this would require, and then one can deem the parameters to be secure or not. However, this in itself implies that the parameters have already been chosen, which have probably been done via appealing to the above rule of thumb in relation to “secure” values of δ_B , and by extrapolation of the runtime of existing software.

We start this work with the idea of achieving a more rational method of obtaining suitable parameters for lattice-based systems in high dimension; with

a special focus on FHE systems. We will still be utilizing the simulation algorithm of [2], but in a way to *generate* parameters as opposed to testing them. In FHE systems the underlying hard problem is essentially the bounded distance decoding problem associated to LWE based lattices. This in effect has three parameters the dimension n (i.e. the ring dimension when considering ring-LWE based schemes such as the BGV system [1]), the modulus q and the distance between a lattice vector and the target vector. In LWE systems, this last quantity is essentially given by the standard deviation r chosen in the Gaussian sampling of the error vector. For fixed n we know that as the ratio r/q becomes larger the problem becomes harder to solve.

In BGV it is common to fix the value of r , and hence the only parameters one can play with are q and n . On one hand we would like q to be large so as to allow deeper circuits to be evaluated by the FHE scheme, but a large q implies low security by the above rule of thumb. To compensate for this one also selects large values of n , as can be seen in [8] where rings of dimension over 60000 are considered. Thus there is a tension in selecting q and n , between the evaluation power and the security of the resulting scheme.

In this paper we adopt the following approach. We first select a security parameter sec . This is a value, such as 80, 128 or 256, for which we feel that visiting 2^{sec} nodes in a BKZ algorithm is infeasible. Then, for a particular lattice dimension d (which for reasons we will explain later satisfies $d \geq n$) we determine the best δ_B one could obtain via a BKZ algorithm limited to visiting 2^{sec} nodes. This step is performed by using the BKZ 2.0 simulator from [2] called with various values of β and R on the estimated Gram-Schmidt lengths of an LLL-reduced basis of a random, d dimensional lattice. The notion of a random lattice will be explained in the next section. In this way the δ_B we obtain is not a fixed value (such as 1.005) but is in essence a function of d and sec . We then utilize this δ_B value in the distinguishing attack analysis of Micciancio and Regev [11], so as to obtain an equation linking n and q , in a way which guarantees 2^{sec} security. This equation can then be combined with any equation linking q and n needed to obtain evaluation of circuits of the correct depth, so as to then obtain a given set of parameters for a given specific application and/or system.

It should be noted first and foremost that things change over time. The available computing power increases as time passes by and new algorithms or attacks can be discovered. Furthermore, it is tricky to make claims about the security of lattice schemes, because it is often unclear how the behaviour of attacks in low dimensions extrapolates to higher dimensions. This work analyses one attack, which is currently believed to be the best generic attack against lattice-based schemes. It is currently unknown whether generic attacks are the best attack in every setting. In structured lattices, such as ideal or symplectic lattices, there may exist better attacks that are not yet known to the cryptographic community. Finally, in order to have confidence in any cryptographic scheme, there should be a reasonably large margin between parameters that are trivially broken and recommended secure ones. It is important to take this into account, especially when selecting parameters for lattice-based schemes.

2 Lattice Background

In this section we present the basics on lattices which we will require, and in addition present our notation.

A (full rank) lattice of dimension d is the discrete subgroup of \mathbb{R}^d generated (over \mathbb{Z}) by a set of vectors $[\mathbf{b}_1, \dots, \mathbf{b}_d]$ in \mathbb{R}^d called the basis. It is common to represent the basis as a matrix B in which row i of the matrix B is given by the vector \mathbf{b}_i (all vectors will be row vectors). Note that this is mathematically not so nice as we then always deal with row vectors, but from a programming point of view it is nicer due to being able to deal with swapping rows (i.e. basis vectors) via pointer arithmetic. Therefore, this convention is common in the literature on lattice basis reduction. We write

$$\mathcal{L}(B) = \{\mathbf{z} \cdot B : \mathbf{z} \in \mathbb{Z}^d\}.$$

A lattice basis is not unique and each basis is related to another via the relation $B' = Z \cdot B$ where $Z \in \text{GL}_d(\mathbb{Z})$, i.e. Z is an integer matrix with determinant ± 1 . We often use the shorthand \mathcal{L} for $\mathcal{L}(B)$ if the underlying basis (which of course does not really matter) is clear.

On vectors in \mathbb{C}^d we can define the following norms

$$\|\mathbf{x}\|_p = \begin{cases} \left(\sum_{i=1}^d |x_i|^p\right)^{1/p} & p \neq \infty \\ \max_{i=1}^d |x_i| & p = \infty. \end{cases}$$

Being a discrete structure there is a well defined quantity of a non-zero minimum of the lattice, which we denote by

$$\lambda_1^{(p)}(\mathcal{L}) := \min\{\|\mathbf{x}\|_p : \mathbf{x} \in \mathcal{L}, \mathbf{x} \neq \mathbf{0}\}.$$

We can also define the successive minima $\lambda_i^{(p)}(\mathcal{L})$, which are defined as the smallest radius r such that the d -dimensional ball of radius r centred on the origin contains i linearly independent lattice points. To ease notation, and because we will be mainly working with the 2-norm, we write $\lambda_i(\mathcal{L}) = \lambda_i^{(2)}(\mathcal{L})$.

For any basis B we define the fundamental region as the set

$$\mathcal{P}(B) = \left\{ \sum_{1 \leq i \leq d} x_i \cdot \mathbf{b}_i : x_i \in [0, 1) \right\}.$$

The d -dimensional volume, $\Delta(\mathcal{L}) = \text{Vol}(\mathcal{P}(B))$, is called the fundamental volume, and can be computed via $\Delta(\mathcal{L}) = |\det(B)|$. It is clear that this quantity is an invariant of the lattice, and does not depend on the precise basis chosen. The *dual* \mathcal{L}^* of a lattice \mathcal{L} is the set of all vectors $\mathbf{y} \in \mathbb{R}^d$ such that $\mathbf{y} \cdot \mathbf{x}^\top \in \mathbb{Z}$ for all $\mathbf{x} \in \mathcal{L}$. Given a basis matrix B of \mathcal{L} we can compute the basis matrix B^* of \mathcal{L}^* via $B^* = (B^{-1})^\top$. Hence we have $\Delta(\mathcal{L}^*) = 1/\Delta(\mathcal{L})$.

The classic result in lattice theory (a.k.a. geometry of numbers), is that of Minkowski, which relates the minimal distance to the fundamental volume.

Theorem 1 (Minkowski’s Theorem). *For any d dimensional lattice \mathcal{L} we have*

$$\lambda_1(\mathcal{L}) \leq \sqrt{d} \cdot \Delta(\mathcal{L})^{1/d}.$$

The notion of a random lattice stems from work by Goldstein and Mayer [9]. Consider lattices with a prime determinant p . For large p the vast majority of these lattices are of the following type:

$$\begin{pmatrix} p & & & & \\ x_1 & 1 & & & \\ \vdots & & \ddots & & \\ x_{d-1} & & & & 1 \end{pmatrix}.$$

Goldstein and Mayer show that lattices generated by taking p at random and taking x_i independently and uniformly at random in $\{0, \dots, p - 1\}$ are in some (natural) sense random. These lattices are often studied when considering the behaviour of basis reduction algorithms [12,5].

For such random lattices the first minimum is approximated by the *Gaussian Heuristic*, which states that for a random lattice we have

$$\lambda_1(\mathcal{L}) \approx \sqrt{\frac{d}{2 \cdot \pi \cdot e}} \cdot \Delta(\mathcal{L})^{1/d}.$$

Hermite showed that there is an absolute constant γ_d , depending only on d , such that

$$\lambda_1(\mathcal{L}) \leq \sqrt{\gamma_d} \cdot (\Delta(\mathcal{L}))^{1/d}.$$

The value of γ_d (called “Hermite’s constant”) is, however, only known for $1 \leq d \leq 8$ and $d = 24$.

A specific basis B is said to have *Hermite factor* δ_B^d , or *root Hermite factor* δ_B , if

$$\|\mathbf{b}_1\|_2 = \delta_B^d \cdot \Delta(\mathcal{L})^{1/d}.$$

The root Hermite factor of the *lattice* is said to be the constant δ_L such that

$$\lambda_1(\mathcal{L}) = \delta_L^d \cdot \Delta(\mathcal{L})^{1/d}.$$

In lattice basis reduction algorithms we are trying to determine an output lattice basis such that $\delta_B = \delta_L$, i.e. the first vector in the basis is the shortest vector.

By the Gaussian heuristic we have for a random lattice

$$\delta_L \approx \left(\sqrt{\frac{d}{2 \cdot \pi \cdot e}} \right)^{1/d}.$$

3 Estimating BKZ

In this section we provide an overview of the prior work on analysing the BKZ algorithm and then present our results on estimating the output δ_B from BKZ,

for a specific dimension and with an explicit limit on the number of nodes evaluated. In later sections we will use this analysis to estimate parameters for the LWE based systems used in FHE schemes.

BKZ OVERVIEW. Throughout the paper we assume the input basis to the BKZ algorithm has been LLL reduced (i.e., reduced by the LLL algorithm). The BKZ algorithm, as modified in [2] and called BKZ 2.0, is parameterized by two parameters R and β and operates as follows. The algorithm executes the following round function R times. In each round we iterate the index i from one to $d - \beta$, and for each value of i we take the β -dimensional projected lattice generated by the basis vectors $\mathbf{b}_i, \dots, \mathbf{b}_{i+\beta-1}$ projected onto the orthogonal space spanned by the first $i - 1$ basis vectors. A small vector is obtained in the projection of this lattice, and the resulting vector is inserted into the main lattice basis at the i th position. The search for the small vector in the projected lattice is performed by an enumeration method using a heuristic called *extreme pruning* [6].

HISTORICAL BACKGROUND. The line of work aimed at assessing the behaviour of basis reduction algorithms in practice was started by Gama and Nguyen [5]. They considered this behaviour from an experimental point of view and tried to extrapolate it to higher dimensions (although not the astronomical dimensions required in FHE schemes). Specifically, they analyse the behaviour of basis reduction algorithms when applied to solving various lattice problems, such as Hermite-SVP, Approximate SVP and Unique SVP. However, since BKZ 2.0 did not exist at the time, they analysed the original BKZ which did not use extreme pruning and did not abort after a fixed number of rounds, but would instead run until termination.

The most interesting result from these experiments was that basis reduction algorithms output a basis B which appeared to solve Hermite-SVP, i.e. finding a short basis vector, with Hermite Factor δ_B^d . The interesting part is that on average, the δ_B observed in practice was much smaller than theoretical worst-case bounds obtained from analysing the reduction algorithms theoretically. It should be noted that this worst-case behaviour was tied to the basis of the particular lattice, rather than to the lattice itself. Applying the basis reduction algorithms to a ‘randomized’ basis of the same lattice resulted in average-case rather than worst-case behaviour. Gama and Nguyen conjectured that the value of δ_B of the output basis depends mostly on the basis reduction algorithm that was used and not on the input lattice (unless this lattice has special structure). The value also depended on the dimension d but appeared to converge quickly as d increases.

Gama and Nguyen drew several conclusions. Most importantly, they concluded that with the basis reduction algorithms available at that time, $\delta_B = 1.01$ was the best reachable root-Hermite factor. They also examined the run-time of exact SVP solvers and concluded that up to dimension 60 the shortest vector problem could be solved within an hour, whereas dimension 100 seemed out of reach. They also observed that BKZ with block sizes much higher than 25 was not realistic in higher dimensions due to run-time constraints. Once again,

these observations were before the discovery of extreme pruning and before the adoption of aborting BKZ after a fixed number of rounds R .

It should also be noted that this work was not aimed at cryptography, but only at basis reduction algorithms in a general setting. Hence, Gama and Nguyen did not experiment specifically with lattices that arise from a cryptographic setting, but instead with random lattices from the Goldstein Mayer distribution [9] (as described in Section 2) and some specially structured lattices for the unique shortest vector problem.

Gama, Nguyen and Regev in 2010 [6] proposed improved heuristics for solving SVP using enumeration via a technique called extreme pruning. Potentially, this technique could be used with the enumeration of the β dimensional projected lattices within the BKZ algorithm. However, this heuristic technique requires a pretty good estimate of the length of the shortest vector. But Gama and Nguyen had already observed that the projected lattices that occur in BKZ with low block size (say $\beta < 50$) do not follow the distribution of random lattices. More specifically, these projected lattices did not adhere to the Gaussian Heuristic, which would have given a good approximation to the length of the shortest vector. Thus, extreme pruning cannot trivially be applied to BKZ with low block size.

But then Chen and Nguyen [2] made the observation that the projected lattices that appear in BKZ for higher block sizes (say $\beta > 50$) behave like random lattices as far as the Gaussian Heuristic is concerned. This enables the introduction of extreme pruning and several other heuristic improvements to BKZ, resulting in the BKZ 2.0 algorithm outlined above. The BKZ 2.0 algorithm is able to reduce lattices with much higher block sizes in practice than the original BKZ. This observation about the projected lattices and Gaussian Heuristic also allowed Chen and Nguyen to create a simulator for BKZ 2.0, which simulates the behaviour of the algorithm on the lengths of the Gram-Schmidt vectors of the basis. This makes it much easier to heuristically explain (for large enough block size) the behaviour of BKZ in practice and the associated output δ_B , even for block sizes that we might not be able to run in practice.

Chen and Nguyen use the simulator to estimate the approximate security of the NTRU encryption scheme and the Gentry-Halevi FHE challenges. Specifically for the challenges by Gentry and Halevi they reason as follows. From the parameters of the scheme they can derive that they require a root-Hermite factor of δ_B . They use the simulation to estimate that this requires R rounds of BKZ with block size β (starting from an LLL-reduced basis). Using an upper bound for the cost of a block size β enumeration derived from experiments, they convert the R rounds into the number of enumeration nodes (given that each round consists of $d - \beta$ enumerations where d is the dimension of the lattice). This number of nodes gives a rough estimate for the bit-security of the specific parameters of the scheme.

OUR APPROACH. In the heuristic approach by Chen and Nguyen (and others), an estimated security level is essentially derived from a system with given parameters. However, we would like to choose our parameters according to a given security

level. Thus, we reverse the analysis by Chen and Nguyen and try to answer the question: Given a security level of sec such that the adversary can only perform 2^{sec} operations, how should we choose our parameters such that our system is secure against this adversary?

Say we choose the dimension d of a Goldstein Mayer lattice and a security level sec . Now, an adversary can attempt to run BKZ with block size β , for varying β . For each β , we can approximate the cost of a single enumeration using the tables from Chen and Nguyen [2]. Then, we can compute how many enumerations we could maximally perform with this block size without exceeding 2^{sec} nodes. This bound on the number of enumerations gives us a bound on the number of rounds R , say $R(\beta, d, \text{sec})$, for the dimension d as well. Now we can simulate the behaviour of $R(\beta, d, \text{sec})$ rounds of BKZ with block size β on a random LLL-reduced basis of a d -dimensional Goldstein Mayer lattice, using the simulation algorithm from [2]. This allows us to predict the root-Hermite factor δ_B of the output basis from BKZ. Thus, on input of d , sec and β , we obtain a value of δ_B . If we perform this procedure for all block sizes β , we find an estimated value of δ_B (one for each β). Taking the minimum of all such δ_B we obtain an estimate for the best value of δ_B which can be obtained by an adversary which is limited to enumerating at most 2^{sec} nodes.

Doing this for a number of increasing dimensions we find the data in Table 1 for the estimate of the best δ_B an adversary can obtain in a given dimension d . Unsurprisingly we see that as the dimension increases the best value of δ_B that one can obtain also increases, although the increase is not too pronounced. This can be explained as follows. If we allow BKZ with block size β to run indefinitely, so for unbounded R , the simulation suggests that δ_B of the output basis converges to some value that seems to only depend on β (consistent with the observations from [5]). However, as the dimension increases, performing a round of BKZ becomes more costly. Furthermore, the simulation also indicates that in higher dimensions it converges more slowly to this value δ , i.e., it takes a larger number of rounds R to reach it. In higher dimensions, BKZ with block size β reaches a worse δ_B in $R(\beta, d, \text{sec})$ rounds than BKZ with block size $\beta' < \beta$ in $R(\beta', d, \text{sec})$ rounds. The results in Table 1 assume that the estimated number of nodes visited during an enumeration reported in [2] cannot be improved by further algorithmic improvements.

For $d > 2^{17}$, the BKZ simulator is rather slow, but for the applications in Section 5 dimensions up to 2^{17} are sufficient. Therefore, only dimensions up to d^{17} were considered here. The value of δ_B achievable when evaluating at most 2^{256} nodes is achieved by performing BKZ with block size 250. Since Chen and Nguyen only give the cost of enumerations up to block size 250, it is possible that an attacker could use BKZ with a higher block size and achieve a better δ_B , while evaluating no more than 2^{256} nodes. Because it was not possible to reproduce the costs for varying block sizes and because it is unclear how to realistically extrapolate the costs to higher block sizes, the value of δ_B here corresponds to block size 250.

Table 1. Smallest achievable δ_B by BKZ in dimension d and evaluating at most 2^{sec} nodes

	d							
sec	1024	2048	4096	8192	16384	32768	65536	131072
80	1.0081	1.0081	1.0084	1.0084	1.0088	1.0088	1.0092	1.0092
128	1.0067	1.0067	1.0067	1.0069	1.0069	1.0069	1.0069	1.0072
256	1.0055	1.0055	1.0055	1.0055	1.0055	1.0055	1.0055	1.0055

4 Estimating LWE Parameters

Our goal is to provide estimates for LWE parameters for specific cryptographic systems in large dimensions, given the estimates in the previous section. Before proceeding we recap a little on notation and prior analysis so as to fix notation. The LWE problem, and hence to the best of our knowledge the ring-LWE problem, is based upon arithmetic in q -ary lattices.

q -ARY LATTICES. A q -ary lattice \mathcal{L} of dimension n is one such that $q\mathbb{Z}^n \subset \mathcal{L} \subset \mathbb{Z}^n$ for some integer q . Note that all integer lattices are q -ary lattices for a value of q which is an integer multiple of $\Delta(A)$. Our interest will be in special lattices which are q -ary for a value of q much smaller than the determinant. Much of our discussion follows that in [11].

Suppose we are given a matrix $A \in \mathbb{Z}_q^{n \times d}$, with $d \geq n$, we then define the following two d -dimensional q -ary lattices.

$$\begin{aligned} \Lambda_q(A) &= \{ \mathbf{y} \in \mathbb{Z}^d : \mathbf{y} = \mathbf{z} \cdot A \pmod{q} \text{ for some } \mathbf{z} \in \mathbb{Z}^n \}, \\ \Lambda_q^\perp(A) &= \{ \mathbf{y} \in \mathbb{Z}^d : \mathbf{y} \cdot A^\top = 0 \pmod{q} \}. \end{aligned}$$

Suppose we have $\mathbf{y} \in \Lambda_q(A)$ and $\mathbf{y}' \in \Lambda_q^\perp(A)$ then we have $\mathbf{y} = \mathbf{z} \cdot A$ and $\mathbf{y}' \cdot A^\top = 0 \pmod{q}$. This implies that

$$\mathbf{y} \cdot \mathbf{y}'^\top = (\mathbf{z} \cdot A) \cdot \mathbf{y}'^\top = \mathbf{z} \cdot (\mathbf{y}' \cdot A^\top)^\top \in q \cdot \mathbb{Z}.$$

Hence, the two lattices are, up to normalisation, duals of each other. We have $\Lambda_q(A) = q \cdot \Lambda_q^\perp(A)^*$ and $\Lambda_q^\perp(A) = q \cdot \Lambda_q(A)^*$.

To fix ideas consider the following example; Let $n = 2, m = d = 3, q = 1009$ and set

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 5 & 6 \end{pmatrix}.$$

To define a basis B of $\Lambda_q(A)$ we can take the row-HNF of the 5×3 matrix

$\begin{pmatrix} A \\ q \cdot I_3 \end{pmatrix}$ to obtain

$$B = \begin{pmatrix} 1009 & 0 & 0 \\ 1 & 1 & 0 \\ 336 & 0 & 1 \end{pmatrix}.$$

The basis of $\Lambda_q^\perp(A)$ is given by

$$B^* = q \cdot ((B^\top)^{-1}) = \begin{pmatrix} 1 & -1 & -336 \\ 0 & 1009 & 0 \\ 0 & 0 & 1009 \end{pmatrix}.$$

The properties of the above example hold in general; namely if q is prime and (in general) if d is a bit larger than n then we have $\Delta(\Lambda_q(A)) = q^{d-n}$ and $\Delta(\Lambda_q^\perp(A)) = q^n$.

We now turn to discussing how short the vectors are that one can find in q -ary lattices. Let us focus on the lattice $\Lambda_q^\perp(A)$, which will be more important for our analysis. We know that this contains vectors of length q (since it is a q -ary lattice), we assume that lattice reduction will output a basis B with root Hermite factor δ_B for some value of δ_B . This means that computationally the shortest vector we can produce in the lattice $\Lambda_q^\perp(A)$ will be of size

$$\min(q, \delta_B^d \cdot q^{n/d})$$

since $\Delta(\Lambda_q^\perp(A)) = q^n$.

LWE PROBLEM. The LWE problem is parametrized by four parameters n, d, q and $r = s/\sqrt{2\pi}$. To define the problem we introduce the Gaussian distribution in one variable with parameter s (and mean zero) as the distribution with probability distribution function proportional to

$$f(x) = \frac{1}{s} \exp\left(-\frac{\pi \cdot x^2}{s^2}\right).$$

Thus we have that the standard deviation is given by $r = s/\sqrt{2 \cdot \pi}$. The (spherical) multivariate normal distribution on \mathbb{R}^n , with Gaussian parameter s (resp. standard deviation r) is given by

$$f(\mathbf{x}) = \frac{1}{s} \exp\left(-\frac{\pi \cdot \|\mathbf{x}\|_2^2}{s^2}\right) = \frac{1}{r \cdot \sqrt{2 \cdot \pi}} \exp\left(-\frac{\|\mathbf{x}\|_2^2}{2 \cdot r^2}\right).$$

Sampling from this distribution is performed by simply sampling each component of the vector \mathbf{x} independently from $N(0, r)$.

The discrete Gaussian distribution, with support on the lattice \mathcal{L} with Gaussian parameter s (equivalently standard deviation $r = s/\sqrt{2 \cdot \pi}$, denoted $D_{\mathcal{L},s}$, is the probability distribution on \mathcal{L} which selects $\mathbf{x} \in \mathcal{L}$ with probability proportional to $\exp(-\pi \cdot \|\mathbf{x}\|_2^2/s^2)$.

Definition 1 (LWE Decision Problem). Given (A, \mathbf{v}) where $A \in \mathbb{Z}_q^{n \times d}$ and $\mathbf{v} \in \mathbb{Z}_q^d$ determine which of the following distributions \mathbf{v} is from:

1. \mathbf{v} is chosen uniformly at random from \mathbb{Z}_q^d .
2. $\mathbf{v} = \mathbf{s} \cdot A + \mathbf{e}$ where $\mathbf{e}, \mathbf{s} \leftarrow D_{\mathbb{Z}^n, s}$.

The link between LWE and q -ary lattices is then immediately obvious. Given A and \mathbf{v} the decision problem is to determine whether \mathbf{v} is a random point or an element which is close to a point in the lattice $\Lambda_q(A)$.

The natural ‘‘attack’’ against the decision LWE problem is to first find a short vector \mathbf{w} in the dual lattice $\Lambda_q(A)^*$ and then check whether $\mathbf{w} \cdot \mathbf{v}^\top$ is close to an integer. If it is, one concludes that the input vector is an LWE sample, whereas if it is not one concludes that the input vector is random. Thus to ensure security, following the argument in [11, Section 5.4.1], we require

$$r \geq \frac{1.5}{\|\mathbf{w}\|_2}.$$

Now from earlier, we deduce that when applying lattice reduction to the lattice $\Lambda_q(A)^* = \frac{1}{q}\Lambda_q^\perp(A)$ we will obtain a vector \mathbf{w} with

$$\|\mathbf{w}\|_2 \approx \frac{1}{q} \min(q, \delta_B^d \cdot q^{n/d}).$$

The point is that we have some freedom in choosing d here, since it is related to the number of LWE samples we take. In the traditional analysis [11] one assumes δ_B is already given and one then applies calculus to minimize the above estimate for $\|\mathbf{w}\|_2$ by picking d as a function of q , n and δ_B . But as we presented in Section 3 the value of δ_B is essentially a function of d and sec .

OUR ANALYSIS. We make the heuristic assumption that the behaviour of applying the BKZ lattice basis reduction technique to the d dimensional lattice $\Lambda_q(A)^*$ performs roughly the same as the application to the Goldstein Mayer lattices in Section 3. For the above distinguishing attack to fail to work we require

$$q^{n/d-1} \geq \frac{1.5}{r \cdot \delta_B^d} = c_{r,d,\text{sec}}.$$

For fixed values of r we can derive, using the method in Section 3, values of $c_{r,d,\text{sec}}$ for any value of sec and d that we require. We therefore require, to ensure security, that for all $d \geq n$ we have

$$n \log_2 q - d \log_2 q \geq d \cdot \log_2 c_{r,d,\text{sec}}.$$

Note that, as a sanity check, for fixed n this means we have an upper bound on $\log_2 q$ of

$$\log_2 q \leq \min_{d>n} \frac{-d \cdot \log_2 c_{r,d,\text{sec}}}{d - n}. \quad (1)$$

We end this section by discussing what this means for a simple LWE based system at the security level of 80 bits; in particular we determine what the maximum value of q could be when we fix $n = 4000$ and $r = 3.2$. We first derive a more detailed version of Table 1 and use linear interpolation to determine estimated δ_B values for dimensions not in our table; this needs to be done once and for all, in all of our analysis. Retuning to considering our specific values of

n and r : We enumerate all $d > n$ up to 2^{17} , and use the linear interpolation of Table 1 to determine a value of δ_B for reducing a lattice of dimension d at this security level. This enables us to obtain an upper bound on $\log_2 q$, over all values of d , from Equation 1. Indeed we obtain an upper bound of $\log_2 q$ of 195 and the “best” value of d for the distinguishing attack comes out as $d = 8045$ with $\delta_B \approx 1.0084$. We compare this with the traditional analysis which assumes δ_B given and then computed d as $d = \sqrt{n \cdot \log(q)/\log(\delta_B)}$, which would give us a value of $d \approx 8045$ as well, as expected. However, we reiterate that this traditional method of obtaining d comes from somehow estimating the value of δ_B one would obtain in performing BKZ on lattices of (an as yet unknown) dimension d .

5 Application of Our Method to Two Examples

As a first application we re-evaluate the parameters in (the full version of) [8]. The authors of [8] determine parameters for their SHE scheme so as to homomorphically evaluate large circuits, including the AES circuit. They select a security level equivalent to 80 bits of security and derive sizes for the resulting parameters to evaluate circuits of multiplicative depth L , for various values of L . In order to compare the results, we will consider the same security level.

In [8, Appendix C], they use the security analysis by Lindner and Peikert [10] to derive a lower bound on the approximate ring dimension $n = \phi(m)$ depending on the largest modulus Q , standard deviation r and the security level sec , which guarantees the security of the scheme. In particular the lower bound is

$$n \geq \frac{\log(Q/r)(\text{sec} + 110)}{7.2}. \tag{2}$$

To guarantee the functionality of the L -leveled homomorphic scheme, they then derive an estimate on the size of Q needed to evaluate a circuit of depth L , this is given by

$$Q \approx 2^{22.5 \cdot L - 3.6} \cdot r \cdot n^L.$$

The individual moduli in the SHE scheme are given by

$$p_0 \approx 2^{23.9} \cdot n, \quad p_i \approx 2^{11.3} \sqrt{n} \text{ for } i = 1, \dots, L - 2, \quad p_{L-1} \approx \sqrt{n} + 11,$$

and

$$P \approx 2 \cdot 308^L \cdot \zeta^{L-2} \cdot r \cdot n^{L/2}.$$

Combining the two equations for Q , setting $\text{sec} = 80$, $\zeta = 8$ and $r = 3.2$ they derive values of n and Q for various values of L .

In our analysis, we replace the security-related lower bound (2) on n by the equivalent upper bound from Equation (1) on Q , given n . Now, we increase n , in steps of 100 from a given starting value, until the upper bound on Q is above the estimate for Q needed to ensure correct evaluation of a circuit of multiplicative depth L . We present our results, and the comparison with those in [8] in Table 2.

As one can see the methodology for choosing parameters in this paper results in roughly the same values for the moduli, but also produces significantly smaller lattice dimensions. In practice, this will translate into faster overall performance figures for the SHE scheme.

Table 2. Table comparing the estimates from [8] with our estimates. Here $\ell_2(x) = \log_2(x)$.

L	Estimates from [8]					Our Estimates				
	n	$\ell_2(p_0)$	$\ell_2(p_i)$	$\ell_2(p_{L-1})$	$\ell_2(P)$	n	$\ell_2(p_0)$	$\ell_2(p_i)$	$\ell_2(p_{L-1})$	$\ell_2(P)$
10	9326	37.1	17.9	7.5	177.3	7100	36.7	17.7	6.6	163.3
20	19434	38.1	18.4	8.1	368.8	14300	37.7	18.2	7.0	369.0
30	29749	38.7	18.7	8.4	564.2	21600	38.3	18.5	7.3	550.6
40	40199	39.2	18.9	8.6	762.2	28500	38.7	18.7	7.5	743.3
50	50748	39.5	19.1	8.7	962.1	35500	39.0	18.6	7.6	937.9
60	61376	39.8	19.2	8.9	1163.5	42900	39.3	18.9	7.7	1134.3
70	72071	40.0	19.3	9.0	1366.1	50400	39.5	19.1	7.9	1332.1
80	82823	40.2	19.4	9.1	1569.8	57900	39.7	19.2	7.9	1530.9
90	93623	40.4	19.5	9.2	1774.5	65500	39.9	19.3	8.0	1730.6

As another example we look at the example parameters used in the SPDZ MPC protocol, see [3,4]. In [3] parameters are given for instantiating the SPDZ MPC protocol over fields of prime characteristic of size 32, 64 and 128 bits. The resulting parameter sets have lattice dimensions 8192, 16384 and 32768 respectively. In the prime characteristic case greater efficiency is obtained in the protocol if one has lattices of dimension a power of two. If one performs the same analysis as in [3] for the case of characteristic two one finds that the resulting dimension will have size *roughly* 8192.

By using our analysis we find that we can securely use dimensions of size roughly 4096 (for characteristic two), 8192 (for prime characteristic of size roughly 2^{32}) 16384 (for prime characteristic of size roughly 2^{64}) 16384 (for prime characteristic of size roughly 2^{128}). Thus we obtain a more efficient scheme for the case of characteristic two and for very large prime characteristic only. The reason for the lack of a general improvement is that for odd prime characteristic, the dimensions are restricted to a power of two due to scheme specific efficiency.

Acknowledgements. The authors would like to acknowledge the partial support of of the ERC (Advanced Grant ERC-2010-AdG-267188-CRIPTO), the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) (under agreement number FA8750-11-2-0079)¹, and

¹ The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

by EPSRC (Grant EP/I03126X). The second author has been supported in part by a Royal Society Wolfson Merit Award. The authors also thank the anonymous reviewers for their helpful comments and insights.

References

1. Brakerski, Z., Gentry, C., Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS, pp. 309–325. ACM (2012)
2. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
3. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. IACR Cryptology ePrint Archive, 2012:642 (2012)
4. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, Canetti (eds.) [13], pp. 643–662
5. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
6. Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010)
7. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC, pp. 169–178. ACM (2009)
8. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, Canetti (eds.) [13], pp. 850–867
9. Goldstein, D., Mayer, A.: On the equidistribution of Hecke points. Forum Math. 15, 165–189 (2003)
10. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
11. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post-Quantum Cryptography, pp. 147–192. Springer (2009)
12. Nguyen, P.Q., Stehlé, D.: LLL on the average. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 238–256. Springer, Heidelberg (2006)
13. Safavi-Naini, R., Canetti, R. (eds.): CRYPTO 2012. LNCS, vol. 7417. Springer, Heidelberg (2012)
14. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In: Budach, L. (ed.) FCT 1991. LNCS, vol. 529, pp. 68–85. Springer, Heidelberg (1991)

Sub-linear Blind Ring Signatures without Random Oracles

Essam M. Ghadafi

University of Bristol, United Kingdom

Abstract. In this paper we provide the first provably secure blind ring signature construction that does not rely on random oracles, which solves an open problem raised by Herranz and Laguillaumie at ISC 2006. We present different instantiations all of which are round-optimal (i.e. have a two-move signing protocol), yield sub-linear size signatures, and meet strong security requirements. In order to realize our constructions efficiently, we construct a sub-linear size set membership proof which works in the different bilinear group settings, which may be of independent interest.

As a secondary contribution, we show how to generically combine our set membership proof with any secure signature scheme meeting some conditions to obtain ring signatures whose security does not rely on random oracles. All our constructions work over the efficient prime-order bilinear group setting and yield signatures of sub-linear size. In addition, our constructions meet strong security requirements: namely, anonymity holds under full key exposure and unforgeability holds against insider-corruption. Finally, we provide some example instantiations of the generic construction.

1 Introduction

Background. A Ring Signature (RS), introduced by Rivest, Shamir and Tauman [45], allows a signer to choose an arbitrary set of signers called a “ring” and anonymously sign a message on behalf of the ring providing that the signer himself is a member of the ring. Generating the signature does not require the cooperation of other members of the ring and hence they need not even be aware of their inclusion in the ring.

Besides correctness, the security of ring signatures [45,8] requires anonymity and unforgeability. Informally, anonymity requires that a signature does not reveal the identity of the ring member who produced it. On the other hand, unforgeability requires that an adversary cannot forge new signatures on behalf of an honest ring. In [8], the authors provide various variants of those requirements. We will prove the security of our constructions under the strongest definitions provided in [8].

Ring signatures were originally used for anonymous leaking of authoritative secrets. For other applications of ring signatures see, e.g. [45,39,21].

Like group blind signatures [38], blind ring signatures extend blind signatures to the multi-signer setting. However, in contrast to the former, the latter provide more flexibility in the choice of the group as it is done in an ad hoc manner without requiring prior cooperation or join protocols. In addition, anonymity of the signer is not revocable. Besides the three security properties required from traditional ring signatures, the security

of blind ring signatures requires blindness which informally states that members of the ring cannot learn which message is being signed on behalf of the ring. Also, they cannot match a signature to its signing session.

Applications of blind ring signatures include distributed e-cash systems [38], where a client's e-coin is signed by a member of a coalition of banks chosen in an ad hoc manner. The choice of the coalition could be specified by either the issuing bank or the client himself. Other applications of the primitive include multi-authority e-voting and e-auction systems.

Related Work. BLIND RING SIGNATURES. Only a few blind ring signature schemes [16,50,35,51] were proposed. All of those constructions are secure in the Random Oracle Model (ROM) [7]. The scheme in [16] yields signatures of linear size and its security requires both random oracles and the generic group model [48]. In [50], the authors presented a static blind ring signature scheme that requires both random oracles and the generic group model. This scheme requires that the group (i.e. the ring) is fixed and hence it yields signatures of constant size. The schemes in [35,51] also yield signatures of linear size. We note here that the blindness requirement of [35] was proven using a different game than the standard definition for blindness [37,43] where the adversary only interacts once with the challenger and does not get to see the final signature.

RING SIGNATURES. The first construction by Rivest, Shamir and Tauman [45] is based on trapdoor permutations and is secure in the random oracle model. Subsequently, other constructions relying on random oracles followed [4,12,36,21,40].

A few constructions which do not rely on random oracles were proposed. Bender et al. [8] gave a generic construction requiring generic ZAPs [22], making it inefficient. They also gave two constructions for two-signer rings. Other constructions which do not require random oracles include [47,17,14,46,15]. The constructions in [46,15] use a weaker notion of unforgeability than the one we use in this paper.

All existing constructions apart from [17] (which yields sub-linear size signatures in composite-order groups in the Common Reference String (CRS) model) and [21,40] (which yield signatures of constant size in the ROM) yield signatures of linear size.

The Challenges. The subtlety one faces when designing blind ring signatures lies in the dual privacy requirement: that is the dilemma of having parts of the witness of the same proof of knowledge coming from different parties who do not trust each other. On the one hand, the signer needs to hide his identity and parts of the signature that could identify him (i.e. the anonymity requirement). On the other hand, the user needs to hide the message and parts of the signature which could reveal the linkage between a signature and its signing session (i.e. the blindness requirement). One might consider addressing such an issue by resorting to secure multiparty computation, however, such an approach would massively degrade the efficiency of the resulting construction.

Due to the nature of random oracles, in the random oracle model this obstacle is easier to tackle by, for instance, using divertible proofs of knowledge e.g. [20,41]. In the standard model, the issue is more subtle. To get around this issue, we exploit some properties of Groth-Sahai proofs [33], namely: the randomizability of the proofs [6] and the ability to transform some proofs without knowledge of the original witness

[29,27]. This way we obtain the required divertibility needed to achieve the dual privacy requirement.

The second technical challenge is that unforgeability of ring-related signatures requires that the signature is bound to the ring in order to prevent the adversary from transforming a signature by some ring into a signature by a different ring. The scenario is more serious when the construction involves a malleable proof system such as the Groth-Sahai proof system which we use in our constructions. This is because the malleability of the proof system allows one to easily transform a proof for some statement into another proof for a related statement.

When constructing ring signatures, one can easily bind the signature to the ring by simply signing both the message and the ring. For instance, this could be efficiently achieved by signing the hash of the concatenation of both the message and the ring. Unfortunately, this approach does not work in a blind signing protocol. That is because necessitating that the message remains hidden from the signer, one needs to prove that such hashing was applied correctly without revealing the message, which cannot be efficiently realized due to the complex structure of hash functions. To bind the signature to the ring w.r.t. which it was produced, we deploy a different approach. We use a signature scheme that simultaneously signs a pair of messages to construct a partially-blind signature scheme where we hide the actual message from the signer but we include the details of the ring as the public information shared between the user and the signer.

The remaining challenge which is inherent even in traditional ring signatures is the size of the signatures. Almost all previous blind ring signatures e.g. [16,35,51] and most existing traditional ring signatures e.g. [8,47,14] yield signatures whose size grows linearly with the size of the ring. This limitation is usually inherited from the underlying OR proof used to prove that the signature verifies w.r.t. a verification key contained in the ring without revealing which one it is. In [17], the authors used some techniques from private information retrieval applications to construct a membership proof that has a sub-linear size. Unfortunately, their protocol is limited to the rather inefficient composite-order bilinear group setting. As a part of our contribution, we adapt their technique to the prime-order setting and thus we obtain a sub-linear size set membership proof that works in the 3 different settings of prime-order bilinear groups. Although this on its own is not a major contribution, it is of independent interest as we believe it could have further applications beyond the scope of this paper.

Our Contribution. Our main contribution is the first blind ring signature schemes that do not rely on idealized assumptions. This solves a problem that remained open since 2006 [35]. To realize our constructions efficiently, we instantiate the idea used for the membership proof from [17] in the prime-order bilinear group setting. All our constructions yield signatures of sub-linear size and thus are shorter than those of previous constructions. In addition, our schemes meet strong security requirements and their security is based solely on falsifiable complexity assumptions [44].

Our final contribution is a generic construction that combines our set membership proof with any signature scheme in the standard model satisfying some conditions to get sub-linear size ring signatures without random oracles. Again, our focus is on constructions in the efficient prime-order bilinear group setting.

Paper Organization. The rest of the paper is organized as follows: In Section 2, we give some preliminary definitions. In Section 3, we define blind ring signatures and present their security definitions. In Section 4, we present a new set membership proof. In Section 5, we present our blind ring signature constructions. Finally, in Section 6 we present our ring signature constructions.

2 Preliminaries

Notation. Given a probability distribution S , we denote by $x \leftarrow S$ the operation of selecting an element according to S . If A is a probabilistic machine, we denote by $A(x_1, \dots, x_n)$ the output distribution of A on inputs (x_1, \dots, x_n) . By p.p.t., we mean running in probabilistic polynomial time in the relevant security parameter. By $[1, n]$, we denote the set $\{1, 2, \dots, n\}$. A function $v(\cdot) : \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible in c if for every polynomial $p(\cdot)$ and all sufficiently large values of c , it holds that $v(c) < \frac{1}{p(c)}$.

Bilinear Groups. A bilinear group is a tuple $\mathcal{P} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \tilde{G}, e)$ where $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are groups of a prime order p and G and \tilde{G} generate \mathbb{G}_1 and \mathbb{G}_2 , respectively. The function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear map. We use multiplicative notation for all the groups although usually \mathbb{G}_1 and \mathbb{G}_2 are chosen to be additive. We let $\mathbb{G}_1^\times := \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$ and $\mathbb{G}_2^\times := \mathbb{G}_2 \setminus \{1_{\mathbb{G}_2}\}$. For clarity, elements from \mathbb{G}_2 will be accented with $\tilde{\cdot}$.

Following [28], we classify prime-order bilinear groups into 3 main types:

- **Type-1:** This is the symmetric pairing setting in which $\mathbb{G}_1 = \mathbb{G}_2$.
- **Type-2:** $\mathbb{G}_1 \neq \mathbb{G}_2$ but there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.
- **Type-3:** Again $\mathbb{G}_1 \neq \mathbb{G}_2$, but now there is no known efficiently computable isomorphism.

We assume that all groups are cyclic and there is an algorithm BGrpSetup that takes a security parameter λ and a type $\text{tp} \in \{1, 2, 3\}$ and outputs a description of bilinear groups of Type- tp .

Complexity Assumptions. We will use the following assumptions from the literature:

CDH. For a group $\mathbb{G} := \langle G \rangle$ of a prime order p given $(G, G^a, G^b) \in \mathbb{G}^3$ for $a, b \leftarrow \mathbb{Z}_p$, it is hard to compute G^{ab} .

DDH. For a group $\mathbb{G} := \langle G \rangle$ of a prime order p given $(G, G^a, G^b, C) \in \mathbb{G}^4$ for $a, b \leftarrow \mathbb{Z}_p$, it is hard to decide whether or not $C = G^{ab}$.

Co-CDH [18]. In Type-2 bilinear groups given $(G, G^a, \tilde{G}, \tilde{G}^b) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$ for $a, b \leftarrow \mathbb{Z}_p$, it is hard to compute G^{ab} .

Co-CDH* [18]. In Type-3 bilinear groups given $(G, G^a, G^b, \tilde{G}, \tilde{G}^b) \in \mathbb{G}_1^3 \times \mathbb{G}_2^2$ for $a, b \leftarrow \mathbb{Z}_p$, it is hard to compute G^{ab} .

SXDH. The DDH assumption holds in both groups \mathbb{G}_1 and \mathbb{G}_2 .

DLIN [11]. For Type-1 bilinear groups where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ and G generates \mathbb{G} , given the tuple $(G^a, G^b, G^{ra}, G^{sb}, G^t)$ where $a, b, r, s, t \in \mathbb{Z}_p$ are unknown, it is hard to tell whether $t = r + s$ or t is random.

- **Pairing Product Equation (PPE):** $\prod_{i=1}^n e(A_i, \underline{Y}_i) \cdot \prod_{i=1}^m \prod_{j=1}^n e(\underline{X}_i, \underline{Y}_j)^{k_{i,j}} = t_T.$
- **Multi-Scalar Multiplication Equation (MSME)¹:** $\prod_{i=1}^n A_i^{y_i} \prod_{i=1}^m \underline{X}_i^{b_i} \prod_{i=1}^m \prod_{j=1}^n \underline{X}_i^{k_{i,j} y_j} = T.$
- **Quadratic Equation (QE) in \mathbb{Z}_p :** $\sum_{i=1}^n a_i \underline{y}_i + \sum_{i=1}^m \underline{x}_i b_i + \sum_{i=1}^m \sum_{j=1}^n \underline{x}_i \underline{y}_j = t.$

Fig. 1. Types of equations one can use Groth-Sahai proofs for

q-SDH [10]. For a group $\mathbb{G} := \langle G \rangle$ of a prime order p given $(G, G^x, \dots, G^{x^q}) \in \mathbb{G}^{q+1}$ for $x \leftarrow \mathbb{Z}_p$, it is hard to output a pair $(c, G^{\frac{1}{x+c}}) \in \mathbb{Z}_p \times \mathbb{G}$ for an arbitrary $c \in \mathbb{Z}_p \setminus \{-x\}$.

WFCDH [26]. In Type-1 bilinear groups, given $(G, G^a, G^b) \in \mathbb{G}^3$ for $a, b \leftarrow \mathbb{Z}_p$, it is hard to output a tuple $(G^r, G^{ra}, G^{rb}, G^{rab}) \in \mathbb{G}^{\times 4}$ for an arbitrary $r \in \mathbb{Z}_p$.

AWFCDH [26]. In asymmetric bilinear groups, given $(G, G^a, \tilde{G}) \in \mathbb{G}_1^2 \times \mathbb{G}_2$ for $a \leftarrow \mathbb{Z}_p$, it is hard to output a tuple $(G^b, G^{ab}, \tilde{G}^b, \tilde{G}^{ab}) \in \mathbb{G}_1^{\times 2} \times \mathbb{G}_2^{\times 2}$ for an arbitrary $b \in \mathbb{Z}_p$.

q-DHSDH [26]. In symmetric bilinear groups, given $(G, H, K, G^x) \in \mathbb{G}^4$ for $x \leftarrow \mathbb{Z}_p$, and $q - 1$ tuples $(W_i := (K \cdot G^{u_i})^{\frac{1}{x+v_i}}, U_{1,i} := G^{u_i}, U_{2,i} := H^{u_i}, V_{1,i} := G^{v_i}, V_{2,i} := H^{v_i})_{i=1}^{q-1}$, where $u_i, v_i \leftarrow \mathbb{Z}_p$, it is hard to output a new tuple $(W^*, U_1^*, U_2^*, V_1^*, V_2^*)$ of this form.

q-ADHSDH [26]. In asymmetric bilinear groups, given $(G, F, K, G^x, \tilde{G}, \tilde{G}^x) \in \mathbb{G}_1^4 \times \mathbb{G}_2^2$ for $x \leftarrow \mathbb{Z}_p$, and $q - 1$ tuples $(W_i := (K \cdot G^{u_i})^{\frac{1}{x+v_i}}, U_{1,i} := G^{u_i}, \tilde{U}_{2,i} := \tilde{G}^{u_i}, V_{1,i} := F^{v_i}, \tilde{V}_{2,i} := \tilde{G}^{v_i})_{i=1}^{q-1}$ for $u_i, v_i \leftarrow \mathbb{Z}_p$, it is hard to output a new tuple $(W^*, U_1^*, \tilde{U}_2^*, V_1^*, \tilde{V}_2^*)$ of this form.

Groth-Sahai (GS) Proofs. Groth and Sahai [33,34] introduced a proof system in the CRS model that yields Non-Interactive Witness-Indistinguishable (NIWI) and Zero-Knowledge (NIZK) proofs. The system can be instantiated in composite-order or prime-order bilinear groups. The equations one can prove with the system are in Figure 1 where in the description $X_1, \dots, X_m, Y_1, \dots, Y_n \in \mathbb{G}$, $x_1, \dots, x_m, y_1, \dots, y_n \in \mathbb{Z}_p$ are secret variables (hence underlined), whereas $A_i, T \in \mathbb{G}$, $a_i, b_i, k_{i,j}, t \in \mathbb{Z}_p$, $t_T \in \mathbb{G}_T$ are public constants. Note that in the asymmetric setting, there are two types of MSM equations depending on which group the elements belong to. The system is defined by a tuple of algorithms

$$(\text{GSSetup}, \text{GSProve}, \text{GSVerify}, \text{GSExtract}, \text{GSSimSetup}, \text{GSSimProve}).$$

Algorithm **GSSetup** takes as input the description of a bilinear group \mathcal{P} and outputs a *soundness* reference string crs and an extraction key xk . **GSProve** takes as input a reference string crs , a witness and a set of equations, and outputs a proof Ω for the satisfiability of the equations. For clarity, we will underline the elements of the witness in the description of the equations. **GSVerify** takes as input a reference string crs , a proof Ω and a set of equations, and outputs 1 if the proof is valid or 0 otherwise. In the rest of the paper we will omit the set of equations from the input to the **GSVerify** algorithm.

GSExtract takes as input a soundness reference string crs , the extraction key xk and a valid proof Ω , and outputs the witness used in the proof. GSSimSetup takes as input a bilinear group \mathcal{P} and outputs a *simulation* string crs_{Sim} and a trapdoor key tr that allows to simulate proofs. GSSimProve takes as input crs_{Sim} and the simulation trapdoor tr and produces a simulated proof Ω_{Sim} .

The system works by committing to the elements of the witness (using the algorithm GSCCommit) and then producing a proof of satisfiability for each equation. If a witness component is involved in multiple equations, the same commitment is re-used when verifying the proofs which makes the proofs correlated.

The proofs come in two flavors: the soundness setting yields extractable proofs, whereas the simulation setting yields simulatable proofs. The system's security requires that the distributions of strings crs and crs_{Sim} are indistinguishable and simulated proofs are indistinguishable from real proofs.

The proof system has perfect completeness, perfect soundness, composable witness-indistinguishability and composable zero-knowledge. For formal definitions of those properties refer to [34,30].

As formalized by [6], GS proofs can be rerandomized by rerandomizing the associated GS commitments and updating the proofs accordingly so that we obtain fresh proofs that are unlinkable to the original ones. Rerandomizing a proof requires knowledge of neither the witness nor the associated randomness used in the original GS commitments. We define an algorithm GSRandomize which takes as input a CRS crs and a proof Ω , and outputs a proof Ω' which is a randomized version of the proof Ω .

For details of the different instantiations see [34,32].

(Partially) Blind Signatures. Blind Signatures (BS) [19] allow a user to obtain a signature on a message hidden from the signer. Partially Blind Signatures (PBS) [3] are an extension of blind signatures where unlike blind signatures, part of the message to be signed is shared public information info which is known to both parties.

The signing protocol $(\text{PBSObtain}(\text{pk}, m, \text{info}), \text{PBSSign}(\text{sk}, \text{info}))$ in these schemes is an interactive protocol between a user who knows a message m and a signer who possesses a secret signing key sk and both parties know the public information info . If the protocol is completed successfully, the user obtains a signature Σ on the message m and the information info .

The security of partially blind signatures [5] is similar to that of blind signatures [37,43] and consists besides correctness of blindness and unforgeability. Intuitively, blindness requires that an adversarial signer does not learn the message being signed and he cannot match a signature to its signing session. In the game, the adversary (modeling an adversarial signer) chooses two messages m_0 and m_1 and common information info and then interacts with the honest user who requests signatures on those messages in an arbitrary order unknown to the adversary. The same information info is used in both interactions. If completed successfully, the adversary gets the two final signatures and wins if it tells the order in which the messages were signed with a probability that is non-negligibly greater than $1/2$.

On the other hand, unforgeability deals with an adversarial user whose goal is to obtain $k + 1$ distinct message/signature pairs after only k interactions w.r.t. the public information info with the honest signer.

<p>PBSSetup(1^λ):</p> <ul style="list-style-type: none"> • $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \tilde{G}, e) \leftarrow \text{BGrpSetup}(1^\lambda, 3)$. • $\mathcal{P} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \tilde{G}, e)$. • $(\text{crs}, \text{sk}) \leftarrow \text{GSSetup}(\mathcal{P})$. • $F, K, L, T \leftarrow \mathbb{G}_1$. • Return $\text{param}_{\text{PBS}} := (\mathcal{P}, \text{crs}, F, K, L, T)$. 	<p>PBSKeyGen($\text{param}_{\text{PBS}}$):</p> <ul style="list-style-type: none"> • Choose $a \leftarrow \mathbb{Z}_p$ and set $A := G^a$ and $\tilde{A} := \tilde{G}^a$. • $\text{sk} := a, \text{pk} := (A, \tilde{A})$. Return (sk, pk). <p>PBSVerify($\text{pk}, (M, \tilde{M}), \text{info}, \Sigma$):</p> <ul style="list-style-type: none"> • Parse Σ as Ω_{sig}. • Return 1 if $\text{GSVerify}(\text{crs}, \Omega_{\text{sig}}) = 1$ Else Return 0.
<p>The signing protocol $(\text{PBSObtain}(\text{pk}, (M, \tilde{M}), \text{info}), \text{PBSSign}(\text{sk}, \text{info}))$</p>	
<ul style="list-style-type: none"> • PBSObtain \rightarrow PBSSign <ul style="list-style-type: none"> – Choose $s \leftarrow \mathbb{Z}_p$ and compute $S := G^s, \tilde{S} := \tilde{G}^s$ and $C := M \cdot T^s$. – $\Omega \leftarrow \text{GSProve}(\text{crs}, (M, \tilde{M}, S, \tilde{S}), \{e(\underline{M}, \tilde{G}) = e(G, \tilde{M}) \wedge e(\underline{S}, \tilde{G}) = e(G, \tilde{S}) \wedge e(\underline{M}, \tilde{G})e(T, \underline{S}) = e(C, \tilde{G})\})$. – Send (C, Ω) to PBSSign. • PBSSign \rightarrow PBSObtain <ul style="list-style-type: none"> – Abort if $\text{GSVerify}(\text{crs}, \Omega) \neq 1$. – Choose $u, v \leftarrow \mathbb{Z}_p$ and set $U' := G^u, V := F^v, W := (K \cdot T^u \cdot C \cdot L^{\text{info}})^{\frac{1}{a+v}}, \tilde{U}' := \tilde{G}^u, \tilde{V} := \tilde{G}^v$. – Send $\sigma := (W, U', \tilde{U}', V, \tilde{V})$ to PBSObtain. • PBSObtain <ul style="list-style-type: none"> – Compute $U := U' \cdot S$ and $\tilde{U} := \tilde{U}' \cdot \tilde{S}$. – Abort if $e(U, \tilde{G}) \neq e(G, \tilde{U}), e(F, \tilde{V}) \neq e(V, \tilde{G})$ or $e(W, \tilde{A} \cdot \tilde{V}) \neq e(K \cdot M \cdot L^{\text{info}}, \tilde{G})e(T, \tilde{U})$. – $\Omega_{\text{sig}} \leftarrow \text{GSProve}(\text{crs}, (V, \tilde{V}, W, U, \tilde{U}), \{e(\underline{V}, \tilde{G}) = e(F, \tilde{V}) \wedge e(\underline{U}, \tilde{G}) = e(G, \tilde{U}) \wedge e(\underline{W}, \tilde{A} \cdot \tilde{V})e(T^{-1}, \tilde{U}) = e(K \cdot M \cdot L^{\text{info}}, \tilde{G})\})$. – Output $\Sigma := \Omega_{\text{sig}}$. 	

Fig. 2. The partially blind signature scheme (in the asymmetric setting) [26,27]

A PARTIALLY BLIND SIGNATURE SCHEME [26,27]. In [26,2], the authors gave a blind signature based on the DLIN, WFCDH and q-DHSDH/q-ADHSDH assumptions in the symmetric setting or the SXDH, AWFCDH and q-ADHSDH assumptions in the asymmetric setting. The message space of the scheme is $\mathcal{M} := \{(G^m, \tilde{G}^m) | m \in \mathbb{Z}_p\}$. To get a partially blind scheme, we use a variant of their blind scheme based on the modified signature scheme from [27] whose message space is $\mathcal{M} \times \mathbb{Z}_p$ as highlighted in [27].

The high-level idea behind the scheme is that the user commits to his message and sends the commitment along with GS proofs to prove that it is well-formed to the signer. The signer uses his secret key to produce a signature on the commitment and the public information info. When the user receives the signature, he uses the randomness he used in the commitment to modify the signature from one on the commitment to one on the message itself. The final signature is a set of GS proofs of knowledge of such a signature. The blindness of the scheme is ensured by the NIWI/NIZK properties of GS proofs and the fact that the first-round commitment is information-theoretically hiding. The scheme in the asymmetric setting is in Figure 2.

Ring Signatures. A ring signature [45] is a tuple $\text{RS} := (\text{RSSetup}, \text{RSKeyGen}, \text{RSSign}, \text{RSVerify})$ of p.p.t. algorithms. Those algorithms are defined as follows; where to aid notation all algorithms (bar RSSetup and RSKeyGen) take as implicit input param_{RS} (output by RSSetup):

- $\text{RSSetup}(1^\lambda)$ takes as input a security parameter λ and outputs common public parameters param_{RS} .

- $\text{RSKeyGen}(\text{param}_{\text{RS}})$ takes as input param_{RS} and outputs a pair (sk, pk) of secret/public keys.
- $\text{RSSign}(\text{sk}_i, m, \mathcal{R})$ takes as input a secret key sk_i , a message $m \in \mathcal{M}$ (where \mathcal{M} is the message space) and a ring $\mathcal{R} := \{\text{pk}_1, \dots, \text{pk}_n\}$ with the condition that $\text{pk}_i \in \mathcal{R}$ and outputs a signature Σ on the message m .
- $\text{RSVerify}(m, \Sigma, \mathcal{R})$ takes as input a message m , a ring signature Σ and a ring \mathcal{R} and outputs 1 if the signature is on the message m w.r.t. ring \mathcal{R} or 0 otherwise.

The security properties required from ring signatures are informally as follows:

- **Correctness:** All honestly generated signatures are accepted by the RSVerify algorithm.
- **Anonymity:** An adversary cannot tell which ring member produced a signature.
- **Unforgeability:** An adversary cannot output a valid signature Σ^* on a message m^* and w.r.t. an honest ring \mathcal{R}^* unless the adversary obtained such a signature by querying the sign oracle on (m^*, \mathcal{R}^*) .

For detailed definitions and variants of those properties, we refer the reader to [8]. We use the strongest variants from [8], namely: anonymity under full key exposure and unforgeability against insider-corruption.

3 Blind Ring Signatures

Definition 1 (Blind Ring Signatures). A *Blind Ring Signature* (BRS) is a tuple $(\text{BRSSetup}, \text{BRSSignKeyGen}, \langle \text{BRSSObtain}, \text{BRSSign} \rangle, \text{BRSSVerify})$ of p.p.t. algorithms. Those algorithms are defined as follows; where to aid notation all algorithms (bar BRSSetup and BRSSignKeyGen) take as implicit input $\text{param}_{\text{BRS}}$ (output by BRSSetup):

- $\text{BRSSetup}(1^\lambda)$ takes as input a security parameter λ and outputs public parameters $\text{param}_{\text{BRS}}$.
- $\text{BRSSignKeyGen}(\text{param}_{\text{BRS}})$ is run by a signer Signer_i to generate his pair of secret/public keys (sk, pk) .
- $\langle \text{BRSSObtain}(m, \mathcal{R}), \text{BRSSign}(\text{sk}_i, \mathcal{R}) \rangle$ is an interactive two-party protocol between a user User and a signer in the ring \mathcal{R} where $\text{pk}_i \in \mathcal{R}$. If the protocol completes successfully, User obtains a blind ring signature Σ on the message m . If any of the parties abort, User outputs \perp . This protocol is initiated by a call to BRSSObtain . The choice of the ring could be influenced by either the signer or the user.
- $\text{BRSSVerify}(m, \Sigma, \mathcal{R})$ verifies if the blind ring signature Σ is on the message m w.r.t. the ring \mathcal{R} .

A tuple $\text{BRS} := (\text{BRSSetup}, \text{BRSSignKeyGen}, \langle \text{BRSSObtain}, \text{BRSSign} \rangle, \text{BRSSVerify})$ is a secure blind ring signature if it has correctness, anonymity, unforgeability and blindness which are defined as follows:

Definition 2 (Correctness). A blind ring signature BRS is correct if for any $\lambda \in \mathbb{N}$, any polynomial $n(\cdot)$, any $\{(\text{pk}_i, \text{sk}_i)\}_{i=1}^{n(\lambda)}$ output by BRSSignKeyGen , any message m in the message space \mathcal{M} and any index $i \in [1, n(\lambda)]$ if Σ is the output of the honest interaction $\langle \text{BRSSObtain}(m, \mathcal{R}), \text{BRSSign}(\text{sk}_i, \mathcal{R}) \rangle$ where $\mathcal{R} = \{\text{pk}_1, \dots, \text{pk}_{n(\lambda)}\}$ then BRSSVerify accepts the signature Σ .

ANONYMITY. We use a strong definition for anonymity where we allow the adversary to use corrupt keys as well as obtaining the secret keys for the two challenge signers i_0 and i_1 and hence capturing security against *full key exposure* and *adversarially-chosen keys* [8].

Definition 3 (Anonymity). A blind ring signature BRS satisfies anonymity if for any $\lambda \in \mathbb{N}$ and polynomial $n(\cdot)$ the success probability of any p.p.t. adversary \mathcal{A} in the following game is negligibly close to $1/2$:

1. The challenger generates $\text{param}_{\text{BRS}}$ and key pairs $\{(\text{pk}_i, \text{sk}_i)\}_{i=1}^{n(\lambda)}$ using $\text{BRSKeyGen}(\text{param}_{\text{BRS}})$. \mathcal{A} is given $\text{param}_{\text{BRS}}$ and $\mathcal{S} := \{\text{pk}_i\}_{i=1}^{n(\lambda)}$.
2. Throughout the game, \mathcal{A} has access to a sign oracle OSign with which it interacts to obtain signatures on messages and by signers in rings of its choice (providing that the signer's public key is in \mathcal{R} and \mathcal{S}). \mathcal{A} can also ask for the secret key of any signer to be revealed at any stage of the game.
3. \mathcal{A} outputs two distinct indices i_0 and i_1 and a ring \mathcal{R} with the only condition that $\text{pk}_{i_0}, \text{pk}_{i_1} \in \mathcal{R}$. It then interacts with the challenger to get a signature by signer Signer_{i_b} where $b \leftarrow \{0, 1\}$.
4. \mathcal{A} outputs a bit b' and succeeds if $b = b'$.

UNFORGEABILITY. Informally, a blind ring signature is unforgeable if the adversary cannot output a blind ring signature w.r.t. to a ring \mathcal{R} of honest signers that was never produced by the sign oracle. Due to the blind nature of the signing protocol and as in standard blind signatures, we follow the $(k, k + 1)$ -unforgeability definition [37,43]. The following definition also protects against insider corruption [8]:

Definition 4 (Unforgeability). A blind ring signature BRS is unforgeable if for any $\lambda \in \mathbb{N}$, and polynomial $n(\cdot)$ the success probability of any p.p.t. adversary \mathcal{A} in the following game is negligible:

1. The challenger generates $\text{param}_{\text{BRS}}$ and key pairs $\{(\text{pk}_i, \text{sk}_i)\}_{i=1}^{n(\lambda)}$ using $\text{BRSKeyGen}(\text{param}_{\text{BRS}})$. \mathcal{A} is given $\text{param}_{\text{BRS}}$ and $\mathcal{S} := \{\text{pk}_i\}_{i=1}^{n(\lambda)}$.
2. Throughout the game, \mathcal{A} has access to the same oracles as in the anonymity game (Definition 3).
3. \mathcal{A} outputs $k + 1$ pairs of message/signature $\{(m_i, \Sigma_i)\}_{i=1}^{k+1}$, and a ring \mathcal{R}^* . \mathcal{A} wins if all the following conditions hold:
 - (a) All $k + 1$ signatures verify correctly (w.r.t. ring \mathcal{R}^*) and all the messages are distinct.
 - (b) All members of the ring \mathcal{R}^* are honest.
 - (c) \mathcal{A} engaged in at most k interactions with the sign oracle w.r.t. ring \mathcal{R}^* .

Note that our definition above is not of strong unforgeability, i.e., we do not require that the adversary cannot output a new signature on an old message.

BLINDNESS. Informally, a blind ring signature is blind if an adversary (modeling a dishonest behavior of signers in the ring) does not learn the message it is signing. Moreover, it cannot link a signature to its signing session.

Definition 5 (Blindness). *A blind ring signature BRS satisfies blindness if for any $\lambda \in \mathbb{N}$ and polynomial $n(\cdot)$ the success probability of any p.p.t. adversary \mathcal{A} in the following game is negligibly close to $1/2$:*

1. *The challenger generates $\text{param}_{\text{BRS}}$ and sends it to \mathcal{A} .*
2. *\mathcal{A} outputs a ring \mathcal{R} (the keys of which are possibly adversarially chosen) and two messages m_0 and m_1 .*
3. *The honest user interacts with the adversary concurrently to get signatures on those two messages in an arbitrary order unknown to the adversary by choosing a bit $b \leftarrow \{0, 1\}$. \mathcal{A} is sent the signatures Σ_b, Σ_{1-b} . If any of the interactions did not finish or any of the signatures do not verify w.r.t. \mathcal{R} , \mathcal{A} is not informed about the other signature.*
4. *\mathcal{A} outputs a bit b' and wins if $b = b'$.*

As noted by [35], since the ring is public, it is a natural requirement that the two challenge signatures are signed w.r.t. the same ring. Otherwise, blindness can be trivially broken if different rings were used.

Unlike the blindness definition used in [35], which does not give the final challenge signatures to the adversary, we give the adversary the two final signatures. This is important because blindness should capture the case that a blind signature is not linkable to its signing session. Take, for example, the e-cash application where the issuing bank eventually gets to see the coins when they are deposited. Also, unlike [35], our definition allows the adversary to use corrupt keys of its choice which again provides a strong definition of blindness [1,42].

4 Sub-linear Size Set Membership Proof over Prime-Order Groups

In this section we construct a non-interactive set membership proof. The proof allows a prover to prove that a value X_γ is contained in a set $\{X_1, \dots, X_N\} \in \mathbb{G}^N$. Our construction is based on the underlying idea of the proof in [17] which is specific to the composite-order bilinear group setting and is based on the subgroup decision assumption [13]. Unlike the proof in [17], our proof is more general and works in both composite-order and prime-order bilinear groups. However, for efficiency reasons our focus is on the prime-order setting. We provide different instantiations over the 3 main types of the prime-order setting as summarized in Table 1. We note that it might also be possible to use the recent techniques, e.g. [25], for translating composite-order based protocols to the prime-order setting to obtain a variant of the original protocol in [17] in the prime-order setting.

The idea from [17] is to represent the set by a square $n \times n$ matrix where $n = \lceil \sqrt{N} \rceil$. If N is not square, we can repeat X_1 as many times as required to obtain a set whose size is square. As we will see, this does not affect the complexity of the proof.

The prover knows a secret value X_γ and wants to produce a non-interactive proof that such a value is contained in a square $n \times n$ matrix \mathbf{X} without revealing the secret value.

Thus, we construct a proof for the statement $\Omega_{\text{mem}} := \text{PoK}\{(X_\gamma \in \mathbb{G}) : X_\gamma \in \mathbf{X}\}$, where the matrix is

$$\mathbf{X} = \begin{pmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n,1} & X_{n,2} & \dots & X_{n,n} \end{pmatrix}$$

For simplicity, the proof we give here is in the symmetric setting and is based on the DLIN assumption and can be instantiated in the asymmetric setting as summarized in Table 1.

Let $(\mathbb{G}, \mathbb{G}_T, p, G, e)$ be a description of symmetric bilinear groups. We summarize the proof in the following steps where we assume that $X_\gamma = X_{\alpha,\beta}$ (i.e. X_γ lies in row α and column β) and crs is the reference string used for the proof system:

1. The prover chooses 2 secret binary vectors $\mathbf{y}, \mathbf{z} \in \{0, 1\}^n$ as follows

$$y_i = \begin{cases} 1 & \text{if } i = \alpha, \\ 0 & \text{if } i \neq \alpha \end{cases} \quad z_i = \begin{cases} 1 & \text{if } i = \beta, \\ 0 & \text{if } i \neq \beta \end{cases}$$

The vectors \mathbf{y}, \mathbf{z} will be used to anonymously single out the row and the column containing the secret value, respectively. The prover first needs to prove that each element of those vectors has a value $\in \{0, 1\}$ which is done by the following QE proofs

$$\begin{aligned} \Omega_{y_i} &\leftarrow \text{GSProve}(\text{crs}, (y_i), \{y_i(y_i - 1) = 0\}), \\ \Omega_{z_i} &\leftarrow \text{GSProve}(\text{crs}, (z_i), \{z_i(z_i - 1) = 0\}). \end{aligned}$$

Additionally, the prover needs to prove that each vector contains only a single value of 1. This could be achieved by generating two extra proofs for the equations $\sum_{i=1}^n y_i = 1$ and $\sum_{i=1}^n z_i = 1$, respectively, which only adds two linear QE proofs and no extra commitments to the complexity. Alternatively, if witness-indistinguishability is sufficient, one can prove this for free by exploiting the homomorphic property of GS commitments (which are ElGamal ciphertexts [23] in the SXDH instantiation and Linear ciphertexts [11] in the DLIN instantiation). Thus, by choosing the *GS randomness* used for committing to one of those GS commitments to be the inverse of the sum of the corresponding randomness used for committing to the remaining values in the vector and then by multiplying the GS commitments to the values in each vector, the randomness cancels out and we can recover the sum of the values in the vector which allows the verifier to verify such a claim. Let $\mathcal{C}_{y_i} \leftarrow \text{GSCommit}(y_i, \tau_{y_i})$ and $\mathcal{C}_{z_i} \leftarrow \text{GSCommit}(z_i, \tau_{z_i})$ be the GS commitments used in committing to y_i and z_i , respectively. We set $\tau_{y_n} := -\sum_{i=1}^{n-1} \tau_{y_i}$ and $\tau_{z_n} := -\sum_{i=1}^{n-1} \tau_{z_i}$.

Note that since the randomness $\tau_{y_1}, \dots, \tau_{y_{n-1}}$ and $\tau_{z_1}, \dots, \tau_{z_{n-1}}$ is chosen uniformly, the randomness τ_{y_n} and τ_{z_n} is also uniform.

The verifier can verify that indeed each vector contains only a single value of 1 by checking that $\prod_i^n \mathcal{C}_{y_i} = \prod_i^n \mathcal{C}_{z_i} = \text{GSCommit}(1, 0)$, i.e. the product is equal to a trivial GS commitment to 1.

In total, this step requires $2n$ GS commitments and $2n$ QE proofs.

Table 1. Complexity of the proof

Component/Instantiation	DLIN	DDH $_{\mathbb{G}_1}$ + DLIN $_{\mathbb{G}_2}$		SXDH	
		\mathbb{G}_1	\mathbb{G}_2	\mathbb{G}_1	\mathbb{G}_2
GS Commitments	$9n + 3$	$4n$	$9n + 3$	$4n$	$6n + 2$
GS proofs	$21n + 9$	$12n + 4$	$15n + 3$	$12n + 4$	$10n + 2$
Total	$30n + 12$	$16n + 4$	$24n + 6$	$16n + 4$	$16n + 4$

2. The prover anonymously singles out the row containing his secret value. To do so, for each column j in the matrix compute $X_{\alpha,j} := \prod_{i=1}^n X_{i,j}^{y_i}$. Note that \mathbf{X}_α contains the messages in row α of the matrix \mathbf{X} . The prover generates the following MSME proofs to prove that each $X_{\alpha,j}$ was computed correctly

$$\Omega_{X_{\alpha,j}} \leftarrow \text{GSProve} \left(\text{crs}, (X_{\alpha,j}, \{y_i\}_{i=1}^n), \left\{ \prod_{i=1}^n X_{i,j}^{y_i} \cdot \underline{X_{\alpha,j}^{-1}} = 1 \right\} \right).$$

3. Finally, the prover proves that the value X_γ is contained in the secret vector \mathbf{X}_α . This is achieved by the following MSME proof

$$\Omega_{X_\gamma} \leftarrow \text{GSProve} \left(\text{crs}, (X_\gamma, \{X_{\alpha,i}\}_{i=1}^n, \{z_i\}_{i=1}^n), \left\{ \prod_{i=1}^n \underline{X_{\alpha,i}^{z_i}} \cdot \underline{X_\gamma^{-1}} = 1 \right\} \right).$$

The membership proof Ω_{mem} is $((\mathbf{C}_y, \mathbf{C}_z, \mathbf{C}_{X_\alpha}, \mathbf{C}_{X_\gamma}), (\Omega_y, \Omega_z, \Omega_{X_\alpha}, \Omega_{X_\gamma}))$.

To verify the proof, the verifier verifies the proofs $\Omega_{y_i}, \Omega_{z_i}, \Omega_{X_{\alpha,i}}$ for all $i \in [1, n]$ and Ω_{X_γ} , and checks that $\prod_i^n \mathcal{C}_{y_i} = \prod_i^n \mathcal{C}_{z_i} = \text{GSCommit}(1, 0)$.

Note that when the proof is instantiated over asymmetric bilinear groups we need to commit to the vectors \mathbf{y} and \mathbf{z} in both groups \mathbb{G}_1 and \mathbb{G}_2 and provide a proof for the equality of the commitments for each vector.

Theorem 1. *The set membership proof is correct, sound and zero-knowledge.*

Proof. Correctness and soundness follow from the correctness and soundness of GS proofs and the fact that by checking that $\prod_i^n \mathcal{C}_{y_i} = \text{GSCommit}(1, 0)$ and $\prod_i^n \mathcal{C}_{z_i} = \text{GSCommit}(1, 0)$, the verifier ensures that only one non-zero value is contained in each vector. The witness-indistinguishability of the membership proof also follows from that of GS proofs.

When zero-knowledge is required, all the equations we prove (which are of types QE and MSME) are simulatable at no extra cost. Simply by using trivial witnesses (i.e. 0 for exponent values and 1 for group elements), we can simulate all the proofs. Thus, the proof is zero-knowledge.

Complexity of the Proof. We summarize in Table 1 the size (in group elements) of the proof in the different GS instantiations. We note here that in the asymmetric setting, the total size of the proof is irrespective of whether the set is in \mathbb{G}_1^N or \mathbb{G}_2^N . Although the size of the commitments and proofs are swapped between the two cases, the total size remains the same.

To speed up verification, one can apply batch verification techniques [31,9] to Groth-Sahai proofs.

5 Blind Ring Signature Construction

Overview of the Construction. Some of the recent round-optimal blind signature constructions e.g. [2] are instantiations of Fischlin’s generic construction [24], and combine the GS proof system with commitment and signature schemes that are compatible with one another. The latter is referred to as structure-preserving signatures [2]. In Fischlin’s construction, the user sends a commitment to the message to the signer who in turn returns a signature on the commitment. The user then constructs the final blind signature by encrypting both the signature and the commitment and providing a NIZK proof of knowledge that the signature is valid on the commitment and that the commitment is to the message in question.

We exploit some properties of GS proofs. First, the rerandomizability of the proofs [6]. Second, that they are independent of the public terms in the equations being proven [29,27], which as shown by [29], allows transforming GS NIWI proofs into new NIWI/NIZK proofs by adding some/all of those public terms to the witness without knowledge of the original witness. The latter property was used by [29] to construct a group blind signature scheme.

Additionally, we require that:

1. The verification equations of the signature scheme has the form that all the monomials (e.g. the pairing in the case of PPE equations) involving the message are independent of the signing key, i.e. they involve neither the verification key nor any signature component that depends on the signing key so that we can exploit the second property above. An example scheme satisfying this condition is the automorphic scheme from [26].
2. The signature scheme signs $n + 1$ group elements or n group elements and an integer where n is the size (in group elements) of the commitment so that we bind the signature to the ring. To this end, we require a collision-resistant hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathcal{M}_{\text{SIG}}$ to map the ring into the message space of the signature scheme.

The high-level idea of our generic construction is as follows: The user sends a commitment to the message to the signer. The signer signs the commitment along with the ring information and instead of sending the signature to the user in the clear, sends a GS proof of knowledge Ω'_{sig} of his public key and the signature σ such that the signature is on the public commitment to the message and that it verifies w.r.t. to the signer’s public key. In order to reduce the communication overhead, one does not need to hide the whole signature and it is sufficient to just hide the components which depend on the secret key. One might additionally need to hide other parts of the signature to ensure that the proof is in a transformable form. In addition, using the set membership proof from Section 4, the signer generates a proof Ω'_{mem} to prove that his public key is in the ring. The signer sends proofs Ω'_{sig} and Ω'_{mem} plus any remaining public components of the signature to the user.

If the proofs are valid, the user first rerandomizes the proofs Ω'_{sig} and Ω'_{mem} (and their GS commitments) into Ω_{sig} and Ω_{mem} , respectively. The new proofs are now unlinkable to the original ones. Additionally, he transforms the NIWI proof Ω_{sig} into a NIZK proof

by adding the commitment to the message and the remaining public components of the signature (if any) to the witness of the proof. Finally, the user adds a NIZK proof Ω_{com} to prove that the commitment is indeed to the message. The final blind ring signature is a set of GS proofs ($\Omega_{\text{sig}}, \Omega_{\text{mem}}, \Omega_{\text{com}}$) and their associated GS commitments. It is vital that the proofs are correlated, i.e. proofs Ω_{sig} and Ω_{mem} involve the same public key, and proofs Ω_{sig} and Ω_{com} involve the same commitment. Thanks to the nature of GS proofs, in our instantiations this correlation is directly realized by sharing the same GS commitment for those shared components of the witness between the proofs.

Anonymity is ensured by the NIWI/NIZK properties of the proofs and the fact that any remaining public components of the signature the user sees are independent of the signer's key. Blindness follows from the properties required by Fischlin's generic construction plus the composable rerandomizability [6] of the GS proof system. Finally, unforgeability is reduced to the unforgeability of the underlying signature scheme, the soundness of the proofs, the binding property of the commitment scheme, and the collision-resistant property of the hash function.

Efficient Instantiation. In order to get an efficient construction, we will base our signing protocol on a variant of the blind signing protocol from [26] using the signature scheme from [27] which has a short public key and is capable of signing a pair of group elements and an integer. Thus, obtaining a partially blind signing protocol as illustrated in Figure 2. We note here that the blind signature in [26] deviates from Fischlin's generic construction [24] for blind signatures in that the final signature is on the message itself rather than its commitment and it requires proofs of knowledge in the signature request protocol.

To obtain a blind ring signature on the message $(M, \tilde{M}) \in \mathbb{G}_1 \times \mathbb{G}_2$, the user commits to the message using Pedersen commitment $C := M \cdot T^s$ for some random $s \leftarrow \mathbb{Z}_p$ and computes $S := G^s$ and $\tilde{S} := \tilde{G}^s$. He then sends the commitment C along with GS proofs of knowledge Ω to prove that: the commitment C is indeed to the message M and that the message and the randomness pairs are well-formed.

The signer first verifies the proofs and if they are valid, produces a signature $\sigma := (U', \tilde{U}', V, \tilde{V}, W)$ on the commitment C and the public integer $\mathcal{H}(\mathcal{R})$ (for some collision-resistant hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$) using the variant of the automorphic signature scheme [26] as in [27]. However, instead of sending the signature in the clear, the signer sends a GS proof of knowledge Ω'_{sig} of his public verification key (A, \tilde{A}) and the signature σ such that the signature verifies w.r.t. his key. Since the components U' and \tilde{U}' of σ are independent of the signing key, we need not hide them. Additionally, the signer generates a proof of membership Ω'_{mem} to prove that his key is in the ring. The signer's response is $(\Omega'_{\text{sig}}, \Omega'_{\text{mem}}, U', \tilde{U}')$.

The user first verifies the GS proofs Ω'_{sig} and Ω'_{mem} , and that the pair (U', \tilde{U}') is well-formed. If they are valid, the user rerandomizes those proofs into Ω_{sig} and Ω_{mem} , respectively, using the algorithm `GSRandomize`. The new proofs are unlinkable to the original ones. The user then transforms the proof Ω_{sig} by making the signature verify w.r.t. to the message itself rather than its commitment: he computes $U := U' \cdot S$ and $\tilde{U} := \tilde{U}' \cdot \tilde{S}$, and transforms the last equation in Ω_{sig} from $e(\underline{W}, \underline{\tilde{A}} \cdot \underline{\tilde{V}}) = e(K \cdot C \cdot L^{\mathcal{H}(\mathcal{R})}, \underline{\tilde{G}})e(T, \underline{\tilde{U}'})$ into $e(\underline{W}, \underline{\tilde{A}} \cdot \underline{\tilde{V}})e(T^{-1}, \underline{\tilde{U}}) = e(K \cdot M \cdot L^{\mathcal{H}(\mathcal{R})}, \underline{\tilde{G}})$. In addition, he hides the components (U, \tilde{U}) by adding a proof for the equation $e(\underline{U}, \underline{\tilde{G}}) = e(\underline{G}, \underline{\tilde{U}})$.

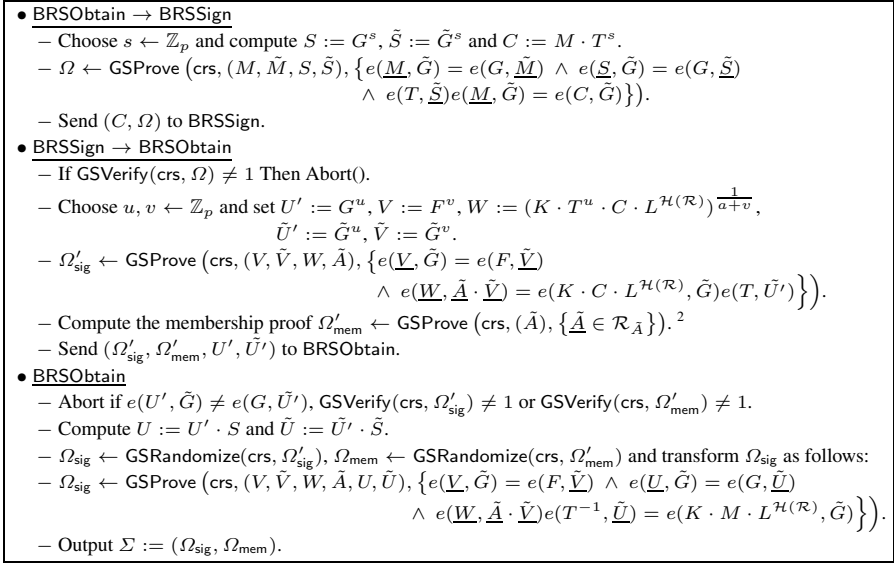


Fig. 3. The signing protocol

The final blind ring signature is $\Sigma := (\Omega_{\text{sig}}, \Omega_{\text{mem}})$. Again, the two proofs share the same GS commitment to the signer's verification key.

The detailed construction in the asymmetric setting is as follows:

- **BRSSSetup**(1^λ):
 - Run $\mathcal{P} \leftarrow \text{BGrpSetup}(1^\lambda, 3)$ and $(\text{crs}, \text{sk}) \leftarrow \text{GSSetup}(\mathcal{P})$. Parse \mathcal{P} as $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \tilde{G}, e)$.
 - Choose a suitable collision-resistant hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ and $F, K, L, T \leftarrow \mathbb{G}_1$.
 - Set $\text{param}_{\text{BRS}} := (\mathcal{P}, \text{crs}, \mathcal{H}, F, K, L, T)$. Return $\text{param}_{\text{BRS}}$.
- **BRSKeyGen**($\text{param}_{\text{BRS}}$):
 - Choose $a \leftarrow \mathbb{Z}_p$ and set $A := G^a$ and $\tilde{A} := \tilde{G}^a$. Set $\text{sk} := a$, $\text{pk} := (A, \tilde{A})$. Return (sk, pk) .
- The signing protocol $\langle \text{BRSSObtain}((M, \tilde{M}), \mathcal{R}), \text{BRSSign}(\text{sk}, \mathcal{R}) \rangle$ is in Figure 3.
- **BRSVerify**($((M, \tilde{M}), \Sigma, \mathcal{R})$):
 - Parse Σ as $(\Omega_{\text{sig}}, \Omega_{\text{mem}})$.
 - Return 1 if $\text{GSVerify}(\text{crs}, \Omega_{\text{mem}}) = 1$ and $\text{GSVerify}(\text{crs}, \Omega_{\text{sig}}) = 1$. Otherwise, return 0.

We provide a proof for the following Theorem in the full version [30].

Theorem 2. *The construction is a secure blind ring signature scheme.*

² We only prove membership for one component of the key. The verifier can verify that all keys in the ring are well-formed. Alternatively, one can add a proof for the equation $e(\underline{A}, \tilde{G}) = e(G, \tilde{A})$. It is a matter of trade-off between communication and computation complexities.

Table 2. Size of the blind ring signature in the different instantiations

Setting	Signature Size	Assumptions
Type-1	\mathbb{G}^{30n+42}	DLIN, q-ADHSDH and WFCDH
Type-2	$\mathbb{G}_1^{16n+22} + \mathbb{G}_2^{24n+30}$	DDH $_{\mathbb{G}_1}$, DLIN $_{\mathbb{G}_2}$, q-ADHSDH and AWFCDH
Type-3	$\mathbb{G}_1^{16n+22} + \mathbb{G}_2^{16n+20}$	SXDH, q-ADHSDH and AWFCDH

Efficiency of the Construction. As mentioned earlier, our construction is the first realization in the standard model and also the first to offer sub-linear signatures instead of linear ones. Table 2 summarizes the size of the signature as well as the required assumptions for the different instantiations. Type-1 instantiation uses the DLIN instantiation of GS proofs, Type-2 uses GS proofs based on DDH in \mathbb{G}_1 and DLIN in \mathbb{G}_2 as used in [32], and Type-3 uses the SXDH instantiation of the proofs.

To give example concrete figures, we consider a security level equivalent to 128-bit symmetric key security. For a ring consisting of 10,000 members, the Type-1 instantiation, where the size of elements of group \mathbb{G} is 512 bits, yields signatures of size of approximately 190 kB. At the same security level in the asymmetric setting where elements of \mathbb{G}_1 are of size 256 bits and those of \mathbb{G}_2 are of size 512 bits, the signature size is 203 kB and 152 kB in the Type-2 and Type-3 instantiations, respectively. Again, the verification of the signature can be made more efficient by batch verifying Groth-Sahai proofs [31,9].

6 Generic Construction of Ring Signatures over Prime-Order Bilinear Groups

Here we provide a generic construction for ring signatures without random oracles by combining the set membership proof from Section 4 with any compatible signature scheme.

Let $\text{Sig} := ([\text{SigSetup}], \text{SigKeyGen}, \text{SigSign}, \text{SigVerify})$ be an existentially unforgeable signature scheme secure against adaptive chosen-message attack that works in any of the 3 main types (cf. Section 2) of prime-order bilinear groups. Let \mathcal{M}_{Sig} be its message space, (sk, pk) be its key pair and $\sigma := (\sigma_1, \dots, \sigma_n)$ be its signatures for some positive integer n with the condition that for any $i \in [1, n]$, σ_i is a group element if it depends on sk .³ Our construction is as follows:

- $\text{RSSetup}(1^\lambda)$: Run $\mathcal{P} \leftarrow \text{BGrpSetup}(1^\lambda, \text{tp})$ for $\text{tp} \in [1, 3]$, $(\text{crs}, \text{xk}) \leftarrow \text{GSSetup}(\mathcal{P})$. Choose a collision-resistant hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \mathcal{M}_{\text{Sig}}$. The public parameters is then $\text{param}_{\text{RS}} := (\mathcal{P}, \text{crs}, \mathcal{H})$. Note that if Sig requires setup, then the output of SigSetup is also added to param_{RS} .
- $\text{RSKeyGen}(\text{param}_{\text{RS}})$: Run SigKeyGen to obtain (sk, pk) .
- $\text{RSSign}(\text{sk}_i, m, \mathcal{R})$: To sign a message $m \in \{0, 1\}^*$ w.r.t. a ring $\mathcal{R} := \{\text{pk}_1, \dots, \text{pk}_N\}$ where $\text{pk}_i \in \mathcal{R}$, run $\sigma \leftarrow \text{SigSign}(\text{sk}_i, \mathcal{H}(m, \mathcal{R}))$. Then generate the

³ Unlike structure-preserving signatures [2], we do not require that the messages are group elements.

Table 3. Example instantiations of the generic ring signatures construction

Instantiation	Setting	Signature Size	Complexity Assumptions
Waters	Type-1	\mathbb{G}^{30n+19}	CDH + DLIN
	Type-2	$\mathbb{G}_1^{16n+10} + \mathbb{G}_2^{24n+13}$	Co-CDH + DDH $_{\mathbb{G}_1}$ + DLIN $_{\mathbb{G}_2}$
	Type-3	$\mathbb{G}_1^{16n+11} + \mathbb{G}_2^{16n+9}$	Co-CDH* + SXDH
FBB	Type-1	$\mathbb{G}^{42n+39} + \mathbb{Z}_p^4$	q-SDH + DLIN
	Type-2	$\mathbb{G}_1^{20n+14} + \mathbb{G}_2^{30n+21} + \mathbb{Z}_p^4$	q-SDH + DDH $_{\mathbb{G}_1}$ + DLIN $_{\mathbb{G}_2}$
	Type-3	$\mathbb{G}_1^{20n+14} + \mathbb{G}_2^{20n+14} + \mathbb{Z}_p^3$	q-SDH + SXDH

following two Groth-Sahai proofs where $\bar{\sigma}$ is the subset of σ which depends on the secret key sk.

$$\Omega_{\text{sig}} \leftarrow \text{GSProve}\{\text{crs}, (\text{pk}_i, \bar{\sigma}), \{\text{SigVerify}(\underline{\text{pk}}_i, \mathcal{H}(m, \mathcal{R}), \sigma) = 1\}\},$$

$$\Omega_{\text{mem}} \leftarrow \text{GSProve}\{\text{crs}, (\text{pk}_i), \{\underline{\text{pk}}_i \in \mathcal{R}\}\}.$$

The ring signature is then $\Sigma := (\Omega_{\text{sig}}, \Omega_{\text{mem}}, \{\sigma\} \setminus \{\bar{\sigma}\})$. Again, the proofs Ω_{sig} and Ω_{mem} must be correlated, i.e. they involve the same public key pk_i . This is checked by ensuring that both proofs use the same GS commitment to pk_i when verifying the signature.

- RSV $\text{erify}(m, \Sigma, \mathcal{R})$: To verify that the ring signature Σ is a valid signature on the message m w.r.t. the ring \mathcal{R} , verify the two proofs Ω_{mem} and Ω_{sig} .

We provide a proof for the following Theorem in the full version [30].

Theorem 3. *The generic construction is a secure ring signature scheme for message space $\{0, 1\}^*$.*

In the full paper [30] we provide two example instantiations. In the first we instantiate the Sig scheme using the full Boneh-Boyen signature scheme [10], whereas in the second instantiation we use Waters signature scheme [49]. The efficiency summary of those instantiations is provided in Table 3.

6.1 Instantiating the Construction in [17] over Prime-Order Groups

We note that by combining our set membership proof with the weakly secure Boneh-Boyen signature scheme [10] and one-time signatures instantiated over prime-order groups, we get efficient instantiations in prime-order groups of the composite-order construction given in [17].

Acknowledgments. This work was supported by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO and EPSRC via grant EP/H043454/1.

References

1. Abdalla, M., Nampreppe, C., Neven, G.: On the (Im)possibility of blind message authentication codes. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 262–279. Springer, Heidelberg (2006)

2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
3. Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (1996)
4. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n Signatures from a Variety of Keys. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 415–432. Springer, Heidelberg (2002)
5. Abe, M., Okamoto, T.: Provably Secure Partially Blind Signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Heidelberg (2000)
6. 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)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security 1993, pp. 62–73. ACM (1993)
8. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006)
9. Blazy, O., Fuchsbauer, G., Izabachène, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch Groth-Sahai. Cryptology ePrint Archive, Report 2010/040, <http://eprint.iacr.org/2010/040>
10. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology* 21(2), 149–177 (2008)
11. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
12. 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)
13. 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)
14. Boyen, X.: Mesh Signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007)
15. Brakerski, Z., Kalai, Y.T.: A Framework for Efficient Signatures, Ring Signatures and Identity Based Encryption in the Standard Model. In: Cryptology ePrint Archive, Report 2010/086, <http://eprint.iacr.org/2010/086.pdf>
16. Chan, T.K., Fung, K., Liu, J.K., Wei, V.K.: Blind Spontaneous Anonymous Group Signatures for Ad Hoc Groups. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 82–94. Springer, Heidelberg (2005)
17. 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)
18. Chatterjee, S., Hankerson, D., Knapp, E., Menezes, A.: Comparing Two Pairing-Based Aggregate Signature Schemes. In: Cryptology ePrint Archive, Report 2009/060, <http://eprint.iacr.org/2009/060.pdf>
19. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO 1982, pp. 199–203. Plenum Press (1983)
20. Desmedt, Y.G., Goutier, C., Bengio, S.: Special Uses and Abuses of the Fiat Shamir Passport Protocol. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 21–39. Springer, Heidelberg (1988)

21. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Nicolosi and Victor Shoup. Anonymous Identification in Ad Hoc Groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
22. Dwork, C., Naor, M.: Zaps and their applications. In: FOCS 2000, pp. 283–293 (1999)
23. 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)
24. Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
25. Freeman, D.M.: Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
26. Fuchsbauer, G.: Automorphic Signatures in Bilinear Groups and an Application to Round-Optimal Blind Signatures. In: Cryptology ePrint Archive, Report 2009/320, <http://eprint.iacr.org/2009/320.pdf>
27. Fuchsbauer, G.: Commuting signatures and verifiable encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 224–245. Springer, Heidelberg (2011)
28. Galbraith, S., Paterson, K., Smart, N.P.: Pairings for cryptographers. *Discrete Applied Mathematics* 156, 3113–3121 (2008)
29. Ghadafi, E.: Formalizing group blind signatures and practical constructions without random oracles. In: Boyd, C., Simpson, L. (eds.) ACISP. LNCS, vol. 7959, pp. 330–346. Springer, Heidelberg (2013)
30. Ghadafi, E.: Sub-linear Blind Ring Signatures without Random Oracles. In: Cryptology ePrint Archive, Report 2013/612, <http://eprint.iacr.org/2013/612.pdf>
31. Ghadafi, E., Smart, N.P., Warinschi, B.: Practical zero-knowledge proofs for circuit evaluation. In: Parker, M.G. (ed.) Cryptography and Coding 2009. LNCS, vol. 5921, pp. 469–494. Springer, Heidelberg (2009)
32. Ghadafi, E., Smart, N.P., Warinschi, B.: Groth-Sahai proofs revisited. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 177–192. Springer, Heidelberg (2010)
33. 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)
34. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups (full version), <http://www.brics.dk/~jg/WImoduleFull.pdf>
35. Herranz, J., Laguillaumie, F.: Blind Ring Signatures Secure Under the Chosen-Target-CDH Assumption. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 117–130. Springer, Heidelberg (2006)
36. Herranz, J., Sáez, G.: Forking Lemmas for Ring Signature Schemes. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 266–279. Springer, Heidelberg (2003)
37. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997)
38. 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)
39. Naor, M.: Deniable Ring Authentication. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 481–498. Springer, Heidelberg (2002)
40. Nguyen, L.: Accumulators from Bilinear Pairings and Applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)

41. Nguyen, K.Q., Mu, Y., Varadharajan, V.: Divertible Zero-Knowledge Proof of Polynomial Relations and Blind Group Signature. In: Pieprzyk, J.P., Safavi-Naini, R., Seberry, J. (eds.) ACISP 1999. LNCS, vol. 1587, pp. 117–128. Springer, Heidelberg (1999)
42. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
43. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. *Journal of Cryptology* 13(3), 361–396 (2000)
44. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
45. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
46. Schäge, S., Schwenk, J.: A CDH-Based Ring Signature Scheme with Short Signatures and Public Keys. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 129–142. Springer, Heidelberg (2010)
47. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 166–180. Springer, Heidelberg (2007)
48. 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)
49. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
50. Wu, Q., Zhang, F., Susilo, W., Mu, Y.: An Efficient Static Blind Ring Signature Scheme. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 410–423. Springer, Heidelberg (2006)
51. Zhang, J., Chen, H., Liu, X., Liu, C.: An Efficient Blind Ring Signature Scheme without Pairings. In: Shen, H.T., Pei, J., Özsü, M.T., Zou, L., Lu, J., Ling, T.-W., Yu, G., Zhuang, Y., Shao, J. (eds.) WAIM 2010. LNCS, vol. 6185, pp. 177–188. Springer, Heidelberg (2010)

Constructions of Signcryption in the Multi-user Setting from Identity-Based Encryption

Rintaro Nakano and Junji Shikata

Graduate School of Environment and Information Sciences,
Yokohama National University, Japan
{nakano-rintaro-xf, shikata}@ynu.ac.jp

Abstract. The security of signcryption requires both confidentiality and integrity, and various constructions of signcryption have been proposed so far. Although insider security in the multi-user setting is desirable in signcryption, in the standard model (i.e., without random oracles) there are only few generic constructions of signcryption which meet both strong insider indistinguishability and strong insider unforgeability in the multi-user setting. In this paper, we propose two generic constructions of signcryption schemes in the standard model, and our constructions achieve such strong security. Our generic constructions are simple and quite different from the existing constructions of signcryption, and ours are based on the elegant known techniques for constructing strongly secure public-key encryption from identity-based encryption.

1 Introduction

1.1 Background

The notion of signcryption was introduced by Zheng [16] in 1997. The purpose of signcryption is to achieve security of encryption and digital signatures simultaneously, i.e., confidentiality and integrity. Various constructions of signcryption schemes have been proposed so far [1,2,9,11,12,14]. The early security model of signcryption schemes considers the two-user setting that consists only of a single sender and a single receiver. However, security in the two-user setting does not imply security in the multi-user setting that consists of multiple senders and receivers. Thus, it is important to consider security of signcryption schemes in the multi-user setting. In addition, signcryption schemes have two kinds of security definitions, namely an outsider security and an insider security. In the outsider security, an external adversary only knows public information (i.e., public parameters and public-keys of entities). On the other hand, in the insider security, an internal adversary can know some private-keys. Note that the insider security is stronger, and hence it is sufficient and reasonable to consider the insider security. In 2004, the strongest security definition, which consists of strong insider confidentiality and strong insider integrity in the multi-user setting, was first formalized by Libert et al. [11].

Currently, several constructions of signcryption schemes that achieve the strongest security in the random oracle model are known (for example, see

[2,11,12]). However, there are only few constructions of signcryption schemes proposed in the standard model (i.e., without random oracles) [9,12,14]. In [14], Tan proposed a direct construction of strongest signcryption schemes, while generic constructions of strongest signcryption schemes were proposed by Matsuda et al. in [12] and Chiba et al. in [9]. The generic construction is useful in the sense that various primitives under various computational assumptions can be flexibly combined in a general setting of parameters as the need of applications depending on the situation. However, in [12] (and [14]), a restricted model called a *key registration* (KR) model is assumed. In the KR model, users are required to obtain a certificate by registering their public-keys at a certificate authority before the public-keys are used in interaction with other users. And, it is desirable to construct signcryption schemes without KR. Therefore, in [9] generic constructions of signcryption schemes without KR were proposed for the first time. In this paper, we also propose generic constructions of strongest signcryption schemes without KR in the standard model (see the next subsection for details).

1.2 Our Contribution

Our main purpose is to propose a simple approach to constructing strongly secure signcryption in the multi-user setting by extending the elegant techniques in [4,5,8] for constructing strongly secure PKE (public-key encryption) from IBE (identity-based encryption). In this paper, we propose two generic constructions of signcryption schemes without random oracles, so the resulting constructions of signcryption do not require random oracles if concrete instantiations without random oracles are applied. Moreover, our constructions achieve the strongest security, i.e., both of multi-user indistinguishability against insider chosen ciphertext attack (MU-IND-iCCA) and multi-user strong unforgeability against insider chosen message attack (MU-sUF-iCMA) without KR. Our idea for two generic constructions lies in the following technical points.

- The first construction is based on the technique in [4,8] for constructing PKE from IBE and OTS (one-time signature), and the technique in [3] for constructing DS (digital signature) from (weak) DS and OTS. Our idea for constructing signcryption is to combine primitives IBE, (weak) DS, and OTS based on the techniques in [4,8,3], and to optimize it such that OTS will be used only once (see Section 4.1).
- The second construction is based on the technique in [4,5] for constructing PKE from IBE, MAC and encapsulation. In addition to IBE, we note that MAC and encapsulation are effectively utilized in [4,5], while OTS is used in [4,8]. In addition to this idea, we consider some technique to achieve strong unforgeability of signcryption schemes by using MAC and encapsulation in addition to (weak) DS. This idea successfully leads to the second construction for signcryption schemes such that each of IBE, (weak) DS, MAC, and encapsulation is used only once (see Section 4.2).

Our generic constructions achieve the strongest security (i.e., MU-IND-iCCA security and MU-sUF-iCMA security) without KR as well as the ones in [9], however, our techniques are quite different from the ones in [9].

Furthermore, if we apply reasonable instantiations of cryptographic primitives to our generic constructions, it turns out that our constructions achieve the strongest security under the same computational assumptions as the ones in [9] (i.e., MU-IND-iCCA security under the DBDH assumption, and MU-sUF-iCMA security under the CDH assumption), and our construction has the same or shorter size of public/secret-keys and less computational cost compared to other existing constructions, however, our construction has larger ciphertext-size. In this sense, there is a tradeoff between parameters of efficiency. Therefore, it would be possible to adapt a construction of signcryption effectively and properly to suit the situation of applications.

The rest of this paper is organized as follows. In Section 2, we review several cryptographic primitives, such as IBE, DS, MAC, and encapsulation schemes, which are used for our constructions of signcryption schemes as building blocks. In Section 3, we describe the model and the security definition of signcryption schemes. In Section 4, we propose two generic constructions of signcryption schemes without random oracles. Section 5 is devoted to compare our constructions with other known constructions. Finally, in Section 6, we conclude the paper.

2 Preliminaries

In this paper, we use the following notation. If we write $(y_1, y_2, \dots, y_m) \leftarrow A(x_1, x_2, \dots, x_n)$ for an algorithm A having n inputs and m outputs, it means to input x_1, x_2, \dots, x_n into A and to get the resulting output y_1, y_2, \dots, y_m . If A is a probabilistic algorithm, we write $(y_1, y_2, \dots, y_m) \leftarrow A(r; x_1, x_2, \dots, x_n)$, where r is a random value used in A . If x is a string, then $|x|$ denotes its bit-length. We denote a concatenation of x and y by $x||y$. If we write negligible ϵ in k , it means a function $\epsilon : \mathbb{N} \rightarrow [0, 1]$ where $\epsilon(k) < 1/g(k)$ for any polynomial g and sufficiently large k . Furthermore, in this paper, a polynomial-time algorithm is abbreviated as PTA.

2.1 Identity-Based Encryption (IBE)

An id-based encryption (IBE) consists of a five-tuple of PTAs as follows, where a space of ids $\mathcal{ID}_{\text{IBE}}$ and a space of plaintexts \mathcal{M}_{IBE} are determined by a security parameter k .

- $prm \leftarrow \text{IBE.Setup}(1^k)$: A setup algorithm is probabilistic, and it takes a security parameter k as input and then outputs a public parameter prm .
- $(mpk, msk) \leftarrow \text{IBE.Kg}(prm)$: A key generation algorithm is probabilistic, and it takes a public parameter prm as input and then outputs a master public-key mpk and a master secret-key msk .

- $SK_{ID} \leftarrow \text{IBE.Der}(msk, ID)$: A key derivation algorithm is probabilistic, and it takes a master secret-key msk and an id ID as input and then outputs a secret-key SK_{ID} for ID .
- $C \leftarrow \text{IBE.Enc}(mpk, ID, M)$: An encryption algorithm is probabilistic, and it takes a master public-key mpk , an id ID and a plaintext $M \in \mathcal{M}_{\text{IBE}}$ as input and then outputs a ciphertext C .
- M or $\perp \leftarrow \text{IBE.Dec}(SK_{ID}, ID, C)$: A decryption algorithm is deterministic algorithm, and it takes a secret-key sk_{ID} for ID , an id ID , and a ciphertext C as input and then outputs a plaintext $M \in \mathcal{M}_{\text{IBE}}$ or an invalid symbol $\perp \notin \mathcal{M}_{\text{IBE}}$.

We require that for all $k \in \mathbb{N}$, all $prm \leftarrow \text{IBE.Setup}(1^k)$, all $(mpk, msk) \leftarrow \text{IBE.Kg}(prm)$, all $ID \in \mathcal{ID}_{\text{IBE}}$, all $SK_{ID} \leftarrow \text{IBE.Der}(msk, ID)$, all $M \in \mathcal{M}_{\text{IBE}}$, it holds that $M = \text{IBE.Dec}(SK_{ID}, ID, C)$, where $C \leftarrow \text{IBE.Enc}(mpk, ID, M)$.

Definition 1 (IND-sID-CPA). For IBEs, a notion of indistinguishability against selective id chosen plaintext attack (IND-sID-CPA) is defined as follows. Let $A = \{A_1, A_2, A_3\}$ be a polynomial-time adversary against $\text{IBE} = \{\text{IBE.Setup}, \text{IBE.Kg}, \text{IBE.Der}, \text{IBE.Enc}, \text{IBE.Dec}\}$, and we consider the following game:

- Step 1. $(ID^*, st_1) \leftarrow A_1(prm)$.
- Step 2. $(mpk, msk) \leftarrow \text{IBE.Kg}(prm)$.
- Step 3. $(M_0, M_1, st_2) \leftarrow A_2^{O(\cdot)}(st_1, mpk)$.
- Step 4. $b \leftarrow_R \{0, 1\}$, $C^* \leftarrow \text{IBE.Enc}(mpk, ID^*, M_b)$.
- Step 5. $b' \leftarrow A_3^{O(\cdot)}(st_2, C^*)$.

We require $|M_0| = |M_1|$, and O is an oracle which takes an id ID as input and then returns $\text{IBE.Der}(msk, ID)$. A is allowed to access the above oracle at any time, however, it cannot submit the target identity ID^* to O . We define the advantage of A in the above game by

$$\text{Adv}_{\text{IBE}, A}^{\text{IND-sID-CPA}}(k) := \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

An IBE meets ϵ -IND-sID-CPA, if there exists a negligible ϵ in k such that $\text{Adv}_{\text{IBE}, A}^{\text{IND-sID-CPA}}(k) \leq \epsilon$ holds for any probabilistic PTA A .

2.2 Digital Signatures (DS)

A digital signature (DS) consists of a four-tuple of PTAs as follows, where a space of messages \mathcal{M}_{SIG} is determined by a security parameter k .

- $prm \leftarrow \text{DS.Setup}(1^k)$: A setup algorithm is probabilistic, and it takes a security parameter k as input and then outputs a public parameter prm .
- $(VK, SK) \leftarrow \text{DS.Kg}(prm)$: A key generation algorithm is probabilistic, and it takes a public parameter prm as input and then outputs a verification-key VK and a signing-key SK .

- $S \leftarrow \text{DS.Sign}(SK, M)$: A signing algorithm is probabilistic, and it takes a signing-key SK and a message $M \in \mathcal{M}_{SIG}$ as input and then outputs a signature S .
- $1 \text{ or } 0 \leftarrow \text{DS.Vrfy}(VK, M, S)$: A verification algorithm is deterministic, and it takes a verification-key VK , a message $M \in \mathcal{M}_{SIG}$, and a signature S as input and then outputs 1 (accept) or 0 (reject).

We require that for all $k \in \mathbb{N}$, all $prm \leftarrow \text{DS.Setup}(1^k)$, all $(VK, SK) \leftarrow \text{DS.Kg}(prm)$, all $M \in \mathcal{M}_{SIG}$, it holds that $1 = \text{SIG.Vrfy}(VK, M, S)$, where $S \leftarrow \text{DS.Sign}(SK, M)$.

Definition 2 ((s)UF-CMA, (s)UF-OT). For DSs, notions of strong unforgeability against chosen message attack (sUF-CMA) and strong unforgeability against one-time attack (sUF-OT) are defined as follows. Let A be a polynomial-time adversary against a digital signature $DS = \{\text{DS.Setup}, \text{DS.Kg}, \text{DS.Sign}, \text{DS.Vrfy}\}$, and we consider the following game:

- Step 1. $prm \leftarrow \text{DS.Setup}(1^k)$, $(VK, SK) \leftarrow \text{DS.Kg}(prm)$.
- Step 2. $(M^*, S^*) \leftarrow A^{O(\cdot)}(VK)$.

O is an oracle which takes a message M as input and then returns $S \leftarrow \text{DS.Sign}(SK, M)$. A is allowed to access the above oracle, and we require $(M^*, S^*) \neq (M, S)$. Let $[A \text{ wins}]$ be an event of $1 = \text{DS.Vrfy}(VK, M^*, S^*)$. We define the advantage of A in the above game as follows: for $ATK \in \{CMA, OT\}$,

$$\text{Adv}_{\text{DS}, A}^{\text{sUF-}ATK}(k) := \Pr[A \text{ wins}],$$

where A is allowed to access the oracle O at most polynomial time if $ATK = CMA$, and A is allowed to access to O at most one time if $ATK = OT$. A DS meets ϵ -sUF-CMA (resp., ϵ -sUF-OT), if there exists a negligible ϵ in k such that $\text{Adv}_{\text{DS}, A}^{\text{sUF-CMA}}(k) \leq \epsilon$ (resp., $\text{Adv}_{\text{DS}, A}^{\text{sUF-OT}}(k) \leq \epsilon$) holds for any probabilistic PTA A . For a DS satisfying ϵ -sUF-OT, we write $OTS = \{\text{OTS.Setup}, \text{OTS.Kg}, \text{OTS.Sign}, \text{OTS.Vrfy}\}$.

Similarly, notions of unforgeability against chosen message attack (UF-CMA) and unforgeability against one-time attack (UF-OT) are defined by replacing the above event $[A \text{ wins}]$ with the event of $1 = \text{DS.Vrfy}(VK, M^*, S^*)$ such that $M^* \neq M$.

2.3 Message Authentication Code (MAC)

A message authentication code (MAC) consists of a two-tuple of PTAs as follows, where a space of keys \mathcal{K}_{MAC} is determined by a security parameter k .

- $\tau \leftarrow \text{MAC.Sign}(K, M)$: A message authentication algorithm is deterministic, and it takes a symmetric-key $K \in \mathcal{K}_{MAC}$ and a message $M \in \{0, 1\}^*$ as input and then outputs a tag τ .
- $1 \text{ or } 0 \leftarrow \text{MAC.Vrfy}(K, M, \tau)$: A verification algorithm is deterministic, and it takes a symmetric-key $K \in \mathcal{K}_{MAC}$, a message $M \in \{0, 1\}^*$, and a tag τ as input and then outputs 1 (accept) or 0 (reject).

We require that for all $k \in \mathbb{N}$, all $K \in \mathcal{K}_{\text{MAC}}$, all M , it holds that $1 = \text{MAC.Vrfy}(K, M, \tau)$, where $\tau \leftarrow \text{MAC.Sign}(K, M)$.

Definition 3 (sUF-OT). For MACs, a notion of strong unforgeability against one-time attack (sUF-OT) is defined as follows. Let A be a polynomial-time adversary against $\text{MAC} = \{\text{MAC.Sign}, \text{MAC.Vrfy}\}$, and we consider the following game:

- Step 1. $K \leftarrow_R \mathcal{K}_{\text{MAC}}$.
- Step 2. $(M^*, \tau^*) \leftarrow A^{O(\cdot)}(1^k)$.

O is an oracle which takes a plaintext M as input and then returns $\tau \leftarrow \text{MAC.Sign}(K, M)$. A is allowed to access the above oracle at most one time. We require $(M^*, \tau^*) \neq (M, \tau)$. Let $[A \text{ wins}]$ be an event of $1 = \text{MAC.Vrfy}(K, M^*, \tau^*)$. We define the advantage of A in the above game by $\text{Adv}_{\text{MAC}, A}^{\text{sUF-OT}}(k) := \Pr[A \text{ wins}]$.

A MAC meets ϵ -sUF-OT, if there exists a negligible ϵ in k such that $\text{Adv}_{\text{MAC}, A}^{\text{sUF-OT}}(k) \leq \epsilon$ holds for any probabilistic PTA A .

2.4 Encapsulation

The encapsulation was first defined by Boneh and Katz in [5], and it may be viewed as a weak variant of commitment.

An encapsulation consists of a three-tuple of PTAs as follows, where a space of random strings $\mathcal{R}_{\text{encap}}$ is determined by a security parameter k .

- $prm \leftarrow \text{E.Setup}(1^k)$: A setup algorithm is probabilistic, and it takes a security parameter k as input and then outputs a public parameter prm .
- $(r, com, dec) \leftarrow \text{E.Enc}(prm)$: An encapsulation algorithm is probabilistic, and it takes a public parameter prm as input and then outputs a random string $r \in \mathcal{R}_{\text{encap}}$, a public commitment com and a de-commitment dec .
- r or $\perp \leftarrow \text{E.Rec}(prm, com, dec)$: A recovering algorithm takes as input a public parameter prm , a public commitment com , and a de-commitment dec . It outputs a string $r \in \mathcal{R}_{\text{encap}}$ or an invalid symbol $\perp \notin \mathcal{R}_{\text{encap}}$.

We require that for all $k \in \mathbb{N}$, all $prm \leftarrow \text{E.Setup}(1^k)$, all $(r, com, dec) \leftarrow \text{E.Enc}(prm)$, it holds that $\text{E.Rec}(prm, com, dec) = r$.

Definition 4 (Security). For encapsulation schemes, security notions of hiding and binding are defined as follows. Let $A = \{A_1, A_2\}$ be a polynomial-time adversary against $\text{Encap} = \{\text{E.Setup}, \text{E.Enc}, \text{E.Rec}\}$.

Hiding. We consider the following game:

- Step 1. $prm \leftarrow \text{E.Setup}(1^k)$, $r_0 \leftarrow_R \mathcal{R}_{\text{encap}}$, $(r_1, com, dec) \leftarrow \text{E.Enc}(prm)$, and $b \leftarrow_R \{0, 1\}$.
- Step 2. $b' \leftarrow A_1(prm, com, r_b)$.

We define the advantage of A_1 in the above game by

$$\text{Adv}_{\text{Encap}, A_1}^{\text{hiding}}(k) := \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

Binding. We consider the following game:

- Step 1. $prm \leftarrow \text{E.Setup}(1^k)$, $(r, com, dec) \leftarrow \text{E.Enc}(prm)$.
 Step 2. $dec^* \leftarrow A_2(prm, com, dec)$.

Let $[A_2 \text{ wins}]$ be an event of $\text{E.Rec}(prm, com, dec^*) \notin \{\perp, r\}$. We define the advantage of A_2 in the above game by

$$Adv_{\text{Encap}, A_2}^{\text{binding}}(k) := \Pr[A_2 \text{ wins}].$$

An Encap is said to be ϵ -secure, if there exists a negligible ϵ in k such that $Adv_{\text{Encap}, A_1}^{\text{hiding}}(k) \leq \epsilon$ and $Adv_{\text{Encap}, A_2}^{\text{binding}}(k) \leq \epsilon$ hold for any probabilistic PTA $A = \{A_1, A_2\}$.

3 Signcryption

In this section, we describe the model and security definition of signcryption schemes.

A signcryption scheme (SCS) consists of a five-tuple of PTAs as follows, where a space of plaintexts \mathcal{M}_{SCS} are determined by a security parameter k .

- $prm \leftarrow \text{Setup}(1^k)$: A setup algorithm is probabilistic, and it takes a security parameter k as input and then outputs a public parameter prm .
- $(pk_R, sk_R) \leftarrow \text{KeyGen}_R(prm)$: A receiver's key generation algorithm is probabilistic, and it takes a public parameter prm as input and then outputs a receiver's public-key pk_R and a receiver's secret-key sk_R .
- $(pk_S, sk_S) \leftarrow \text{KeyGen}_S(prm)$: A sender's key generation algorithm is probabilistic, and it takes a public parameter prm as input and then outputs a sender's public-key pk_S and a sender's secret-key sk_S .
- $\sigma \leftarrow \text{SC}(prm, pk_R, sk_S, M)$: A signcrypt algorithm is probabilistic, and it takes a public parameter prm , a receiver's public-key pk_R , a sender's secret key sk_S and a plaintext $M \in \mathcal{M}_{\text{SCS}}$ as input and then outputs a ciphertext σ .
- $M \text{ or } \perp \leftarrow \text{USC}(prm, pk_S, sk_R, \sigma)$: An unsigncrypt algorithm takes a public parameter prm , a sender's public-key pk_S , a receiver's secret-key sk_R and a ciphertext σ as input and then outputs a plaintext $M \in \mathcal{M}_{\text{SCS}}$ or an invalid symbol $\perp \notin \mathcal{M}_{\text{SCS}}$.

We require that for all $k \in \mathbb{N}$, all $prm \leftarrow \text{Setup}(1^k)$, all $(pk_R, sk_R) \leftarrow \text{KeyGen}_R(prm)$, all $(pk_S, sk_S) \leftarrow \text{KeyGen}_S(prm)$, all $M \in \mathcal{M}_{\text{SCS}}$, it holds that

$$\text{USC}(prm, pk_S, sk_R, \text{SC}(prm, pk_R, sk_S, M)) = M.$$

The security of signcryption schemes consists of confidentiality and integrity. In this paper, we adopt the strongest security considered in [9,12,14]. Specifically, we consider the *multi-user indistinguishability against insider chosen ciphertext attack* (MU-IND-iCCA, for short) and the *multi-user strong unforgeability against insider chosen message attack* (MU-sUF-iCMA, for short). These notions are formalized as follows.

Definition 5 (MU-IND-iCCA). For SCSs, a notion of multi-user indistinguishability against insider chosen ciphertext attack (MU-IND-iCCA) is defined as follows. Let $A = \{A_1, A_2\}$ be a polynomial-time adversary against a signcryption scheme $SCS = \{\text{Setup}, \text{KeyGen}_R, \text{KeyGen}_S, \text{SC}, \text{USC}\}$, and we consider the following game:

- Step 1. $\text{prm} \leftarrow \text{Setup}(1^k)$, $(pk_R, sk_R) \leftarrow \text{KeyGen}_R(\text{prm})$.
- Step 2. $(M_0, M_1, pk_S^*, sk_S^*, st) \leftarrow A_1^{O(\cdot)}(\text{prm}, pk_R)$.
- Step 3. $b \leftarrow_R \{0, 1\}$, $\sigma^* \leftarrow \text{SC}(\text{prm}, pk_R, sk_S^*, M_b)$.
- Step 4. $b' \leftarrow A_2^{O(\cdot)}(st, \sigma^*)$.

We require $|M_0| = |M_1|$, and $O(\cdot)$ is an unencrypt oracle which takes (pk_S, σ) as input and then returns $\text{USC}(\text{prm}, pk_S, sk_R, \sigma)$. A is allowed to access the above oracle at any time, however, it cannot submit (pk_S^*, σ^*) to O in Step 4. We define the advantage of A in the above game by

$$\text{Adv}_{\text{SCS}, A}^{\text{MU-IND-iCCA}}(k) := \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

A SCS meets ϵ -MU-IND-iCCA, if there exists a negligible ϵ in k such that $\text{Adv}_{\text{SCS}, A}^{\text{MU-IND-iCCA}}(k) \leq \epsilon$ holds for any probabilistic PTA A .

Definition 6 (MU-sUF-iCMA). For SCSs, a notion of multi-user strong unforgeability against insider chosen message attack (MU-sUF-iCMA) is defined as follows. Let A be a polynomial-time adversary against a signcryption scheme $SCS = \{\text{Setup}, \text{KeyGen}_R, \text{KeyGen}_S, \text{SC}, \text{USC}\}$, and we consider the following game:

- Step 1. $\text{prm} \leftarrow \text{Setup}(1^k)$, $(pk_S, sk_S) \leftarrow \text{KeyGen}_S(\text{prm})$.
- Step 2. $(pk_R^*, sk_R^*, \sigma^*) \leftarrow A^{O(\cdot)}(\text{prm}, pk_S)$.

Here, O is a signcrypt oracle which takes (pk_R, M) as input and then returns $\text{SC}(\text{prm}, pk_R, sk_S, M)$. A is allowed to access the above oracle at most polynomial time, and suppose that $\{(pk_{R,1}, M_1, \sigma_1), (pk_{R,2}, M_2, \sigma_2), \dots, (pk_{R,t}, M_t, \sigma_t)\}$ is a set of queries and answers obtained by accessing O . Let $[A \text{ wins}]$ be an event that

$$\text{USC}(\text{prm}, pk_S, sk_R^*, \sigma^*) = M^* \wedge (pk_R^*, M^*, \sigma^*) \neq (pk_{R,i}, M_i, \sigma_i) \text{ for } 1 \leq i \leq t.$$

We define the advantage of A in the above game by

$$\text{Adv}_{\text{SCS}, A}^{\text{MU-sUF-iCMA}}(k) := \Pr[A \text{ wins}].$$

A SCS meets ϵ -MU-sUF-iCMA, if there exists a negligible ϵ in k such that $\text{Adv}_{\text{SCS}, A}^{\text{MU-sUF-iCMA}}(k) \leq \epsilon$ holds for any probabilistic PTA A .

4 Our Constructions

In this section, we propose two generic constructions of signcryption schemes.

4.1 Construction 1

Although it is natural to think that PKE (public-key encryption) and DS can be used to construct signcryption schemes, it is known that security for both MU-IND-iCCA and MU-sUF-iCMA cannot be achieved if we combine PKE and DS in a trivial way (see [1]). Therefore, we need some technical idea to achieve security for both MU-IND-iCCA and MU-sUF-iCMA.

First, we note that, in [8], Canetti et al. constructed IND-CCA secure PKE from IND-sID-CPA secure IBE and sUF-OT secure OTS in a generic way (i.e., generic construction). Secondly, we also note that, in [3], Bellare and Shoup constructed sUF-CMA secure DS from UF-CMA secure DS and sUF-OT secure OTS in a generic way. Our idea for constructing signcryption is to combine primitives PKE, DS, and OTS based on their ideas in [8,3], and to optimize it such that OTS will be used only once, since OTS would be used twice without optimizing it (one for IND-CCA secure PKE and the other for sUF-CMA secure DS). Based on this idea, we propose the following generic construction.

Let $IBE = \{IBE.Setup, IBE.Kg, IBE.Der, IBE.Enc, IBE.Dec\}$ be an id-based encryption, let $DS = \{DS.Setup, DS.Kg, DS.Sign, DS.Vrfy\}$ be a digital signature, and let $OTS = \{OTS.Setup, OTS.Kg, OTS.Sign, OTS.Vrfy\}$ be a one-time signature. Then, a signcryption scheme $SCS = \{Setup, KeyGen_R, KeyGen_S, SC, USC\}$ is constructed as follows.

- $prm \leftarrow Setup(1^k)$: It computes $prm_{IBE} \leftarrow IBE.Setup(1^k)$, $prm_{DS} \leftarrow DS.Setup(1^k)$, and $prm_{OTS} \leftarrow OTS.Setup(1^k)$. Then, it outputs $prm = (prm_{IBE}, prm_{DS}, prm_{OTS})$.
- $(pk_R, sk_R) \leftarrow KeyGen_R(prm)$: It computes $(mpk, msk) \leftarrow IBE.Kg(prm_{IBE})$. Then, it outputs $pk_R = mpkCsk_R = mskD$
- $(pk_S, sk_S) \leftarrow KeyGen_S(prm)$: It computes $(VK, SK) \leftarrow DS.Kg(prm_{DS})$. Then, it outputs $pk_S = VKCsk_S = SKD$
- $\sigma \leftarrow SC(prm, pk_R, sk_S, M)$: It computes $(vk, sk) \leftarrow OTS.Kg(prm_{OTS})$, $S \leftarrow DS.Sign(sk_S, M || vk || pk_R)$, $C \leftarrow IBE.Enc(pk_R, vk, M || S)$, $s \leftarrow OTS.Sign(sk, C)$. Then, it outputs $\sigma = (vk, C, s)D$
- M or $\perp \leftarrow USC(prm, pk_S, sk_R, \sigma)$: If $OTS.Vrfy(vk, C, s) = 1$, it computes $SK_{vk} \leftarrow IBE.Der(sk_R, vk)$ and $M || S \leftarrow IBE.Dec(SK_{vk}, vk, C)$. And, if $DS.Vrfy(pk_S, M || vk || pk_R, S) = 1$, it outputs MD Otherwise, it outputs \perp .

We can show that the resulting signcryption SCS in the above construction has security of both MU-IND-iCCA and MU-sUF-iCMA, if given IBE is IND-sID-CPA secure, OTS is sUF-OT secure, and DS is UF-CMA secure, as follows. The proofs of Theorems 1 and 2 are given in Appendices A and B, respectively.

Theorem 1. *If given IBE meets ϵ_1 -IND-sID-CPA and OTS meets ϵ_2 -sUF-OT, then the resulting SCS in the above construction meets δ -MU-IND-iCCA with $\delta \leq \epsilon_1 + \frac{1}{2}q\epsilon_2$, where q is the number of queries to the unsigncrypt oracle.*

Theorem 2. *If given DS meets ϵ_1 -UF-CMA and OTS meets ϵ_2 -sUF-OT, then the resulting SCS in the above construction meets δ -MU-sUF-iCMA with $\delta \leq \epsilon_1 + q\epsilon_2$, where q is the number of queries to the unsigncrypt oracle.*

4.2 Construction 2

In [5], Boneh et al. constructed IND-CCA secure PKE from IND-sID-CPA secure IBE, sUF-OT secure MAC and secure encapsulation in a generic way. Note that MAC and encapsulation are effectively utilized in [5], while OTS is used in [8]¹. In addition to this idea, we consider some technique to achieve strong unforgeability of signcryption schemes by using MAC and encapsulation in addition to DS. This idea successfully leads to the following construction for signcryption schemes such that each of IBE, DS, MAC, and encapsulation is used only once.

Let $\text{IBE} = \{\text{IBE.Setup}, \text{IBE.Kg}, \text{IBE.Der}, \text{IBE.Enc}, \text{IBE.Dec}\}$ be an id-based encryption, let $\text{DS} = \{\text{DS.Setup}, \text{DS.Kg}, \text{DS.Sign}, \text{DS.Vrfy}\}$ be a digital signature, and let $\text{MAC} = \{\text{MAC.Sign}, \text{MAC.Vrfy}\}$ and $\text{Encap} = \{\text{E.Setup}, \text{E.Enc}, \text{E.Rec}\}$ be a MAC and an encapsulation, respectively. Then, a signcryption scheme $\text{SCS} = \{\text{Setup}, \text{KeyGen}_R, \text{KeyGen}_S, \text{SC}, \text{USC}\}$ is constructed as follows.

- $\text{prm} \leftarrow \text{Setup}(1^k)$: It computes $\text{prm}_{\text{IBE}} \leftarrow \text{IBE.Setup}(1^k)$, $\text{prm}_{\text{DS}} \leftarrow \text{DS.Setup}(1^k)$, and $\text{prm}_E \leftarrow \text{E.Setup}(1^k)$. Then, it outputs $\text{prm} = (\text{prm}_{\text{IBE}}, \text{prm}_{\text{DS}}, \text{prm}_E)$.
- $(\text{pk}_R, \text{sk}_R) \leftarrow \text{KeyGen}_R(\text{prm})$: It computes $(\text{mpk}, \text{msk}) \leftarrow \text{IBE.Kg}(\text{prm}_{\text{IBE}})$. Then, it outputs $\text{pk}_R = \text{mpkCsk}_R = \text{mskD}$.
- $(\text{pk}_S, \text{sk}_S) \leftarrow \text{KeyGen}_S(\text{prm})$: It computes $(VK, SK) \leftarrow \text{DS.Kg}(\text{prm}_{\text{DS}})$. Then, it outputs $\text{pk}_S = VKCsk_S = SKD$.
- $\sigma \leftarrow \text{SC}(\text{prm}, \text{pk}_R, \text{sk}_S, M)$: It computes $(r, \text{com}, \text{dec}) \leftarrow \text{E.Enc}(\text{prm}_E)$, $S \leftarrow \text{DS.Sign}(\text{sk}_S, M \parallel \text{com} \parallel \text{pk}_R)$, $C \leftarrow \text{IBE.Enc}(\text{pk}_R, \text{com}, M \parallel S \parallel \text{dec})$, $\tau \leftarrow \text{MAC}(r, C)$. Then, it outputs $\sigma = (\text{com}, C, \tau)D$.
- M or $\perp \leftarrow \text{USC}(\text{prm}, \text{pk}_S, \text{sk}_R, \sigma)$: It computes $SK_{\text{com}} \leftarrow \text{IBE.Der}(\text{sk}_R, \text{com})$, $M \parallel S \parallel \text{dec} \leftarrow \text{IBE.Dec}(SK_{\text{com}}, \text{com}, C)$, and $r \leftarrow \text{E.Rec}(\text{prm}_E, \text{com}, \text{dec})$. And, if $\text{MAC.Vrfy}(r, C, \tau) = 1$ and $\text{DS.Vrfy}(\text{pk}_S, M \parallel \text{com} \parallel \text{pk}_R, S) = 1$, it outputs MD . Otherwise, it outputs \perp .

We can show that the resulting signcryption SCS in the above construction has security of both MU-IND-iCCA and MU-sUF-iCMA, if given IBE is IND-sID-CPA secure, DS is UF-CMA secure, MAC is sUF-OT secure, and encapsulation is secure, as follows. The proofs of Theorems 3 and 4 are given in Appendices C and D, respectively.

Theorem 3. *If given IBE meets ϵ_1 -IND-sID-CPA, MAC meets ϵ_2 -sUF-OT and encapsulation is ϵ_3 -secure, then the resulting SCS in the above construction meets δ -MU-IND-iCCA with $\delta \leq 4\epsilon_1 + q\epsilon_2 + 2\epsilon_3$, where q is the number of queries to the unsigncrypt oracle.*

Theorem 4. *If given DS meets ϵ_1 -UF-CMA, MAC meets ϵ_2 -sUF-OT and encapsulation is ϵ_3 -secure, then the resulting SCS in the above construction meets δ -MU-sUF-iCMA with $\delta \leq \epsilon_1 + q\epsilon_2 + 2\epsilon_3$, where q is the number of queries to the unsigncrypt oracle.*

¹ However, we do not know whether secure OTS can be constructed from MAC and encapsulation in a generic way.

5 Comparison

In this section, we compare our constructions with other existing ones of signcryption schemes.

5.1 Comparison of Generic Constructions

In this subsection, we compare generic constructions of signcryption schemes in the standard model (i.e., without random oracles). We summarize existing such constructions in Table 1. In Table 1, TBE is a *tag-based public-key encryption* in which the encryption and decryption algorithms take a tag as an additional input; TBKEM is a *tag-based key encapsulation mechanism*; and security of IND-tag-CCA means the strongest security of TBE and TBKEM (see [12] for details). In Table 1, MU-wUF-iCMA means *multi-user weak unforgeability against insider chosen message attack*, and it is weaker than MU-sUF-iCMA.

Table 1. Comparison of generic constructions of signcryptions in the standard model

Constructions	Primitives	Achievable security
MMS-StTE[12]	IND-tag-CCA secure TBE sUF-CMA secure DS	MU-IND-iCCA MU-wUF-iCMA
MMS-SC[12]	IND-tag-CCA secure TBKEM sUF-CMA secure DS	MU-IND-iCCA MU-sUF-iCMA(KR)
SC _{tk} [9]	IND-tag-CCA secure TBKEM sUF-CMA secure DS, IND-CCA secure DEM	MU-IND-iCCA MU-sUF-iCMA
SC _{kem} [9]	IND-CCA secure KEM, sUF-CMA secure DS IND-CCA secure DEM, sUF-OT secure MAC	MU-IND-iCCA MU-sUF-iCMA
Construction 1 (ours)	IND-sID-CPA secure IBE, UF-CMA secure DS sUF-OT secure OTS	MU-IND-iCCA MU-sUF-iCMA
Construction 2 (ours)	IND-sID-CPA secure IBE, UF-CMA secure DS secure encapsulation, sUF-OT secure MAC	MU-IND-iCCA MU-sUF-iCMA

First, we compare cryptographic primitives providing confidentiality in all constructions in Table 1. As shown in Table 1, each construction achieves security of MU-IND-iCCA of signcryption schemes. Note that IND-tag-CCA secure TBE, IND-tag-CCA secure TBKEM, or IND-CCA secure KEM/DEM is used as a primitive in MMS-StTE[12], MMS-SC[12], SC_{tk}[9], and SC_{kem}[9]. On the other hand, our constructions (i.e., Constructions 1 and 2) use IND-sID-CPA secure IBE. These constructions employ different kinds of primitives. However, it is known that IBE implies KEM [4,5,7,8] and also KEM implies TBE and TBKEM [10,12].

Secondly, we compare cryptographic primitives providing integrity in all constructions in Table 1. Note that MMS-StTE[12] cannot achieve security of MU-sUF-iCMA of signcryption (i.e., it only achieves MU-wUF-iCMA). MMS-SC[12] achieves MU-sUF-iCMA, however, it assumes the key registration (KR) model.

In the KR model, the adversary is required to reveal a secret-key corresponding to a public-key which is used in trying to attack signcryption schemes. This model assumes that the public-key infrastructure (PKI) is available and all users can execute zero-knowledge proofs with a certificate authority, which proves knowledge of their secret-keys before they obtain a certificate for their public-keys. However, issuing these proofs places a heavy duty on the certificate authority. From the above aspect, it is not desirable to assume the KR model, since this type of registration is not used in most practical systems. SC_{tk} [9], SC_{kem} [9] and our constructions achieve security of MU-sUF-iCMA without KR. SC_{tk} [9] uses sUF-CMA secure DS, while SC_{kem} [9] uses sUF-CMA secure DS and sUF-OT secure MAC. In contrast, our constructions achieve MU-sUF-iCMA by using UF-CMA secure DS and sUF-OT secure OTS, or UF-CMA secure DS and sUF-OT secure MAC and encapsulation. Note that, in DS, sUF-CMA security is stronger than UF-CMA security and sUF-OT security. However, sUF-CMA secure DS can be constructed in a generic way by combining UF-CMA secure DS and sUF-OT secure OTS [3]. Hence, in this sense, there is essentially no difference between the assumption about primitives. In summary, we observe that SC_{tk} , SC_{kem} and ours are better than MMS-StTE[12] and MMS-SC[12] in terms of integrity, since the constructions in [12] cannot achieve MU-sUF-iCMA without KR.

As a result of comparison, in confidentiality, it can be seen that our constructions need strong primitive, since our constructions require IBE. Furthermore, in integrity, each construction employs the same level of primitives, though our constructions use DS of weak security. However, we stress that our goal is not to displace the existing constructions [9] but rather to propose another approach to simply constructing strongly secure signcryption by extending well-known techniques for constructing strongly secure PKE from IBE [4,5,8].

5.2 Comparison of Direct Constructions

In this subsection, we compare direct constructions of signcryption schemes by applying reasonable instantiations of cryptographic primitives to generic constructions (see Appendix E for tables which summarize constructions of primitives used in this subsection), and we focus on comparison of SC_{tk} , SC_{kem} , and Constructions 1 and 2 (ours), since these four constructions achieve the strong security (i.e., MU-IND-iCCA and MU-sUF-iCMA) without KR.

In Table 2, we describe constructions of signcryption schemes about achievable security and applied concrete primitives and computational assumptions required for it. In confidentiality, we apply the primitives based on DBDH assumption to each construction, and in integrity, we apply the primitives based on CDH assumption to each construction, in order to fairly compare them based on the same computational assumptions.

Next, we compare computational cost and sizes of ciphertexts and public/secret-keys. Table 3 summarizes computational cost, and Table 4 summarizes sizes of ciphertexts and public/secret-keys. In particular, as observed in Table 3, Construction 2 (ours) is the best among the above four constructions in terms of computational cost. Furthermore, as we can see in Table 4, the sizes of

Table 2. Comparison of computational assumptions: for assumptions, DBDH means Decisional Bilinear Diffie-Hellman assumption, and CDH means Computational Diffie-Hellman assumption; for primitives, tBMW1 denotes the TBKEM in [12]. BMW2 denotes the KEM in [7]. BCHK denotes the IBE in [4]. BSW and Waters denote the DSs in [6] and [15], respectively. Okamoto denotes the OTS in [13]. Note that in the comparison BSW is applied to SC_{tk} and SC_{kem} , since it is more efficient than BS[3] using Waters and Okamoto (see Appendix E).

Construction	Confidentiality / Assumption	Integrity / Assumption
SC_{tk} [9] (tBMW1[12],BSW[6])	MU-IND-iCCA /DBDH	MU-sUF-iCMA /CDH
SC_{kem} [9] (BMW2[7],BSW[6])	MU-IND-iCCA /DBDH	MU-sUF-iCMA /CDH
Construction 1 (ours) (BCHK[4],Waters[15],Okamoto[13])	MU-IND-iCCA /DBDH	MU-sUF-iCMA /CDH
Construction 2 (ours) (BCHK[4],Waters[15])	MU-IND-iCCA /DBDH	MU-sUF-iCMA /CDH

Table 3. Comparison of computational cost: exp stands for exponentiation, and m-exp stands for multi-exponentiation. Group multiplications and computation costs of hash functions and symmetric-key primitives (i.e., DEMs) are not considered here.

Construction	Computational cost	
	SC	USC
SC_{tk} [9]	5 exps, 1 m-exp	1 exp, 1 m-exp, 3 pairing
SC_{kem} [9]	4 exps, 2 m-exp	1 exp, 1 m-exp, 3 pairings
Construction 1	4 exps, 3 m-exps	1 exp, 2 m-exps 3 pairings
Construction 2	4 exps, 1 m-exp	1 m-exp, 3 pairings

public/secret-keys in our constructions are equal or slightly shorter than those of SC_{tk} and SC_{kem} , while Constructions 1 and 2 are inferior to both of SC_{tk} and SC_{kem} in terms of ciphertext-size.

Finally, we summarize the above discussion. Our construction achieves MU-IND-iCCA security under DBDH assumption and MU-sUF-iCMA security under CDH assumption with the same or shorter sizes of public/secret-keys and less computational cost compared to other existing constructions, though ciphertext-size becomes larger. In this sense, there is a tradeoff between parameters of efficiency. Thus, our construction would be useful in the situation where large channel capacity for communication is available to transmit ciphertexts. However, if not (i.e., channel capacity is limited), our construction is not well suited. Therefore, it would be possible to adapt a construction of signcryption effectively and properly to suit the situation of applications.

Table 4. Comparison of sizes of ciphertexts and keys: $|\mathbb{G}_{BG}|$ is bit-length of elements in a bilinear group. $|\mathbb{G}_e|$ is bit-length of elements in an elliptic curve. $|\mathbb{Z}_p|$ is the bit-length of elements in a field of prime order p . $|MAC|$ is the bit-length of MAC tag. $|com|$ and $|dec|$ are bit-length of strings in an encapsulation scheme that require $3|com| \leq |dec|$ (see [4]). n is the bit-length of output of a collision-resistant hash function such that $p \geq 2^n$. We assume that DEM has no ciphertext overhead. For example, if we consider 80-bit security, we can apply $|\mathbb{G}_e| = |\mathbb{Z}_p| = 160$ bits, $|\mathbb{G}_{BG}| = 171$ bits, $|MAC| = |com| = 80$ bits, and $|dec| = 240$ bits.

Construction	Ciphertext size	Receiver's key size		Sender's key size	
		pk	sk	pk	sk
$SC_{tk}[9]$	$4 \mathbb{G}_{BG} + \mathbb{Z}_p $	$(2 + n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} + (1 + n) \mathbb{Z}_p $	$(5 + n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $
$SC_{kem}[9]$	$4 \mathbb{G}_{BG} + \mathbb{Z}_p + MAC $	$3 \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} + 2 \mathbb{Z}_p $	$(5 + n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $
Construction 1	$4 \mathbb{G}_{BG} + 2 \mathbb{Z}_p + 4 \mathbb{G}_e $	$3 \mathbb{G}_{BG} $	$3 \mathbb{Z}_p $	$(4 + n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $
Construction 2	$4 \mathbb{G}_{BG} + MAC + com + dec $	$3 \mathbb{G}_{BG} $	$3 \mathbb{Z}_p $	$(4 + n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $

6 Concluding Remarks

We emphasize that our main purpose was to propose a simple approach to constructing strongly secure signcryption in the multi-user setting based on well-known techniques for constructing strongly secure PKE from IBE. In this paper, we have successfully proposed such two generic constructions of signcryption schemes without random oracles, and our constructions achieve the strongest security (i.e., both of MU-IND-iCCA and MU-sUF-iCMA) without KR. Moreover, by applying reasonable instantiations of cryptographic primitives to our generic constructions, we have shown that our constructions have the same or shorter size of public/secret-keys and less computational cost, where the latter is true only for our Construction 2, compared to other existing constructions, however, our constructions have larger ciphertext-size. In this sense, there is a tradeoff between parameters of efficiency. Therefore, it would be possible to adapt a construction of signcryption effectively and properly to suit the situation of applications.

Acknowledgments. The authors wish to thank anonymous referees for their invaluable comments.

References

1. An, J.H., Dodis, Y., Rabin, T.: On the Security of Joint Signature and Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)

2. Baek, J., Steinfeld, R., Zheng, Y.: Formal Proofs for the Security of Signcryption. *J. Cryptology* 20(2), 203–235 (2007)
3. Bellare, M., Shoup, S.: Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir Without Random Oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (2007)
4. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext Security from Identity-based Encryption. *SIAM J. Comput.* 36(5), 1301–1328 (2007)
5. Boneh, D., Katz, J.: Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
6. Boneh, D., Shen, E., Waters, B.: Strongly Unforgeable Signatures Based on Computational Diffie-Hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006)
7. Boyen, X., Mei, Q., Waters, B.: Direct Chosen Ciphertext Security from Identity-based Techniques. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM CCS 2005, pp. 320–329. ACM (2005)
8. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
9. Chiba, D., Matsuda, T., Schuldt, J.C.N., Matsuura, K.: Efficient Generic Constructions of Signcryption with Insider Security in the Multi-user Setting. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 220–237. Springer, Heidelberg (2011)
10. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
11. Libert, B., Quisquater, J.-J.: Efficient Signcryption with Key Privacy from Gap Diffie-Hellman Groups. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 187–200. Springer, Heidelberg (2004)
12. Matsuda, T., Matsuura, K., Schuldt, J.C.N.: Efficient Constructions of Signcryption Schemes and Signcryption Composability. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 321–342. Springer, Heidelberg (2009)
13. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
14. Tan, C.H.: Signcryption Scheme in Multi-user Setting without Random Oracles. In: Matsuura, K., Fujisaki, E. (eds.) IWSEC 2008. LNCS, vol. 5312, pp. 64–82. Springer, Heidelberg (2008)
15. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
16. Zheng, Y.: Digital Signcryption or How to Achieve $\text{Cost}(\text{signature} \ \& \ \text{encryption}) \ll \text{Cost}(\text{signature}) + \text{Cost}(\text{encryption})$. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)

Appendix A: Proof of Theorem 1

Suppose that there exists a probabilistic PTA A that breaks the MU-IND-iCCA security of the signcryption scheme SCS with advantage δ . We say that a ciphertext $\sigma = (vk, C, s)$ is valid if $\text{OTS.Vrfy}(vk, C, s) = 1$. Let $\sigma^* = (vk^*, C^*, s^*)$ be

a challenge ciphertext received by A. *Forge* denotes the event that A submits a valid (pk_S, vk^*, C, s) to the unsigncrypt oracle. *Succ* denotes the event that the adversary wins the game. Then we have

$$\begin{aligned} Adv_{SCS,A}^{MU-IND-iCCA} &= \left| \Pr[\text{Succ}] - \frac{1}{2} \right| \\ &\leq \left| \Pr[\text{Succ} \wedge \text{Forge}] - \frac{1}{2} \Pr[\text{Forge}] \right| \\ &\quad + \left| \Pr[\text{Succ} \wedge \overline{\text{Forge}}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2} \right| \\ &\leq \frac{1}{2} \Pr[\text{Forge}] + \left| \Pr[\text{Succ} \wedge \overline{\text{Forge}}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2} \right|. \end{aligned}$$

To complete the proof, we need the following lemmas.

Lemma 1. $\Pr[\text{Forge}] \leq q\epsilon_2$.

Lemma 2. $\left| \Pr[\text{Succ} \wedge \overline{\text{Forge}}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2} \right| \leq \epsilon_1$.

The proofs of the above lemmas will be given in the full version of this paper. By Lemmas 1 and 2, we obtain

$$Adv_{SCS,A}^{MU-IND-iCCA} \leq \epsilon_1 + \frac{q}{2}\epsilon_2.$$

This completes the proof of Theorem 1. □

Appendix B: Proof of Theorem 2

Suppose that there exists a probabilistic PTA A that breaks the MU-sUF-iCMA security of the signcryption scheme SCS. We use A to construct a probabilistic PTA B that breaks the UF-CMA security of the digital signature DS as follow: B is given as input a public parameter prm and a verification-key $VK = pk_S$, and runs $(pk_R, sk_R) \leftarrow \text{KeyGen}_R(prm)$, and then runs $A(prm, pk_S)$. B can answer any signcrypt queries from A by using its own signing oracle. This provides perfect simulation of the view of A in the game. If A finally outputs a valid forgery $(pk_R^*, sk_R^*, vk^*, C^*, s^*)$, it follows from the USC algorithm that $\text{DS.Vrfy}(pk_S^*, M^* || vk^* || pk_R^*, S^*) = 1$. Thus $(M^* || vk^* || pk_R^*, S^*)$ is a valid forgery for B if $M^* || vk^* || pk_R^*$ was never previously queried to the signing oracle. Let *Bad* denotes the event that $M^* || vk^* || pk_R^*$ was previously queried to the signing oracle, and *Succ* denotes the event that the adversary wins the game. Then, it follows that

$$\epsilon_1 \geq Adv_{DS,B}^{UF-CMA} \geq \Pr[\text{Succ} \wedge \overline{\text{Bad}}].$$

Furthermore, we use A to construct a probabilistic PTA F that breaks the sUF-OT security of the one-time signature OTS as follows: F is given as input a public parameter prm and a verification-key vk^* , and it runs $(pk_R, sk_R) \leftarrow$

$\text{KeyGen}_R(\text{prm})$, $(pk_S, sk_S) \leftarrow \text{KeyGen}_S(\text{prm})$, and then runs $A(\text{prm}, pk_S)$. Let q be the number of queries to the signcrypt oracle. When A submits a j -th signcrypt query (for $1 \leq j \leq q$), F can answer it by using generated one-time key pair (vk_j, sk_j) or using its own signing oracle only once. This provides perfect simulation of the view of A in the game. If A finally outputs a valid forgery $(pk_R^*, sk_R^*, vk^*, C^*, s^*)$, it follows from the USC algorithm that $\text{OTS.Vrfy}(vk^*, C^*, s^*) = 1$. When $M^* \| vk^* \| pk_R^*$ was previously signed by F , the event **Bad** occurs. In this case, (C^*, s^*) is a valid forgery for F , if F did not previously query C^* to the signing oracle and did not receive back s^* as the signature. Then, it follows that

$$q\epsilon_2 \geq q \cdot \text{Adv}_{\text{OTS},F}^{\text{sUF-OT}} \geq \Pr[\text{Succ} \wedge \overline{\text{Bad}}].$$

Thus, we have

$$\begin{aligned} \text{Adv}_{\text{SCS},A}^{\text{MU-sUF-iCMA}} &= \Pr[\text{Succ} \wedge \overline{\text{Bad}}] + \Pr[\text{Succ} \wedge \text{Bad}] \\ &\leq \text{Adv}_{\text{DS},B}^{\text{UF-CMA}} + q \cdot \text{Adv}_{\text{OTS},F}^{\text{sUF-OT}} \\ &\leq \epsilon_1 + q\epsilon_2. \end{aligned}$$

This completes the proof of Theorem 2. \square

Appendix C: Proof of Theorem 3

Suppose that there exists a probabilistic PTA A that breaks the MU-IND-iCCA security of the signcryption scheme SCS. We use A to construct a probabilistic PTA B that breaks the IND-sID-CPA security of the identity-based encryption IBE, or a probabilistic PTA D that breaks security of the encapsulation scheme, or a probabilistic PTA F that breaks the sUF-OT security of the MAC. Then, we define the following games.

Game₀: This is the ordinary MU-IND-iCCA game.

Game₁: When A outputs the query (pk_S, com^*, C, τ) in the Game₀, the oracle returns \perp .

Game₂: A submits two plaintexts M_0 and M_1 , and then, we compute the challenge ciphertext $\sigma^* = (com^*, C^*, \tau^*)$ by $C^* \leftarrow \text{IBE.Enc}(pk_R, com^*, 0^{|M_0|} \| S^* \| 0^{|dec|})$, $\tau^* \leftarrow \text{MAC.Sign}(r^*, C^*)$ in Game₁.

Game₃: The components com^* and C^* of the challenge ciphertext are computed as in Game₂. However, the component τ^* is computed by choosing a random key r .

Then, we define the following events.

- **Succ _{i}** : This is the event that A wins in the Game _{i} .
- **Valid _{i}** : This is the event that A submits a valid ciphertext $\sigma = (com^*, C, \tau)$ in Game _{i} .
- **NoBind _{i}** : This is the event that A submits a ciphertext $\sigma = (com^*, C, \tau)$ such that $M \| S \| dec \leftarrow \text{IBE.Dec}(SK_{com^*}, com^*, C)$ and $\text{E.Rec}(\text{prm}, com^*, dec) = r \notin \{r^*, \perp\}$ in Game _{i} .

- Forge_i : This is the event that A submits a ciphertext $\sigma = (com^*, C, \tau)$ for which $\text{MAC.Vrfy}(r^*, C, \tau) = 1$ in Game_i .

Now, we note that Game_0 and Game_1 are the same until Valid occurs. Thus, we have $|\Pr[\text{Succ}_1] - \Pr[\text{Succ}_0]| \leq \Pr[\text{Valid}_1]$. Furthermore, in Valid_1 , the query includes com^* and it is valid. Hence, we have $\Pr[\text{Valid}_1] \leq \Pr[\text{NoBind}_1] + \Pr[\text{Forge}_1]$. Then, it follows that

$$\begin{aligned} \left| \Pr[\text{Succ}_0] - \frac{1}{2} \right| &\leq |\Pr[\text{Succ}_0] - \Pr[\text{Succ}_1]| + \left| \Pr[\text{Succ}_1] - \frac{1}{2} \right| \\ &\leq \Pr[\text{NoBind}_1] + \Pr[\text{Forge}_1] \\ &\quad + |\Pr[\text{Succ}_1] - \Pr[\text{Succ}_2]| + \left| \Pr[\text{Succ}_2] - \frac{1}{2} \right| \\ &\leq \Pr[\text{NoBind}_1] + \Pr[\text{Forge}_3] \\ &\quad + |\Pr[\text{Forge}_2] - \Pr[\text{Forge}_3]| \\ &\quad + |\Pr[\text{Forge}_1] - \Pr[\text{Forge}_2]| \\ &\quad + |\Pr[\text{Succ}_1] - \Pr[\text{Succ}_2]| + \left| \Pr[\text{Succ}_2] - \frac{1}{2} \right|. \end{aligned}$$

To complete the proof, we need the following lemmas.

Lemma 3. $\Pr[\text{NoBind}_1] \leq \epsilon_3$.

Lemma 4. $\Pr[\text{Succ}_2] = \frac{1}{2}$, $|\Pr[\text{Succ}_1] - \Pr[\text{Succ}_2]| \leq 2\epsilon_1$, and $|\Pr[\text{Forge}_1] - \Pr[\text{Forge}_2]| \leq 2\epsilon_1$.

Lemma 5. $|\Pr[\text{Forge}_2] - \Pr[\text{Forge}_3]| \leq \epsilon_3$ and $\Pr[\text{Forge}_3] \leq q\epsilon_2$.

The proofs of the above lemmas will be given in the full version of this paper. By Lemmas 3, 4, and 5, we obtain

$$\text{Adv}_{\text{SCS},A}^{\text{MU-IND-iCCA}} \leq 4\epsilon_1 + q\epsilon_2 + 2\epsilon_3.$$

This completes the proof. □

Appendix D: Proof of Theorem 4

Suppose that there exists a probabilistic PTA A that breaks the MU-sUF-iCMA security of the signcryption scheme SCS. We use A to construct a probabilistic PTA B that breaks the UF-CMA security of the digital signature DS as follows.

B is given as input a public parameter prm and a verification-key $VK = pk_S$, and runs $(pk_R, sk_R) \leftarrow \text{KeyGen}_R(prm)$, and then runs $A(prm, pk_S)$. B can answer any signcrypt queries from A by using its own signing oracle. This provides perfect simulation of the view of A in the game. If A finally outputs a valid forgery $(pk_R^*, sk_R^*, com^*, C^*, \tau^*)$, it follows from the USC algorithm that $\text{DS.Vrfy}(pk_S^*, M^* || com^* || pk_R^*, S^*) = 1$. Thus, $(M^* || com^* || pk_R^*, S^*)$ is a valid

forgery for B if $M^*||com^*||pk_R^*$ was never previously queried to the signing oracle. Let **Bad** denotes the event that $M^*||com^*||pk_R^*$ was previously queried to the signing oracle, and **Succ** denotes the event that the adversary wins the game. Then, it follows that

$$\epsilon_1 \geq Adv_{DS,B}^{UF-CMA} \geq \Pr[\text{Succ} \wedge \overline{\text{Bad}}].$$

Then, we consider the following three cases in the event **Bad**.

- **E₁**: A outputs a forgery $(pk_R^*, sk_R^*, com^*, C^*, \tau^*)$ such that $M^*||S^*||dec' \leftarrow \text{IBE.Dec}(SK_{com^*}, com^*, C^*)$ and $\text{E.Rec}(prm, com^*, dec') \notin \{\perp, r^*\}$. Consider an adversary D_2 which breaks the binding security of the encapsulation scheme as follows: given as input (prm, com^*, dec^*) , D_2 runs $(pk_R, sk_R) \leftarrow \text{IBE.Kg}(prm)$, $(pk_S, sk_S) \leftarrow \text{DS.Kg}(prm)$ and then runs $\text{A}(prm, pk_S)$. Whenever A submits a query to its signcrypt oracle, D_2 can respond to this query in the usual way. Specifically, D_2 computes $S^* \leftarrow \text{DS.Sign}(sk_S, M^*||com^*||pk_R^*)$ at least once. Finally, A outputs $(pk_R^*, sk_R^*, com^*, C^*, \tau^*)$, then D_2 outputs a value dec' such that $\text{E.Rec}(prm, com^*, dec') \notin \{\perp, r^*\}$. If **E₁** occurs, then D_2 wins. Let Succ_{E_i} denotes the event that the adversary wins in the event **E_i**. Then, we have

$$\epsilon_3 \geq Adv_{\text{Encap}, D_2}^{\text{binding}} \geq \Pr[\text{Succ}_{E_1}].$$

- **E₂**: A outputs a forgery $(pk_R^*, sk_R^*, com^*, C^*, \tau^*)$ such that $\text{MAC.Vrfy}(r_b^*, c, \tau) = 1$. Consider an adversary D_1 which breaks the hiding security of the encapsulation scheme as follows: given as input (prm, com^*, r_b^*) , D_1 runs $(pk_R, sk_R) \leftarrow \text{IBE.Kg}(prm)$, $(pk_S, sk_S) \leftarrow \text{DS.Kg}(prm)$ and then runs $\text{A}(prm, pk_S)$. Whenever A submits a query to its signcrypt oracle, D_1 can respond to this query in the usual way. Specifically, D_1 computes $S^* \leftarrow \text{DS.Sign}(sk_S, M^*||com^*||pk_R^*)$ at least once. Finally, A outputs $(pk_R^*, sk_R^*, com^*, C^*, \tau^*)$, then D checks whether $\text{MAC.Vrfy}(r_b^*, C^*, \tau^*) = 1$. If so, D_1 outputs 1, otherwise 0. If **E₂** occurs, then D_1 wins. Thus, we have

$$\epsilon_3 \geq Adv_{\text{Encap}, D_1}^{\text{hiding}} \geq \Pr[\text{Succ}_{E_2}].$$

- **E₃**: Other cases.

Consider an adversary F which breaks the sUF-OT security of MAC as follows: given as input prm , F runs $(pk_R, sk_R) \leftarrow \text{IBE.Kg}(prm)$, $(pk_S, sk_S) \leftarrow \text{DS.Kg}(prm)$ and then runs $\text{A}(prm, pk_S)$. When A submits a j -th query to its signcrypt oracle (for $1 \leq j \leq q$), F can answer it by using generated key r_j or using its own signing oracle only once. F provides perfect simulation for A in the game. Finally, A outputs $(pk_R^*, sk_R^*, com^*, C^*, \tau^*)$, and then F outputs (C^*, τ^*) . When $M^*||com^*||pk_R^*$ was previously signed by F, the event **E₃** occurs. In this case, (C^*, τ^*) is a valid forgery for F, if F did not previously query C^* to the signing oracle and did not receive back τ^* as the tag. Thus, if **E₃** occurs, then F wins, and we have

$$q\epsilon_2 \geq q \cdot Adv_{\text{MAC}, F}^{sUF-OT} \geq \Pr[\text{Succ}_{E_3}].$$

From the above discussion, we have

$$\begin{aligned}
 Adv_{SCS,A}^{MU-sUF-iCMA} &= \Pr[\text{Succ} \wedge \overline{\text{Bad}}] + \Pr[\text{Succ} \wedge \text{Bad}] \\
 &= \Pr[\text{Succ} \wedge \overline{\text{Bad}}] + \Pr[\text{Succ}_{E_1}] + \Pr[\text{Succ}_{E_2}] + \Pr[\text{Succ}_{E_3}] \\
 &\leq Adv_{DS,B}^{UF-CMA} + Adv_{\text{Encap},D_2}^{\text{binding}} + Adv_{\text{Encap},D_1}^{\text{hiding}} + q \cdot Adv_{MAC,F}^{sUF-OT} \\
 &\leq \epsilon_1 + q\epsilon_2 + 2\epsilon_3.
 \end{aligned}$$

This completes the proof. □

Appendix E: Tables for Primitives Used for Signcryption

We summarize constructions of primitives for confidentiality (encryption) in Table 5, and constructions of primitives for integrity (signatures) in Table 6. These constructions are used as instantiations applied to general constructions of signcrypts in Section 5.2.

Table 5. Constructions of primitives for confidentiality (encryption)

Construction /Primitive	Security /Assumption	Computational cost Enc/Dec	Cipher- text size	Key size	
				pk	sk
tBMW1[12] /TBKEM	IND-tag-CCA /DBDH	3 exps /1 exp, 1 pairing	$2 \mathbb{G}_{BG} $	$(2+n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $ $+(1+n) \mathbb{Z}_p $
BMW2[7] /KEM	IND-CCA /DBDH	2 exps, 1 m-exp /1 exp, 1 pairing	$2 \mathbb{G}_{BG} $	$3 \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $ $+2 \mathbb{Z}_p $
BCHK[4] /IBE	IND-sID-CPA /DBDH	2 exps, 1 m-exp /2 pairings	$2 \mathbb{G}_{BG} $	$3 \mathbb{G}_{BG} $	$3 \mathbb{Z}_p $

Table 6. Constructions of primitives for integrity (signatures): DL means Discrete Logarithm assumption; “–” means that none of operations of exponentiation, multi-exponentiation, and pairing is used

Construction /Primitive	Security /Assumption	Computational cost Sig/Ver	Signature size	Key size	
				pk	sk
BSW[6] /DS	sUF-CMA /CDH	2 exps, 1 m-exp /1 m-exp, 2 pairings	$2 \mathbb{G}_{BG} $ $+ \mathbb{Z}_p $	$(5+n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $
Waters[15] /DS	UF-CMA /CDH	2 exps /2 pairings	$2 \mathbb{G}_{BG} $	$(4+n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $
Okamoto[13] /OTS	sUF-OT /DL	– / 1 exp, 1 m-exp	$2 \mathbb{Z}_p $	$4 \mathbb{G}_e $	$ \mathbb{G}_e $ $+4 \mathbb{Z}_p $
BS[3] using [15] and [13] /DS	sUF-CMA /CDH	2 exps, 2 m-exps / 1 exp, 1 m-exp , 2 pairings	$2 \mathbb{G}_{BG} $ $+2 \mathbb{Z}_p $ $+4 \mathbb{G}_e $	$(4+n) \mathbb{G}_{BG} $	$ \mathbb{G}_{BG} $

Anonymous Constant-Size Ciphertext HIBE from Asymmetric Pairings

Somindu C. Ramanna and Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108
{somindu_r,palash}@isical.ac.in

Abstract. We present a new hierarchical identity-based encryption (HIBE) scheme with constant-size ciphertexts that can be implemented using the most efficient bilinear pairings, namely, Type-3 pairings. In addition to being fully secure, our scheme is anonymous. The HIBE is obtained by extending an asymmetric pairing based IBE scheme due to Lewko and Waters. The extension uses the approach of Boneh-Boyen-Goh to obtain constant-size ciphertexts and that of Boyen-Waters for anonymity. Security argument is based on the dual-system technique of Waters. The resulting HIBE is the only known scheme using Type-3 pairings achieving constant-size ciphertext, security against adaptive-identity attacks and anonymity under static assumptions without random oracles.

Keywords: identity-based encryption(IBE), constant-size ciphertext hierarchical IBE, asymmetric pairings, dual-system encryption.

1 Introduction

The notion of identity-based encryption (IBE) was introduced by Shamir [19] and the first IBE schemes appeared later [6,3]. In IBE, a sender encrypts a message using the receiver's identity itself as the public key and a central authority called private key generator (PKG) generates and securely distributes decryption keys corresponding to identities of different users. Hierarchical IBE (HIBE), proposed by [11,12], reduces the workload of the PKG by allowing it to delegate the key generation ability to lower-level entities. As a result, an individual user can conveniently obtain a decryption key from a lower-level entity instead of obtaining it from the PKG.

Type-3 Pairings: Most practical (H)IBE schemes are built using a bilinear pairing which maps $\mathbb{G}_1 \times \mathbb{G}_2$ to \mathbb{G}_T , where $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are groups of the same order. Well-known examples of such maps arise by suitably choosing \mathbb{G}_1 and \mathbb{G}_2 to be groups of elliptic curve points and \mathbb{G}_T to be a subgroup of the multiplicative group of a finite field. From an implementation point of view, it is most efficient to use bilinear maps where the (common) group order is prime and

it is computationally difficult to find an isomorphism from \mathbb{G}_1 to \mathbb{G}_2 or vice versa. Such pairings are called Type-3 pairings [5,20,10]. Less efficient alternatives are when \mathbb{G}_1 and \mathbb{G}_2 are same (called Type-1 pairings) or when the common group order is a composite number (called composite-order pairings). IBE or HIBE schemes based on Type-3 pairings would have the fastest algorithms and the most compact representations of parameters.

Constant-Size Ciphertext HIBE: In HIBE, identities consist of tuples of varying lengths. Encryption of a message is done for a specified identity tuple. In many HIBE schemes, as the length of the identity tuple increases, so does the length of the resulting ciphertext. Consequently, the bandwidth requirement for communicating the ciphertext also increases.

The solution to this issue is to require the ciphertext size to be independent of the length of the identity tuple. Then, irrespective of the length of the identity tuple, the bandwidth required for the ciphertext would be the same. Such a scheme is called a constant-size ciphertext HIBE. The first such HIBE scheme was proposed by Boneh, Boyen and Goh [2]. While the scheme itself is quite elegant, its proof of security was in a very restricted attack model, the so-called selective-identity model. Lewko and Waters [15] provided the first constant-size ciphertext HIBE scheme which is secure against the usual adaptive-identity attacks. The drawback, however, was that the scheme in [15] used pairings on composite order groups and could not be instantiated with the more efficient Type-3 pairings.

In the following, we use the abbreviation CC-HIBE to denote HIBE schemes with constant-size ciphertexts. We clarify that the constant size here only refers to the number of group elements in the ciphertext. The size of representation of the group elements, however, needs to increase if the value of the security parameter increases.

Anonymity: In (H)IBE schemes with anonymity, ciphertexts do not reveal any information about the identity of the recipient. Abdalla *et.al.* [1] first formalised the notion of anonymity and used it to construct public key encryption with keyword search (PEKS). PEKS enables search on encrypted documents based on some keywords and this capability for search is delegatable. Anonymous HIBE schemes provide means to extend PEKS to more sophisticated primitives such as public key encryption with temporary keyword search (PETKS) and identity-based encryption with keyword search (IBEKS). The first construction of anonymous HIBE without random oracles was given by [4] with security in the selective-id model. Later constructions by [18,7] could achieve security in the adaptive-id setting but were based on composite-order pairings. Two other constructions [8,16] used asymmetric pairings but with security in the selective-id model.

1.1 Our Contributions

Our main motivation in this work is to obtain a constant-size ciphertext HIBE which can be implemented using Type-3 pairings. This allows the benefits of

having constant-size ciphertexts to be combined with the efficiency benefits of using Type-3 pairings. These efficiency considerations are attained while retaining the usual provable guarantees, namely security against adaptive-identity attacks, use of static hardness assumptions, no degradation of security with increase in the depth of the HIBE and the avoidance of random oracles.

The provable properties are achieved using the extremely useful idea of dual-system encryption introduced by Waters [21]. This technique was used by Lewko and Waters [15] to construct an IBE and a CC-HIBE scheme based on composite-order pairings. The authors in [15] went on to convert their composite-order pairing based IBE scheme to one which can be instantiated using Type-3 pairings. However, no such conversion was done for the HIBE scheme in [15] and the authors do not make any remark on whether this can be done or how difficult it would be to do so.

The starting point of our work are the IBE schemes in [15]. Two IBE schemes are given in [15] where the first one is in the setting of composite order groups and the second one is in the Type-3 setting. The IBE in the composite order setting is not anonymous (shown in [7]) due to the following reason – the identity-hash in both the ciphertext and key live in the same subgroup; moreover, elements used to create the hash are public thus providing a test for the recipient identity for any ciphertext. On the other hand, the Type-3 variant, which we refer to as “LW-IBE”, is anonymous. This is because ciphertexts live in \mathbb{G}_1 , keys in \mathbb{G}_2 and the elements required to create the hash in \mathbb{G}_2 are kept secret. Hence there would be no way to test whether a given ciphertext is encrypted to a particular identity or not. However, there has been no proof of anonymity in any follow-up work. The first contribution of the current work is to show that the LW-IBE is anonymous. Two static (though non-standard) computational assumptions (which we denote as LW1 and LW2) along with decision bilinear Diffie-Hellman (DBDH) assumption are used in [15] to show the security of LW-IBE. For proving anonymity, we need to introduce a new computational assumption, called A1, which is again static, but, non-standard.

The second contribution of this paper is to extend the LW-IBE to a constant-size ciphertext HIBE. At a very basic level, the idea for obtaining constant-size ciphertexts is to use the identity hashing technique suggested in [2] over existing IBE schemes. We will refer to this as BBG-hash or BBG-extension. We do not take the path of converting the composite-order pairing based HIBE of [15]. Techniques for such conversions have been proposed by Freeman [9] and Lewko [14]. The latter uses *dual pairing vector spaces* (DPVSs) constructed over pairing groups to simulate features of composite order pairings. But it seems hard to retain the constant size of ciphertexts using these conversion techniques. Instead, we start with LW-IBE and extend it to a CC-HIBE by plugging in the BBG-hash. One complication in doing so arises. In the dual-system technique, two kinds of ciphertexts and keys are defined – *normal* and *semi-functional*. Semi-functional components are required only for proving security and are generated using some secret elements during simulation. The main elements of a dual system proof would be appropriately defining semi-functional components

and generating them using a problem instance in the reduction ensuring correct distribution of all elements provided to the attacker. Extending the decryption key of LW-IBE to the decryption key of a HIBE in a straightforward manner does not retain the structure required for a dual-system proof. Our way of tackling this is to add additional components to the decryption key. On the face of it, this complicates the key generation and delegation mechanisms. However, somewhat counter-intuitively, adding this extra level of complication allows the security reductions to go through.

An offshoot of the extension is that the scheme becomes anonymous. This is because in LW-IBE, the semi-functional space (for both ciphertexts and keys) is created using some secret elements (part of the master secret). The same elements are implicitly used in creating ciphertexts and keys. In case of a direct extension to HIBE, all these elements may have to be revealed in the public parameters to facilitate re-randomisation during delegation of keys. This makes the scheme non-anonymous but at the same time affects dual system arguments for which keeping the elements secret is essential. The way out is to make the scheme anonymous. We also provide a proof of anonymity based on a static assumption.

The computational assumptions required to obtain CPA-security are those used in [15] along with the new assumption required to show that the LW-IBE is anonymous. The last assumption is used to prove the anonymity of the HIBE scheme.

2 Preliminaries

Some basic notation, definitions and the complexity assumptions used in our proofs are presented in this section.

2.1 Notation

For a set \mathcal{X} , the notation $x_1, \dots, x_k \in_{\mathbb{R}} \mathcal{X}$ (or $x_1, \dots, x_k \stackrel{\mathbb{R}}{\leftarrow} \mathcal{X}$) indicates that x_1, \dots, x_k are elements of \mathcal{X} chosen independently at random according to some distribution \mathbb{R} . We use two notations interchangeably. The uniform distribution is denoted by \mathbb{U} . For a (probabilistic) algorithm \mathcal{A} , $x \leftarrow \mathcal{A}(\cdot)$ means that x is chosen according to the output distribution of \mathcal{A} (which of course may be determined by its input). For two integers $a < b$, the notation $[a, b]$ represents the set $\{x \in \mathbb{Z} : a \leq x \leq b\}$. Let \mathbb{G} be a finite cyclic group and \mathbb{G}^\times denote the set of generators of \mathbb{G} . Fix a generator $P_1 \in \mathbb{G}^\times$. The discrete logarithm of an element $Q \in \mathbb{G}$ to base P_1 is written as $\text{dlog}_{P_1} Q$.

2.2 Bilinear Pairings

A bilinear pairing is given by a 7-tuple $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$ where $\mathbb{G}_1 = \langle P_1 \rangle$, $\mathbb{G}_2 = \langle P_2 \rangle$ are written additively and \mathbb{G}_T , a multiplicatively written group, all having the same order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a map with the following properties.

1. *Bilinear*: For $P_1, Q_1 \in \mathbb{G}_1$ and $P_2, Q_2 \in \mathbb{G}_2$, the following holds:
 $e(P_1, P_2 + Q_2) = e(P_1, P_2)e(P_1, Q_2)$ and $e(P_1 + Q_1, P_2) = e(P_1, P_2)e(Q_1, P_2)$.
2. *Non-degenerate*: If $e(P_1, P_2) = 1_T$, the identity element of \mathbb{G}_T , then either P_1 is the identity of \mathbb{G}_1 or P_2 is the identity of \mathbb{G}_2 .
3. *Efficiently computable*: The function e should be efficiently computable.

Three main types of pairings have been identified in the literature [20,10].

Type-1. In this type, the groups \mathbb{G}_1 and \mathbb{G}_2 are the same.

Type-2. $\mathbb{G}_1 \neq \mathbb{G}_2$ and an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is known.

Type-3. Here, $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable isomorphisms between \mathbb{G}_1 and \mathbb{G}_2 are known.

It has been reported [5,20,10] that from an implementation point of view, Type-3 pairings are the fastest to compute and further provide the most compact description of group elements. So, building functionalities which can be instantiated with such pairings is of practical interest. This work is entirely based on Type-3 pairings. The terms ‘Type-3 pairing’ and ‘asymmetric pairing’ are used interchangeably in the rest of the paper.

Note: We introduce some notation: fix $P_1 \in \mathbb{G}_1^\times$ and $P_2 \in \mathbb{G}_2^\times$; for elements $R_1 \in \mathbb{G}_1$ and $R_2 \in \mathbb{G}_2$, the notation $R_1 \sim R_2$ indicates that $\text{dlog}_{P_1} R_1 = \text{dlog}_{P_2} R_2$. The fixed generators P_1 and P_2 will be clear from the context.

2.3 Complexity Assumptions

Described here are certain hardness assumptions in Type-3 setting that are needed for the security reductions. Let $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$ be an asymmetric pairing and \mathcal{A} , a probabilistic polynomial time (PPT) algorithm \mathcal{A} that outputs 0 or 1. We will first define a generic static decision problem Π over a \mathcal{G} . Denote by \mathcal{D} , a distribution consisting of a constant number of elements from \mathbb{G}_1 and \mathbb{G}_2 . Let T_1, T_2 be two elements chosen (according to some distributions) from one of the three groups. The goal of \mathcal{A} is to distinguish between the two distributions (\mathcal{D}, T_1) and (\mathcal{D}, T_2) . The advantage of \mathcal{A} in solving Π is given by

$$\text{Adv}_{\mathcal{G}}^{\Pi}(\mathcal{A}) = |\Pr[\mathcal{A}(\mathcal{D}, T_1) = 1] - \Pr[\mathcal{A}(\mathcal{D}, T_2) = 1]|.$$

We say that (ε, t) - Π assumption holds if for any t -time algorithm \mathcal{A} , $\text{Adv}_{\mathcal{G}}^{\Pi}(\mathcal{A}) \leq \varepsilon$.

Next the required assumptions are stated as instantiations of Π by suitably defining \mathcal{D} and T_1, T_2 .

Assumption LW1 [15]

$$\begin{aligned} &F_1 \xleftarrow{\text{U}} \mathbb{G}_1^\times; F_2 \xleftarrow{\text{U}} \mathbb{G}_2^\times, a, b, s \xleftarrow{\text{U}} \mathbb{Z}_p, Y_1 \xleftarrow{\text{U}} \mathbb{G}_1; \\ &\mathcal{D} = (\mathcal{G}, F_1, bsF_1, sF_1, aF_1, ab^2F_1, bF_1, b^2F_1, asF_1, b^2sF_1, b^3F_1, b^3sF_1, F_2, bF_2), \\ &T_1 = ab^2sF_1, \quad T_2 = Y_1. \end{aligned}$$

Assumption LW2 [15]

$$\begin{aligned}
 &F_1 \xleftarrow{U} \mathbb{G}_1^\times; F_2 \xleftarrow{U} \mathbb{G}_2^\times, d, b, c, x \xleftarrow{U} \mathbb{Z}_p, Y_2 \xleftarrow{U} \mathbb{G}_2; \\
 &\mathcal{D} = (\mathcal{G}, F_1, dF_1, d^2F_1, bxF_1, dbxF_1, d^2xF_1, F_2, dF_2, bF_2, cF_2), \\
 &T_1 = bcF_2, \quad T_2 = Y_2.
 \end{aligned}$$

Decisional Bilinear Diffie-Hellman in Type-3 pairings (DBDH-3) [5]

$$\begin{aligned}
 &F_1 \xleftarrow{U} \mathbb{G}_1^\times; F_2 \xleftarrow{U} \mathbb{G}_2^\times; x, y, z \xleftarrow{U} \mathbb{Z}_p; Y_T \xleftarrow{U} \mathbb{G}_T; \\
 &\mathcal{D} = (\mathcal{G}, F_1, xF_1, yF_1, zF_1, F_2, xF_2, yF_2), \\
 &T_1 = e(F_1, F_2)^{xyz}, \quad T_2 = Y_T.
 \end{aligned}$$

Assumption A1.

$$\begin{aligned}
 &F_1 \xleftarrow{U} \mathbb{G}_1^\times; F_2 \xleftarrow{U} \mathbb{G}_2^\times; a, z, d, s, x \xleftarrow{U} \mathbb{Z}_p; \\
 &\mathcal{D} = (\mathcal{G}, F_1, zF_1, dzF_1, azF_1, adzF_1, szF_1, F_2, zF_2, aF_2, xF_2, (dz - ax)F_2), \\
 &T_1 = sdzF_1, \quad T_2 = Y_1.
 \end{aligned}$$

Discussion. We introduce assumption A1 to show anonymity of LW-IBE as well as our HIBE scheme. The challenge in A1 is an element $Z_1 \in \mathbb{G}_1$; the task is to decide whether $Z_1 = sdzF_1$ or random. Suppose we can successfully create $e(F_1, F_2)^{sdz\delta}$ (for some δ such that δF_2 is given in the instance) using elements in the instance, then the problem becomes easy to solve – just check for equality with $e(Z_1, \delta F_2)$. If they are equal then Z_1 is real; otherwise Z_1 is random. Since s and d appear in separate elements in \mathbb{G}_1 , the only possible way is to compute $e(Z_1, zF_2)$ and compare it to $e((dz - ax)F_2, szF_1)$ after cancelling out $e(F_1, F_2)^{axsz}$. But this extra term cannot be cancelled since a and x appear in separate elements of \mathbb{G}_2 . So our assumption is meaningful and there does not seem to be any way of efficiently solving A1.

Let DDH1 (resp. DDH2) be the decision Diffie-Hellman assumption in group \mathbb{G}_1 (resp. \mathbb{G}_2). It is well-known that in Type-3 setting these problems are computationally hard. The problem LW1 contains an embedded instance of DDH1. The elements sF_1 and ab^2F_1 are provided in the instance and it is required to determine whether Y_1 equals ab^2sF_1 or Y_1 is random. Similarly, LW2 contains an embedded instance of DDH2: the elements bF_2 and cF_2 are provided in the instance and it is required to determine whether Y_2 equals bcF_2 or Y_2 is random. As a result, an algorithm to solve DDH1 (resp. LW1) implies an algorithm to solve LW1 (resp. LW2) so that we can say that LW1 (resp. LW2) is no harder than DDH1 (resp. DDH2). The other direction, however, is not clear and it is due to this reason that the assumptions are considered non-standard.

Similar to the above, the problem A1 contains an embedded instance of DDH1. If $P_1 = zF_1, P_2 = zF_2$, then the elements $P_1, dP_1, sP_1, Z_1, P_2$ (present in the A1-instance) will form a proper DDH1 instance where it is required to determine whether $Z_1 = sdP_1 = sdzF_1$ or not. Hence a DDH1 solver can be used to solve A1. On the other hand, the converse is not known to hold.

2.4 Hierarchical Identity-Based Encryption

A HIBE scheme consists of five probabilistic polynomial time (in the security parameter) algorithms – Setup, Encrypt, KeyGen, Delegate and Decrypt.

- **Setup:** based on an input security parameter κ , generates and outputs the public parameters \mathcal{PP} and the master secret \mathcal{MSK} .
- **KeyGen:** inputs an identity vector \mathbf{id} and master secret \mathcal{MSK} and outputs the secret key $\mathcal{SK}_{\mathbf{id}}$ corresponding to \mathbf{id} .
- **Encrypt:** inputs an identity \mathbf{id} , a message M and returns a ciphertext \mathcal{C} .
- **Delegate:** takes as input a depth ℓ identity vector $\mathbf{id} = (\mathbf{id}_1, \dots, \mathbf{id}_\ell)$, a secret key $\mathcal{SK}_{\mathbf{id}}$ and an identity $\mathbf{id}_{\ell+1}$; returns a secret key for the identity vector $(\mathbf{id}_1, \dots, \mathbf{id}_{\ell+1})$.
- **Decrypt:** inputs a ciphertext \mathcal{C} , an identity vector \mathbf{id} , secret key $\mathcal{SK}_{\mathbf{id}}$ and returns either the corresponding message M or \perp indicating failure.

2.5 Anonymous CPA-Secure HIBE

The security game defined below captures both anonymity and security against a chosen plaintext attack for HIBE schemes. This model, which we call **ano-ind-cpa**, is equivalent to the standard security notions for CPA-security and anonymity and has been used earlier in [8,7].

Setup: The challenger runs the Setup algorithm of the HIBE and gives the public parameters to \mathcal{A} .

Phase 1: \mathcal{A} makes a number of key extraction queries adaptively. For a query on an identity vector \mathbf{id} , the challenger responds with a key $\mathcal{SK}_{\mathbf{id}}$.

Challenge: \mathcal{A} provides two message-identity pairs $(M_0, \widehat{\mathbf{id}}_0)$ and $(M_1, \widehat{\mathbf{id}}_1)$ as challenge with the restriction that neither $\widehat{\mathbf{id}}_0, \widehat{\mathbf{id}}_1$ nor any of their prefixes should have been queried in **Phase 1**. The challenger then chooses a bit β uniformly at random from $\{0, 1\}$ and returns an encryption $\widehat{\mathcal{C}}$ of M_β under the identity $\widehat{\mathbf{id}}_\beta$ to \mathcal{A} .

Phase 2: \mathcal{A} issues more key extraction queries as in **Phase 1** with the restriction that no queried identity \mathbf{id} is a prefix of either $\widehat{\mathbf{id}}_0$ or $\widehat{\mathbf{id}}_1$.

Guess: \mathcal{A} outputs a bit β' .

If $\beta = \beta'$, then \mathcal{A} wins the game. The advantage of \mathcal{A} in breaking the security of the HIBE scheme in the game **ano-ind-cpa** given by

$$\text{Adv}_{\text{HIBE}}^{\text{ano-ind-cpa}}(\mathcal{A}) = \left| \Pr[\beta = \beta'] - \frac{1}{2} \right|.$$

The HIBE scheme is said to be (ϵ, t, q) -ANO-IND-ID-CPA secure if every t -time adversary making at most q queries has $\text{Adv}_{\text{HIBE}}^{\text{ano-ind-cpa}}(\mathcal{A}) \leq \epsilon$.

3 Lewko-Waters IBE

This section reviews the asymmetric pairing-based IBE construction of Lewko-Waters [15]. The description in [15] consists of the usual ciphertexts and keys as well as the so-called semi-functional ciphertexts and keys. We use a compact notation to denote normal and semi-functional ciphertexts and keys. The group elements shown in curly brackets $\{ \}$ are the semi-functional components. To get the scheme itself, these components should be ignored.

Let $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2)$ be an asymmetric pairing. Pick $Q_1, U_1 \in \mathbb{G}_1$ and $Q_2, U_2 \in \mathbb{G}_2$ be such that $Q_2 \sim Q_1$ and $U_2 \sim U_1$. Choose $F_2 \xleftarrow{\mathbb{U}} \mathbb{G}_2^\times$, $a, v, v' \xleftarrow{\mathbb{U}} \mathbb{Z}_p$ and define $V_2 = vF_2, V'_2 = v'F_2$. Let $\tau = v + av'$ so that $\tau F_2 = V_2 + aV'_2$. Identities are elements of \mathbb{Z}_p . The public parameters and master secret are given by

$$\begin{aligned} \mathcal{PP} & : (P_1, aP_1, \tau P_1, Q_1, aQ_1, \tau Q_1, U_1, aU_1, \tau U_1, e(P_1, P_2)^\alpha) \\ \mathcal{MSK} & : (\alpha P_2, P_2, V_2, V'_2, Q_2, U_2, F_2). \end{aligned}$$

The randomisers for the ciphertext and key are s and w, r_1, r_2 respectively. These are elements of \mathbb{Z}_p . For the semi-functional components, μ, σ and γ, π are chosen at random from \mathbb{Z}_p . Elements $V'_1, F_1 \in \mathbb{G}_1$ are such that $V'_1 \sim V'_2$ and $F_1 \sim F_2$.

Ciphertext

$$\begin{aligned} C_0 & = M \cdot e(P_1, P_2)^{\alpha s} \\ C_{1,1} & = s(\text{id}Q_1 + U_1), C_{1,2} = as(\text{id}Q_1 + U_1)\{\mu\sigma F_1\}, \\ C_{1,3} & = -\tau s(\text{id}Q_1 + U_1)\{-\mu\sigma V'_1\} \\ C_{2,1} & = sP_1, C_{2,2} = asP_1\{\mu F_1\}, C_{2,3} = -\tau sP_1\{-\mu V'_1\} \end{aligned}$$

Key

$$\begin{aligned} K_{1,1} & = wP_2 + r_1V_2\{-a\gamma F_2\}, K_{1,2} = r_1V'_2\{+\gamma F_2\}, K_{1,3} = r_1F_2 \\ K_{2,1} & = \alpha P_2 + w(\text{id}Q_2 + U_2) + r_1V_2\{-a\gamma\pi F_2\}, \\ K_{2,2} & = r_2V'_2\{+\gamma\pi F_2\}, K_{2,3} = r_2F_2 \end{aligned}$$

Lewko and Waters show that this scheme is adaptively secure without random oracles under three non-standard but static assumptions – LW1, LW2 and DBDH-3. Since the elements Q_2, U_2 are in the master secret there seems to be no way to check whether a given ciphertext is encrypted to a particular identity or not. In other words, this scheme is anonymous. The proof will be similar to the anonymity proof of the HIBE scheme we describe next. A depth-1 HIBE is similar to LW-IBE except for the public parameters and master secret key. Due to space constraints we omit the proof. It can be found in the full version of this paper [17].

4 Anonymous HIBE from LW-IBE

In this section, we present our HIBE scheme, $\mathcal{LW}\text{-}\mathcal{AHIBE}$, resulting from a BBG-type extension of the LW-IBE scheme. A straightforward BBG-type extension

would lead to problems in adopting the dual system methodology. We introduce some new elements to overcome this problem. The construction is based on a Type-3 prime-order pairing with group order p . Identities are variable length tuples of elements from \mathbb{Z}_p^\times with maximum length h .

The first step towards obtaining constant-size ciphertexts is to add elements $(Q_{1,j})_{j \in [1,h]}, U_1 \in \mathbb{G}_1$ to the public parameters. These are used to create the identity hash – for an identity $\mathbf{id} = (id_1, \dots, id_\ell)$, the hash is given by $\sum_{j=1}^\ell id_j Q_{1,j} + U_1$. This replaces the hash in LW-IBE without affecting the number of elements in the ciphertext. To facilitate key extraction, the corresponding elements in \mathbb{G}_2 also are provided. We introduce some notation here: the tuple $(P_1, (Q_{1,j})_{j \in [1,h]}, U_1)$ is denoted \mathbf{Q}_1 and let its \mathbb{G}_2 counterpart be \mathbf{Q}_2 . Also present in the master secret of LW-IBE are the elements V_2, V'_2, F_2 that provide cancellation analogous to the composite order setting. In the HIBE setting, these elements along with \mathbf{Q}_2 , must be made public to assist in re-randomisation during delegation. Once these are made public, nothing is kept secret except for α . This acts as a stumbling block against a dual system proof. In a proof within the dual system framework, some secret elements are needed to create the so-called semi-functional components that are central to this proof methodology. In the composite order setting, this is achieved by keeping one subgroup hidden from the attacker which essentially forms the semi-functional space. Similarly, schemes based on dual pairing vector spaces have some vectors in the dual bases hidden that assist in generating the semi-functional space. But the strategy for HIBE extension of LW-IBE chalked out above, requires everything to be made public (except α), which in turn limits our ability to define a semi-functional space.

Our solution to this problem is to keep \mathbf{Q}_2 in the master secret. In a way, some elements of the group \mathbb{G}_2 are hidden and provide the basis for generating semi-functional components. To support delegation, suitably randomised copies of the key components are provided in the key itself. This technique was introduced by Boyen and Waters [4] to construct an anonymous HIBE scheme. V_2, V'_2, F_2 are public to help in re-randomisation during delegation; this ensures proper distribution of the delegated key. Note that \mathbf{Q}_2 contains precisely the elements required to check whether a ciphertext is encrypted to a particular identity or not. A by-product of keeping this tuple secret is anonymity. Thus our scheme is secure in the ANO-IND-ID-CPA security model (refer to Section 2.5).

Construction of LW-*AHIBE*

Setup(κ): Let h denote the maximum depth of the HIBE. Choose random generators $P_1 \in \mathbb{G}_1$ and $P_2 \in \mathbb{G}_2$; elements $Q_{1,j}, U_1 \xleftarrow{U} \mathbb{G}_1$ and $Q_{2,j}, U_2 \in \mathbb{G}_2$ such that $Q_{2,j} \sim Q_{1,j}$ for all $1 \leq j \leq h$ and $U_2 \sim U_1$. Let $F_2 \in \mathbb{G}_2$ be chosen at random and v, v' be chosen randomly from \mathbb{Z}_p . Set $V_2 = vF_2, V'_2 = v'F_2$. Pick α, a at random from \mathbb{Z}_p . Set $\tau = v + av'$ so that $\tau F_2 = V_2 + aV'_2$.

$$PP : (P_1, aP_1, \tau P_1, U_1, aU_1, \tau U_1, (Q_{1,j}, aQ_{1,j}, \tau Q_{1,j})_{j \in [1,h]}, V_2, V'_2, F_2, e(P_1, P_2)^\alpha).$$

$$MSK : (\alpha P_2, P_2, Q_{2,1}, \dots, Q_{2,h}, U_2).$$

Encrypt($M, \mathbf{id} = (\mathbf{id}_1, \dots, \mathbf{id}_\ell), \mathcal{PP}$): Choose $s \xleftarrow{\text{U}} \mathbb{Z}_p$. Let $\mathcal{H}_i(\mathbf{id}) = \mathbf{id}_1 Q_{i,1} + \dots + \mathbf{id}_\ell Q_{i,\ell} + U_i$ for $i = 1, 2$. The ciphertext is given by $\mathcal{C} = (C_0, C_{1,1}, C_{1,2}, C_{1,3}, C_{2,1}, C_{2,2}, C_{2,3})$ where the elements are computed as follows.

$$\begin{aligned} C_0 &= M \times e(P_1, P_2)^{as}, \\ C_{1,1} &= s\mathcal{H}_1(\mathbf{id}), C_{1,2} = as\mathcal{H}_1(\mathbf{id}), C_{1,3} = -\tau s\mathcal{H}_1(\mathbf{id}) \\ C_{2,1} &= sP_1, C_{2,2} = asP_1, C_{2,3} = -\tau sP_1 \end{aligned}$$

KeyGen($\mathbf{id} = (\mathbf{id}_1, \dots, \mathbf{id}_\ell), \mathcal{MSK}, \mathcal{PP}$): The key consists of $6(n - \ell + 2)$ group elements computed as follows. Choose $w_1, w_2, r_1, r_2, r_3, r_4, (z_{1,j}, z_{2,j})_{j \in [\ell+1, h]} \xleftarrow{\text{U}} \mathbb{Z}_p$.

$$\begin{aligned} K_{1,1} &= w_1 P_2 + r_1 V_2, K_{1,2} = r_1 V'_2, K_{1,3} = r_1 F_2 \\ K_{2,1} &= \alpha P_2 + w_1 \mathcal{H}_2(\mathbf{id}) + r_2 V_2, K_{2,2} = r_2 V'_2, K_{2,3} = r_2 F_2 \\ D_{j,1} &= w_1 Q_{2,j} + z_{1,j} V_2, D_{j,2} = z_{1,j} V'_2, D_{j,3} = z_{1,j} F_2 \text{ for } \ell + 1 \leq j \leq h \end{aligned}$$

$$\begin{aligned} J_{1,1} &= w_2 P_2 + r_3 V_2, J_{1,2} = r_3 V'_2, J_{1,3} = r_3 F_2 \\ J_{2,1} &= w_2 \mathcal{H}_2(\mathbf{id}) + r_4 V_2, J_{2,2} = r_4 V'_2, J_{2,3} = r_4 F_2 \\ E_{j,1} &= w_2 Q_{2,j} + z_{2,j} V_2, E_{j,2} = z_{2,j} V'_2, E_{j,3} = z_{2,j} F_2 \text{ for } \ell + 1 \leq j \leq h. \end{aligned}$$

The secret key for \mathbf{id} is given by $\mathcal{SK}_{\mathbf{id}} = (\mathcal{S}_1, \mathcal{S}_2)$, where $\mathcal{S}_1 = (K_{1,i}, K_{2,i}, D_{j,i})_{j \in [\ell+1, h], i=1,2,3}$ and $\mathcal{S}_2 = (J_{1,i}, J_{2,i}, E_{j,i})_{j \in [\ell+1, h], i=1,2,3}$. Notice that \mathcal{S}_2 -components are almost same as \mathcal{S}_1 -components except that the secret α is not embedded in \mathcal{S}_2 . The set \mathcal{S}_2 is exclusively used for re-randomisation.

Delegate($\mathbf{id} = (\mathbf{id}_1, \dots, \mathbf{id}_\ell), \mathcal{SK}_{\mathbf{id}}, \mathbf{id}_{\ell+1}, \mathcal{PP}$): Let $\mathbf{id} : \mathbf{id}_{\ell+1}$ denote the $\ell + 1$ -length identity vector $(\mathbf{id}_1, \dots, \mathbf{id}_\ell, \mathbf{id}_{\ell+1})$ obtained by appending $\mathbf{id}_{\ell+1}$ to \mathbf{id} . Choose $r'_1, r'_2, r'_3, r'_4, (z'_{1,j}, z'_{2,j})_{j \in [\ell+2, h]} \xleftarrow{\text{U}} \mathbb{Z}_p$ and $w'_1, w'_2 \xleftarrow{\text{U}} \mathbb{Z}_p^\times$. The components of the key for the identity $\mathbf{id} : \mathbf{id}_{\ell+1}$ are computed as follows.

$$\begin{aligned} K_{1,1} &\leftarrow K_{1,1} + w'_1 J_{1,1} + r'_1 V_2 \\ K_{1,2} &\leftarrow K_{1,2} + w'_1 J_{1,2} + r'_1 V'_2 \\ K_{1,3} &\leftarrow K_{1,3} + w'_1 J_{1,3} + r'_1 F_2 \end{aligned}$$

$$\begin{aligned} K_{2,1} &\leftarrow K_{2,1} + \mathbf{id}_{\ell+1} D_{\ell+1,1} + w'_1 (J_{2,1} + \mathbf{id}_{\ell+1} E_{\ell+1,1}) + r'_2 V_2 \\ K_{2,2} &\leftarrow K_{2,2} + \mathbf{id}_{\ell+1} D_{\ell+1,2} + w'_1 (J_{2,2} + \mathbf{id}_{\ell+1} E_{\ell+1,2}) + r'_2 V'_2 \\ K_{2,3} &\leftarrow K_{2,3} + \mathbf{id}_{\ell+1} D_{\ell+1,3} + w'_1 (J_{2,3} + \mathbf{id}_{\ell+1} E_{\ell+1,3}) + r'_2 F_2 \end{aligned}$$

$$\begin{aligned} J_{1,1} &\leftarrow w'_2 J_{1,1} + r'_3 V_2 & J_{2,1} &\leftarrow w'_2 (J_{2,1} + \mathbf{id}_{\ell+1} E_{\ell+1,1}) + r'_4 V_2 \\ J_{1,2} &\leftarrow w'_2 J_{1,2} + r'_3 V'_2 & J_{2,2} &\leftarrow w'_2 (J_{2,2} + \mathbf{id}_{\ell+1} E_{\ell+1,2}) + r'_4 V'_2 \\ J_{1,3} &\leftarrow w'_2 J_{1,3} + r'_3 F_2 & J_{2,3} &\leftarrow w'_2 (J_{2,3} + \mathbf{id}_{\ell+1} E_{\ell+1,3}) + r'_4 F_2 \end{aligned}$$

For $j = \ell + 2, \dots, h$,

$$\begin{aligned} D_{j,1} &\leftarrow D_{j,1} + w'_1 E_{j,1} + z'_{1,j} V_2 & E_{j,1} &\leftarrow w'_2 E_{j,1} + z'_{2,j} V_2 \\ D_{j,2} &\leftarrow D_{j,2} + w'_1 E_{j,2} + z'_{1,j} V'_2 & E_{j,2} &\leftarrow w'_2 E_{j,2} + z'_{2,j} V'_2 \\ D_{j,3} &\leftarrow D_{j,3} + w'_1 E_{j,3} + z'_{1,j} F_2 & E_{j,3} &\leftarrow w'_2 E_{j,3} + z'_{2,j} F_2 \end{aligned}$$

The above procedure essentially re-randomises all components of the key. As a result the distribution of a key obtained using delegation is the same as the distribution of a key obtained using the key generation procedure. To note the re-randomisation consider the following change of scalars for the modified key.

$$\begin{aligned} w_1 &\leftarrow w_1 + w'_1 w_2; \\ r_1 &\leftarrow r_1 + r'_1 + w'_1 r_3; \\ r_2 &\leftarrow r_2 + r'_2 + \text{id}_{\ell+1} z_{1,\ell+1} + w'_2 (r_4 + \text{id}_{\ell+1} z_{2,\ell+1}); \\ z_{1,j} &\leftarrow z_{1,j} + z'_{1,j} + w'_1 z_{2,j+1} \text{ for } j = \ell + 2, \dots, h \end{aligned}$$

$$\begin{aligned} w_2 &\leftarrow w'_2 w_2; \\ r_3 &\leftarrow w'_2 r_3 + r'_3; \\ r_4 &\leftarrow w'_2 (r_4 + \text{id}_{\ell+1} z_{2,\ell+1}) + r'_4; \\ z_{2,j} &\leftarrow w'_2 z_{2,j} + z'_{2,j} \text{ for } j = \ell + 2, \dots, h \end{aligned}$$

Due to the choice of $w'_1, w'_2, r'_1, r'_2, r'_3, r'_4, (z'_{1,j}), (z'_{2,j})$, these new randomisers are properly distributed.

Decrypt($\mathcal{C}, \text{id} = (\text{id}_1, \dots, \text{id}_\ell), \mathcal{SK}_{\text{id}}, \mathcal{PP}$): Decryption is done as follows.

$$M = C_0 \times \frac{e(C_{1,1}, K_{1,1})e(C_{1,2}, K_{1,2})e(C_{1,3}, K_{1,3})}{e(C_{2,1}, K_{2,1})e(C_{2,2}, K_{2,2})e(C_{2,3}, K_{2,3})} \tag{1}$$

Correctness of decryption of the HIBE scheme follows directly from that of LW-IBE since the decryption procedure remains the same – the additional delegation components do not play any role in decryption. Observe that computing the ratio of pairings in Equation (1) using $J_{1,i}, J_{2,i}$ ($i = 1, 2, 3$) instead of $K_{1,i}, K_{2,i}$ results in 1_T (the identity of \mathbb{G}_T). Hence decryption with \mathcal{S}_2 components provides a test for the recipient identity of a ciphertext.

5 Security of \mathcal{LW} - \mathcal{AHIBE}

We first provide some basic intuition underlying the proof with respect to different stages of security analysis (within the dual system framework), highlighting the similarities and differences with LW-IBE security proof. Then, a detailed security analysis of \mathcal{LW} - \mathcal{AHIBE} is presented in Section 5.2.

5.1 Ideas Underlying the Security Proof

The first step is to define semi-functional (sf) ciphertexts and keys. The definition of sf-ciphertext remains the same as LW-IBE. The keys of \mathcal{LW} - \mathcal{AHIBE} are significantly different from LW-IBE. We formulate the definition of sf-keys on the basis of the following observations.

- Sf-components for $(K_{1,i}, K_{2,i})_{i=1,2}$ are identical to LW-IBE since only these components participate in decryption.
- It is required to define sf-components for $(D_{j,1}, D_{j,2})_{j \in [\ell+1, h]}$ though they are only used during delegation to create the identity-hash. This is because they share the randomiser w_1 with $K_{1,i}, K_{2,i}$ and this randomiser comes from a problem instance in the reductions.

- Once sf-components are defined for \mathcal{S}_1 , it is natural to ask: is it necessary to define sf-parts for \mathcal{S}_2 ? The answer is yes since otherwise the fourth reduction fails – the simulator can test the identity to which the challenge ciphertext is encrypted by decrypting with \mathcal{S}_2 -components.

We would like to emphasise that the definition of semi-functional components (in both ciphertexts and keys), complexity assumptions and the reductions are all inter-linked. Changing the structure of sf-keys may determine the assumption required or affect simulation in some reduction. Also, for the reductions to go through, the sf-components may have to be defined in a particular way. The structure of sf-components we have is in a sense, optimal, subject to assumptions and simulations we provide.

An outline of the four main reductions in the augmented security proof (including anonymity) of LW-IBE is as follows.

First reduction: The goal of this reduction is to show that an attacker cannot distinguish between a normal ciphertext and an sf-ciphertext. It is achieved via a reduction from the LW1 problem. An LW1 instance is embedded in the challenge ciphertext attempting to exploit the adversary's ability to detect the change in order to solve the problem.

Second reduction: In this reduction, it is shown that if the adversary can decide whether the response to the k -th key extraction query is normal or semi-functional, then LW2 problem can be solved. The k -th key is constructed from an instance of LW2 problem in such a way that the key is normal if the instance is 'real' and semi-functional otherwise.

Third reduction: Here, the message that the challenge ciphertext encrypts, is changed to a random element of \mathbb{G}_T . It is shown that solving the DBDH-3 problem is no harder than distinguishing between an sf-encryption of the real message from an sf-encryption of a random element of \mathbb{G}_T .

Fourth reduction: Challenge ciphertext encrypts a random message under a random identity. The identity-hash is created using the challenge in an instance of A1 problem thus making it real or random according to the distribution of the challenge.

This strategy does not directly extend to the hierarchical setting. Several challenges/restrictions emerge as we try to prove security of $\mathcal{LW}\text{-}\mathcal{AHIBE}$.

The first and the third reductions for $\mathcal{LW}\text{-}\mathcal{AHIBE}$ are the closest to the corresponding reductions for LW-IBE appearing in [15]. In these reductions, the simulations of the public parameters; the ciphertext elements; and the components of the key which are present in LW-IBE; are exactly the same as for LW-IBE. The only technicality is to ensure that the extra components of the key can be properly simulated without changing the corresponding assumptions (LW1 for the first reduction and DBDH-3 for the third reduction).

The second reduction presents some technical novelty. Compared to the LW-IBE, the key has additional components required for delegation and re-randomisation; moreover, these have semi-functional parts. To simulate these additional components we need a modification of the game sequence.

Partial Semi-functionality: Consider the second reduction where the k -th key is made semi-functional. LW-IBE reduction embeds a pairwise independent function in the k -th key as well as the challenge ciphertext to ensure independent distribution of the scalars involved in the respective sf-components. This function is determined by the parameters used to create the identity-hash. An attempt to use the same strategy for $\mathcal{LW}\text{-}\mathcal{AHIBE}$, however, causes a problem. The reason is that the identity-hash is now present in three places – challenge ciphertext, \mathcal{S}_1 and \mathcal{S}_2 . In addition, all these have sf-components. One possible way to deal with this is to embed a 3-wise independent function i.e., a degree-2 polynomial in the identity. As result the one extra group element is required in \mathcal{PP} as well as \mathcal{MSK} . Also, encryption and key generation would each require an extra scalar multiplication and a squaring in the underlying field. The other way to get around the problem is to use two separate instances to generate the two hashes in the key. We follow the latter approach since the efficiency of the scheme remains unaffected although the degradation is increased by a factor of 2. The key is changed from normal to semi-functional in two steps – first make \mathcal{S}_1 semi-functional followed by \mathcal{S}_2 . We call a key partial semi-functional if \mathcal{S}_1 is semi-functional and \mathcal{S}_2 is normal.

The second step of the dual-system technique changes the key in the k -th response from normal to semi-functional (without the adversary noticing this). In our case, this is done in two sub-steps – the first step changes from normal to partial semi-functional and the second step changes from partial semi-functional to semi-functional. This leads to a slight degradation in the security bound by a factor of 2.

In the fourth reduction, we prove anonymity of the HIBE scheme based on the assumption A1.

5.2 Detailed Proof

As is typical in the dual-system technique, we first describe semi-functional ciphertexts and keys. These are required only in the reductions and not in the actual scheme.

Semi-functional Ciphertext: Let $C'_0, C'_{1,1}, C'_{1,2}, C'_{1,3}, C'_{2,1}, C'_{2,2}, C'_{2,3}$ be ciphertext elements normally generated by the **Encrypt** algorithm for message M and identity \mathbf{id} . Let V'_1, F_1 be elements of \mathbb{G}_1 such that $V'_1 \sim V'_2$ and $F_1 \sim F_2$. Choose $\mu, \sigma \in \mathbb{Z}_p$ at random. The semi-functional ciphertext generation algorithm will modify the normal ciphertext as: $C_0 = C'_0, C_{1,1} = C'_{1,1}, C_{2,1} = C'_{2,1}$ and

$$C_{1,2} = C'_{1,2} + \mu\sigma F_1, C_{1,3} = C'_{1,3} - \mu\sigma V'_1, C_{2,2} = C'_{2,2} + \mu F_1, C_{2,3} = C'_{2,3} - \mu V'_1.$$

Semi-functional Key: Let $(\mathcal{S}_1, \mathcal{S}_2)$ be a key generated by the **KeyGen** algorithm for identity $\mathbf{id} = (\mathbf{id}_1, \dots, \mathbf{id}_\ell)$ with $\mathcal{S}_1 = (K_{1,i}, K_{2,i}, D_{j,i})_{j \in [\ell+1, h], i=1,2,3}$, $\mathcal{S}_2 = (J_{1,i}, J_{2,i}, E_{j,i})_{j \in [\ell+1, h], i=1,2,3}$. Let $\gamma_1, \pi, \gamma_2, \eta, (\pi_j, \eta_j)_{j \in [\ell+1, h]}$ be uniform random elements chosen from \mathbb{Z}_p . The semi-functional key generation algorithm will modify the normal key as:

$$\begin{aligned}
 K_{1,1} &= K_{1,1} - a\gamma_1 F_2, & K_{1,2} &= K_{1,2} + \gamma_1 F_2, \\
 K_{2,1} &= K_{2,1} - a\gamma_1 \pi F_2, & K_{2,2} &= K_{2,2} + \gamma_1 \pi F_2, \\
 J_{1,1} &= J_{1,1} - a\gamma_2 F_2, & J_{1,2} &= J_{1,2} + \gamma_2 F_2, \\
 J_{2,1} &= J_{2,1} - a\gamma_2 \eta F_2, & J_{2,2} &= J_{2,2} + \gamma_2 \eta F_2,
 \end{aligned}$$

For $j = \ell + 1, \dots, h$

$$\begin{aligned}
 D_{j,1} &= D_{j,1} - a\gamma_1 \pi_j F_2, & D_{j,2} &= D_{j,2} + \gamma_1 \pi_j F_2, \\
 E_{j,1} &= E_{j,1} - a\gamma_2 \eta_j F_2, & E_{j,2} &= E_{j,2} + \gamma_2 \eta_j F_2.
 \end{aligned}$$

The rest of the components remain unchanged.

Partial Semi-functional Key: In a partial semi-functional key, \mathcal{S}_2 is normal and \mathcal{S}_1 is semi-functional.

Note that definitions are similar to [15] except for the delegation and re-randomisation components. Since decryption is not affected by these components of the key, all the requirements for semi-functional keys and ciphertexts are satisfied. A pair of semi-functional ciphertext and key is called *nominally semi-functional* if $\sigma = \pi$ (condition that makes decryption successful).

Structure of the Proof. We consider the security model defined in Section 2.5. The proof is organised as a hybrid over a sequence of $2q + 4$ games defined as follows.

- Game_{real}: **ano-ind-cpa** game defined in Section 2.5.
- Game_{0,1}: the challenge ciphertext is semi-functional and all the keys returned to the adversary are normal.
- Game_{k,0} (for $1 \leq k \leq q$): k -th key is partial semi-functional, the first $k - 1$ keys are semi-functional; the rest of the keys are normal.
- Game_{k,1} (for $1 \leq k \leq q$): similar to Game_{k,0} except that the k -th key is (fully) semi-functional.
- Game_{M-random}: all keys are semi-functional and the challenge ciphertext encrypts a random message to the challenge identity.
- Game_{final}: similar to Game_{M-random} except that the challenge ciphertext now encrypts to a random identity vector.

These games are ordered as Game_{real}, Game_{0,1}, Game_{1,0}, Game_{1,1}, ..., Game_{q,0}, Game_{q,1}, Game_{M-random}, Game_{final} in our hybrid argument. Let X_\square be events that \mathcal{A} wins in Game_□.

For the proof it will be convenient to use the following short-hand: denote by $h(\mathbf{id})$ the sum $\sum_{j=1}^\ell y_j \mathbf{id}_j + u$ and by $g(\mathbf{id})$ the sum $\sum_{j=1}^\ell \lambda_j \mathbf{id}_j + \nu$, where $y_1, \dots, y_n, u, \lambda_1, \dots, \lambda_n, \nu$ are elements of \mathbb{Z}_p to be chosen in the proofs.

Theorem 1. *If the $(\varepsilon_{\text{LW1}}, t')$ -LW1, $(\varepsilon_{\text{LW2}}, t')$ -LW2, $(\varepsilon_{\text{DBDH-3}}, t')$ -DBDH-3 and $(\varepsilon_{\text{A1}}, t')$ -A1 assumptions hold, then $\mathcal{LW}\text{-}\mathcal{A}H\text{IBE}$ is (ε, t, q) -ANO-IND-ID-CPA secure where*

$$\varepsilon \leq \varepsilon_{\text{LW1}} + 2q\varepsilon_{\text{LW2}} + \varepsilon_{\text{DBDH-3}} + \varepsilon_{\text{A1}}$$

and $t = t' - O(q\rho)$, where ρ is an upper bound on the time required for one scalar multiplication in \mathbb{G}_1 or \mathbb{G}_2 .

Proof. For any t -time adversary \mathcal{A} against $\mathcal{LW}\text{-}\mathcal{A}\mathcal{H}\mathcal{I}\mathcal{B}\mathcal{E}$ in the ano-ind-cpa, its advantage in winning the game is given by

$$\text{Adv}_{\mathcal{LW}\text{-}\mathcal{A}\mathcal{H}\mathcal{I}\mathcal{B}\mathcal{E}}^{\text{ano-ind-cpa}}(\mathcal{A}) = \left| \Pr[X_{\text{real}}] - \frac{1}{2} \right|.$$

We know that $\Pr[X_{\text{final}}] = \frac{1}{2}$ and hence we have

$$\begin{aligned} \text{Adv}_{\mathcal{LW}\text{-}\mathcal{A}\mathcal{H}\mathcal{I}\mathcal{B}\mathcal{E}}^{\text{ano-ind-cpa}}(\mathcal{A}) &= |\Pr[X_{\text{real}}] - \Pr[X_{\text{final}}]| \\ &\leq |\Pr[X_{\text{real}}] - \Pr[X_0]| + \sum_{k=1}^q (|\Pr[X_{k-1,1}] - \Pr[X_{k,0}]|) \\ &\quad + \sum_{k=1}^q (|\Pr[X_{k,0}] - \Pr[X_{k,1}]|) + |\Pr[X_{q,1}] - \Pr[X_{M\text{-rand}}]| \\ &\quad + |\Pr[X_{M\text{-rand}}] - \Pr[X_{\text{final}}]| \\ &\leq \varepsilon_{\text{LW1}} + 2q\varepsilon_{\text{LW2}} + \varepsilon_{\text{DBDH-3}} + \varepsilon_{\text{A1}} \end{aligned}$$

The last inequality follows from the lemmas 1, 2, 3, 4 and 5. In all the lemmas, \mathcal{A} is a t -time adversary against $\mathcal{LW}\text{-}\mathcal{A}\mathcal{H}\mathcal{I}\mathcal{B}\mathcal{E}$ and \mathcal{B} is an algorithm running in time t' that interacts with \mathcal{A} and solves one of the three problems LW1, LW2, DBDH-3 or A1. \square

Lemma 1. $|\Pr[X_{\text{real}}] - \Pr[X_{0,1}]| \leq \varepsilon_{\text{LW1}}$.

This proof is similar to that of LW-IBE and hence we omit it. See the full version [17] for details.

Lemma 2. $|\Pr[X_{k-1,1}] - \Pr[X_{k,0}]| \leq \varepsilon_{\text{LW2}}$ for $1 \leq k \leq q$.

Proof. Let $(F_1, dF_1, d^2F_1, bxF_1, dbxF_1, d^2xF_1, F_2, dF_2, bF_2, cF_2, Z_2)$ be the LW2 instance that \mathcal{B} receives. Let $Z_2 = (bc + \gamma)F_2$. \mathcal{B} 's task is to decide whether $\gamma = 0$ (Z_2 is real) or $\gamma \in_{\text{U}} \mathbb{Z}_p$ (Z_2 is random).

Set-Up: \mathcal{B} chooses $\alpha, a, y_v, y_1, \dots, y_h, u, \lambda_1, \dots, \lambda_h, \nu \xleftarrow{\text{U}} \mathbb{Z}_p$ and computes parameters as follows. $P_1 = dF_1$, $Q_{1,j} = \lambda_j(dF_1) + y_jF_1$ for $1 \leq j \leq h$, $U_1 = \nu(dF_1) + uF_1$, $V_2 = -a(bF_2) + dF_2 + y_vF_2$ and $V'_2 = bF_2$ setting $v = -ab + d + y_v$, $v' = b$ and $\tau = d + y_v$. The element τP_1 can be computed as $\tau P_1 = d^2F_1 + y_v(dF_1)$. The parameters $\tau Q_{1,j}$ for $1 \leq j \leq h$ and τU_1 are given by $\tau Q_{1,j} = \lambda_j(d^2F_1) + y_j(dF_1) + y_v\lambda_j(dF_1) + y_vy_jF_1$ and $\tau U_1 = \nu(d^2F_1) + u(dF_1) + y_v\nu(dF_1) + y_vuF_1$. The remaining parameters required to provide \mathcal{PP} to \mathcal{A} are computed using a, α and elements of the problem instance. Elements of the master secret key can also be obtained from the instance and randomisers chosen at setup.

Phases 1 and 2: The key extraction queries for identities $\text{id}_1, \dots, \text{id}_q$ are answered in the following way. If $i < k$, a semi-functional key is returned and if $i > k$ a normal key is returned. \mathcal{B} creates semi-functional keys using the master secret, a and F_2 .

For $i = k$, \mathcal{B} computes of \mathcal{S}_1 using the problem instance in the following manner. Let $\mathbf{id}_k = (\mathbf{id}_1, \dots, \mathbf{id}_\ell)$. \mathcal{B} chooses $w'_1, r'_2, z'_{1,\ell+1}, \dots, z'_{1,h} \xleftarrow{U} \mathbb{Z}_p$.

$$\begin{aligned} K_{1,1} &= w'_1 P_2 - aZ_2 + y_v(cF_2), \quad K_{1,2} = Z_2, \quad K_{1,3} = cF_2 \\ K_{2,1} &= \alpha P_2 + w'_1(g(\mathbf{id}_k)(dF_2) + h(\mathbf{id}_k)F_2) + r'_2 V_2 - ag(\mathbf{id}_k)Z_2 \\ &\quad + y_v g(\mathbf{id}_k)(cF_2) - h(\mathbf{id}_k)cF_2 \\ K_{2,2} &= r'_2 V'_2 + g(\mathbf{id}_k)Z_2, \quad K_{2,3} = r'_2 F_2 + g(\mathbf{id}_k)(cF_2) \end{aligned}$$

and for $j = \ell + 1, \dots, h$, set

$$\begin{aligned} D_{j,1} &= w'_1 Q_{2,j} + z'_{1,j} V_2 - y_j(cF_2) - a\lambda_j Z_2 + y_v \lambda_j(cF_2) \\ D_{j,2} &= z'_{1,j} V'_2 + \lambda_j Z_2, \quad D_{j,3} = z'_{1,j} F_2 + \lambda_j(cF_2) \end{aligned}$$

thus implicitly setting $w_1 = w'_1 - c$, $r_1 = c$, $r_2 = r'_2 + g(\mathbf{id}_k)c$ and $z_{1,j} = z'_{1,j} + \lambda_j c$ for $\ell + 1 \leq j \leq h$.

Let $\mathcal{S}_1 = (K_{1,i}, K_{2,i}, D_{j,i})_{j \in [\ell+1, h], i=1,2,3}$. The second set of components $\mathcal{S}_2 = (J_{1,i}, J_{2,i}, E_{j,i})_{j \in [\ell+1, h], i=1,2,3}$ is created normally. \mathcal{B} returns $\mathcal{SK}_{\mathbf{id}_k} = (\mathcal{S}_1, \mathcal{S}_2)$ as the key for \mathbf{id}_k . If $Z_2 = bcF_2$ then the key for \mathbf{id}_k is normal. It can be easily verified that the components are well-formed (for details, refer to [17]).

If $Z_2 = (bc + \gamma)F_2$ the key will be partial semi-functional with $\gamma_1 = \gamma$, $\pi = g(\mathbf{id}_k)$ and $\pi_j = \lambda_j$ for $\ell + 1 \leq j \leq h$. It is straightforward to check that $\mathcal{SK}_{\mathbf{id}_k}$ is a properly formed partial sf-key. Also, since $(\lambda_j)_{j \in [1, h]}, \nu$ are information theoretically hidden from the adversary, $\pi, (\pi_j)_{j \in [\ell+1, h]}$ are uniformly and independently distributed in \mathcal{A} 's view.

\mathcal{B} could attempt checking whether $\mathcal{SK}_{\mathbf{id}_k}$ is semi-functional by creating a sf-ciphertext for \mathbf{id}_k . Since $V'_1 = bF_1$ is not available to \mathcal{B} , the only way of doing this will lead to σ being the same as π (challenge ciphertext is created via this method). The ciphertext-key pair will be nominally semi-functional and thus provides no information to \mathcal{B} .

Challenge: \mathcal{A} provides two message-identity pairs, $(M_0, \widehat{\mathbf{id}}_0)$ and $(M_1, \widehat{\mathbf{id}}_1)$ to \mathcal{B} . It chooses $\beta \in_U \{0, 1\}$, generates the challenge ciphertext as shown below.

$$\begin{aligned} C_0 &= M_\beta \cdot e(\text{db}x F_1, dF_2)^\alpha \\ C_{1,1} &= g(\widehat{\mathbf{id}}_\beta)(\text{db}x F_1) + h(\widehat{\mathbf{id}}_\beta)(bx F_1) \\ C_{1,2} &= ag(\widehat{\mathbf{id}}_\beta)(\text{db}x F_1) + ah(\widehat{\mathbf{id}}_\beta)(bx F_1) - g(\widehat{\mathbf{id}}_\beta)(d^2 x F_1) \\ C_{1,3} &= -y_v g(\widehat{\mathbf{id}}_\beta)(\text{db}x F_1) - h(\widehat{\mathbf{id}}_\beta)(\text{db}x F_1) - y_v h(\widehat{\mathbf{id}}_\beta)(bx F_1) \\ C_{2,1} &= \text{db}x F_1, \quad C_{2,2} = a(\text{db}x F_1) - d^2 x F_1, \quad C_{2,3} = -y_v(\text{db}x F_1). \end{aligned}$$

This sets $s = bx$, $\mu = -d^2 x$ and $\sigma = g(\widehat{\mathbf{id}}_\beta)$. Since $\lambda_1, \dots, \lambda_h$ and ν are chosen uniformly at random from \mathbb{Z}_p , $\lambda_1 X_1 + \dots + \lambda_h X_h + \nu$ is a pairwise independent function for variables X_1, \dots, X_h over \mathbb{Z}_p . As a result, $\pi = \lambda_1 \mathbf{id}_1 + \dots + \lambda_\ell \mathbf{id}_\ell + \nu$ and $\sigma = \lambda_1 \widehat{\mathbf{id}}_1 + \dots + \lambda_\ell \widehat{\mathbf{id}}_\ell + \nu$ are independent and uniformly distributed. \mathcal{B} returns $\widehat{C} = (C_{1,i}, C_{2,i})_{i=1,2,3}$. Clearly, \widehat{C} is distributed properly since the individual components are well-formed [17].

Guess: \mathcal{A} returns a bit β' as its guess for β .

When the instance is real, \mathcal{B} simulates $\text{Game}_{k-1,1}$ and otherwise simulates $\text{Game}_{k,0}$. \mathcal{B} returns 1 if \mathcal{A} wins the game i.e., $\beta = \beta'$; otherwise it returns 0.

Hence, \mathcal{B} can solve the LW2 instance with advantage

$$\begin{aligned} \text{Adv}_{\mathcal{G}}^{\text{LW2}}(\mathcal{B}) &= |\Pr[\beta = \beta' | Z_2 \text{ is real}] - \Pr[\beta = \beta' | Z_2 \text{ is random}]| \\ &= |\Pr[X_{k-1,1}] - \Pr[X_{k,0}]|. \end{aligned}$$

from which the statement of the lemma follows. □

Lemma 3. $|\Pr[X_{k,0}] - \Pr[X_{k,1}]| \leq \varepsilon_{\text{LW2}}$ for $1 \leq k \leq q$.

The proof is reminiscent of Lemma 2. The reason is as follows: the structure of \mathcal{S}_2 is identical to \mathcal{S}_1 if the αP_2 term is removed from $K_{2,1}$. Moreover, the simulator chooses α and creates αP_2 independent of the instance. Hence the simulation will be similar except that the instance is now embedded in \mathcal{S}_2 and \mathcal{S}_1 is made semi-functional independent of the instance.

Lemma 4. $|\Pr[X_{q,1}] - \Pr[X_{M\text{-rand}}]| \leq \varepsilon_{\text{DBDH-3}}$.

As mentioned earlier, the third reduction closely follows the corresponding reduction in LW-IBE. The proof can be found in the full version of this paper [17].

Lemma 5. $|\Pr[X_{M\text{-rand}}] - \Pr[X_{\text{final}}]| \leq \varepsilon_{A1}$.

Proof. \mathcal{B} receives the following instance of A1 –

$$(\mathcal{G}, F_1, zF_1, dzF_1, azF_1, adzF_1, szF_1, F_2, zF_2, aF_2, xF_2, (dz - ax)F_2, Z_1).$$

Suppose $Z_1 = c \cdot sdzF_1$ and \mathcal{B} has to determine whether $c = 1$ or $c \in_{\mathbb{U}} \mathbb{Z}_p$. The game is simulated as follows.

Set-Up: Pick $\alpha, v, v', y_1, \dots, y_h, u \xleftarrow{\mathbb{U}} \mathbb{Z}_p$ and set the parameters as

$$\begin{aligned} P_1 &= zF_1, V_2 = vF_2, V'_2 = v'F_2, Q_{1,j} = y_j(dzF_1), U_1 = u(dzF_1), \\ aP_1 &= azP_1, aQ_{1,j} = y_j(adzF_1), aU_1 = u(adzF_1), \end{aligned}$$

where $j = 1, \dots, h$ and similarly the elements $\tau P_1, \tau Q_{1,j}$ and τU_1 . Compute $e(P_1, P_2)^\alpha = e(zF_1, zF_2)^\alpha$. \mathcal{B} returns \mathcal{PP} to \mathcal{A} . \mathcal{B} knows $P_2 = zF_2$ and α but not $Q_{2,j}$'s and U_2 . The main idea is to mask the components required to create identity-hash in \mathbb{G}_2 by a scalar multiple of aF_2 so that only semi-functional keys can be created.

Key Extraction Phases 1 and 2: \mathcal{B} computes the key for the i -th identity vector $\text{id}_i = (\text{id}_1, \dots, \text{id}_\ell)$ as follows.

$$\begin{aligned} w_1, w_2, r_1, r_2, r_3, r_4, (z_{1,j}, z_{2,j})_{j=1}^h &\xleftarrow{\mathbb{U}} \mathbb{Z}_p, \gamma_1, \gamma_2 \xleftarrow{\mathbb{U}} \mathbb{Z}_p^\times, \\ \pi', (\pi'_j)_{j=1}^h, \eta', (\eta'_j)_{j=1}^h &\xleftarrow{\mathbb{U}} \mathbb{Z}_p, \\ K_{1,1} &= w_1(zF_2) + r_1V_2 - \gamma_1aF_2, K_{1,2} = r_1V'_2 + \gamma_1F_2, K_{1,3} = r_1F_2 \\ K_{2,1} &= \alpha zF_2 + w_1h(\text{id}_i)(dz - ax)F_2 + r_2V_2 - \gamma_1\pi'(aF_2), \\ K_{2,2} &= r_2V'_2 + w_1h(\text{id}_i)xF_2 + \gamma_1\pi'F_2, K_{2,3} = r_2F_2, \end{aligned}$$

$$\begin{aligned}
J_{1,1} &= w_2(zF_2) + r_3V_2 - \gamma_2aF_2, \quad J_{1,2} = r_3V_2' + \gamma_2F_2, \quad J_{1,3} = r_3F_2 \\
J_{2,1} &= w_2h(\mathbf{id}_i)(dz - ax)F_2 + r_4V_2 - \gamma_2\eta'(aF_2), \\
J_{2,2} &= r_4V_2' + w_2h(\mathbf{id}_i)xF_2 + \gamma_2\eta'F_2, \quad J_{2,3} = r_4F_2,
\end{aligned}$$

For $\ell + 1 \leq j \leq h$,

$$\begin{aligned}
D_{j,1} &= w_1y_j(dz - ax)F_2 + z_{1,j}V_2 - \gamma_1\pi_j'(aF_2), \\
D_{j,2} &= z_{1,j}V_2' + w_1y_j(xF_2) + \gamma_1\pi_j'F_2, \quad D_{j,3} = z_{1,j}F_2 \\
E_{j,1} &= w_2y_j(dz - ax)F_2 + z_{2,j}V_2 - \gamma_2\eta_j'(aF_2), \\
E_{j,2} &= z_{2,j}V_2' + w_2y_j(xF_2) + \gamma_2\eta_j'F_2, \quad E_{j,3} = z_{2,j}F_2
\end{aligned}$$

setting $\pi = \pi' + \gamma_1^{-1}w_1h(\mathbf{id}_i)x$, $\pi_j = \pi_j' + \gamma_1^{-1}w_1y_jx$, $\eta = \eta' + \gamma_2^{-1}w_2h(\mathbf{id}_i)x$ and $\eta_j = \eta_j' + \gamma_2^{-1}w_2y_jx$. Since all these scalars are additively randomised they remain properly distributed in the adversary's view. We show that $D_{j,1}, D_{j,2}$ are well-formed; the rest can be verified in a similar fashion.

$$\begin{aligned}
D_{j,1} &= w_1y_j(dz - ax)F_2 + z_{1,j}V_2 - \gamma_1\pi_j'(aF_2) \\
&= w_1y_jdzF_2 - w_1y_jaxF_2 + z_{1,j}V_2 - \gamma_1(\pi_j - \gamma_1^{-1}w_1y_jx)(aF_2) \\
&= w_1y_jdzF_2 - w_1y_jaxF_2 + z_{1,j}V_2 - a\gamma_1\pi_jF_2 + w_1y_jaxF_2 \\
&= w_1y_jdzF_2 + z_{1,j}V_2 - a\gamma_1\pi_jF_2 \\
D_{j,2} &= z_{1,j}V_2' + w_1y_j(xF_2) + \gamma_1\pi_j'F_2 \\
&= z_{1,j}V_2' + w_1y_j(xF_2) + \gamma_1(\pi_j - \gamma_1^{-1}w_1y_jx)F_2 \\
&= z_{1,j}V_2' + w_1y_jxF_2 + \gamma_1\pi_jF_2 - w_1y_jxF_2 \\
&= z_{1,j}V_2' + \gamma_1\pi_jF_2
\end{aligned}$$

Challenge: \mathcal{B} receives two pairs of messages and identity vectors $(M_0, \widehat{\mathbf{id}}_0)$ and $(M_1, \widehat{\mathbf{id}}_1)$ from \mathcal{A} . It chooses $\beta \xleftarrow{\mathcal{U}} \{0, 1\}$ and $a', \xi \xleftarrow{\mathcal{U}} \mathbb{Z}_p$ at random and generates a semi-functional challenge ciphertext as follows.

$$\begin{aligned}
C_0 &\xleftarrow{\mathcal{U}} \mathbb{G}_T \\
C_{1,1} &= h(\widehat{\mathbf{id}}_\beta)Z_1, \quad C_{1,2} = a'h(\widehat{\mathbf{id}}_\beta)Z_1 + \xi F_1, \\
C_{1,3} &= -vh(\widehat{\mathbf{id}}_\beta)Z_1 - v'a'h(\widehat{\mathbf{id}}_\beta)Z_1 - v'\xi F_1, \\
C_{2,1} &= szF_1, \quad C_{2,2} = a'szF_1, \quad C_{2,3} = -v(szF_1) - v'a'(szF_1),
\end{aligned}$$

where $a' = a + \mu'$, $\mu = \mu'sz$ and $\xi = \mu\sigma'$. The challenge ciphertext $\widehat{C} = (C_0, C_{1,1}, C_{1,2}, C_{1,3}, C_{2,1}, C_{2,2}, C_{2,3})$ is returned to \mathcal{A} . The computation below illustrates that \widehat{C} is a semi-functional encryption with $\sigma = \sigma' + cdh(\widehat{\mathbf{id}}_\beta)$.

$$\begin{aligned}
C_{1,2} &= a'h(\widehat{\mathbf{id}}_\beta)Z_1 + \xi F_1 \\
&= (a + \mu')h(\widehat{\mathbf{id}}_\beta)csdzF_1 + \mu\sigma'F_1 \\
&= ah(\widehat{\mathbf{id}}_\beta)csdzF_1 + \mu'h(\widehat{\mathbf{id}}_\beta)csdzF_1 + \mu\sigma'F_1 \\
&= as\mathcal{H}_1(\widehat{\mathbf{id}}_\beta) + (\mu'sz)(cdh(\widehat{\mathbf{id}}_\beta))F_1 + \mu\sigma'F_1 \\
&= as\mathcal{H}_1(\widehat{\mathbf{id}}_\beta) + \mu(cdh(\widehat{\mathbf{id}}_\beta))F_1 + \mu\sigma'F_1 \\
&= as\mathcal{H}_1(\widehat{\mathbf{id}}_\beta) + \mu\sigma F_1
\end{aligned}$$

Observe that $C_{1,1} = s\mathcal{H}_1(\widehat{\mathbf{id}}_\beta) = (c \cdot h(\widehat{\mathbf{id}}_\beta))(sdzF_1)$. If $c = 1$, then $\sigma = \sigma' + dh(\widehat{\mathbf{id}}_\beta)$ and \widehat{C} is encrypted under $\widehat{\mathbf{id}}_\beta$. Otherwise, c is random, causing $h(\widehat{\mathbf{id}}_\beta)$ and consequently the target identity and σ to be random quantities.

Guess: \mathcal{A} returns its guess β' of β .

If the algorithm \mathcal{B} returns 1 when $\beta = \beta'$ and 0 otherwise, it can solve the A1 instance with advantage

$$\begin{aligned} \text{Adv}_{\widehat{G}}^{\text{A1}}(\mathcal{B}) &= |\Pr[\beta = \beta' | Z_1 \text{ is real}] - \Pr[\beta = \beta' | Z_1 \text{ is random}]| \\ &= |\Pr[X_{M\text{-rand}}] - \Pr[X_{\text{final}}]|. \end{aligned}$$

□

6 Conclusion

We have extended the Lewko-Waters IBE scheme using asymmetric pairings to a constant-size ciphertext HIBE. In addition to CPA-security the HIBE scheme possesses anonymity. Security is based on the assumptions LW1, LW2, DBDH-3 and a new assumption A1 that we introduce. This HIBE is the first example of an anonymous, adaptive-id secure, constant-size ciphertext HIBE which can be instantiated using Type-3 pairings. The assumptions used are static but non-standard. It would be interesting to explore constructions that obtain security under standard assumptions.

Note

A recent work by Lee, Park and Lee [13] proposes a construction identical to ours. Their proof of anonymity, however, relies on different assumptions namely, SXDH and asymmetric 3-party Diffie-Hellman which is a non-standard assumption (while our proof is based on A1). We would like to mention that this appeared in DCC August 2013 issue and was made publicly available after we submitted to IMACC 2013.

Acknowledgement. We thank the reviewers of IMACC 2013 for providing useful comments.

References

1. Abdalla, M., et al.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. 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); full version available at Cryptology ePrint Archive; Report 2005/015
3. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3), 586–615 (2003); earlier version appeared in the proceedings of CRYPTO 2001 (2001)

4. 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)
5. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings – the role of ψ revisited. *Discrete Applied Mathematics* 159(13), 1311–1322 (2011)
6. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) *Cryptography and Coding 2001*. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
7. De Caro, A., Iovino, V., Persiano, G.: Fully secure anonymous HIBE and secret-key anonymous IBE with short ciphertexts. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) *Pairing 2010*. LNCS, vol. 6487, pp. 347–366. Springer, Heidelberg (2010)
8. Ducas, L.: Anonymity from asymmetry: New constructions for anonymous HIBE. In: Pieprzyk, J. (ed.) *CT-RSA 2010*. LNCS, vol. 5985, pp. 148–164. Springer, Heidelberg (2010)
9. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert, H. (ed.) *EUROCRYPT 2010*. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
10. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. *Discrete Applied Mathematics* 156(16), 3113–3121 (2008)
11. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) *ASIACRYPT 2002*. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
12. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) *EUROCRYPT 2002*. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
13. Lee, K., Park, J.H., Lee, D.H.: Anonymous hibe with short ciphertexts: full security in prime order groups. In: *Designs, Codes and Cryptography*, pp. 1–31 (2013)
14. Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) *EUROCRYPT 2012*. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
15. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) *TCC 2010*. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
16. Park, J.H., Lee, D.H.: Anonymous hibe: Compact construction over prime-order groups. *IEEE Transactions on Information Theory* 59(4), 2531–2541 (2013)
17. Ramanna, S.C., Sarkar, P.: Anonymous constant-size ciphertext HIBE from asymmetric pairings. *Cryptology ePrint Archive*, Report 2012/057 (2012), <http://eprint.iacr.org/>
18. Seo, J.H., Kobayashi, T., Ohkubo, M., Suzuki, K.: Anonymous hierarchical identity-based encryption with constant size ciphertexts. In: Jarecki, S., Tsudik, G. (eds.) *PKC 2009*. LNCS, vol. 5443, pp. 215–234. Springer, Heidelberg (2009)
19. 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)
20. Smart, N.P., Vercauteren, F.: On computable isomorphisms in efficient asymmetric pairing-based systems. *Discrete Applied Mathematics* 155(4), 538–547 (2007)
21. 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)

Author Index

- Aguilar-Melchor, Carlos 99
Alwen, Joël 65
- Barbosa, Manuel 65
Bettaieb, Slim 99
Bos, Joppe W. 45
- Castagnos, Guilhem 193
Clark, John A. 120
Cohen, Gérard D. 85
- Damgård, Ivan 270
- Farshim, Pooya 65
Fouque, Pierre-Alain 158, 252
- Gaborit, Philippe 99
Galindo, David 173
Gardner, David 16
Gennaro, Rosario 65
Ghadafi, Essam M. 304
Gordon, S. Dov 65
- Hirose, Shoichi 213
- Jakobsen, Thomas P. 270
- Karpman, Pierre 252
Kölbl, Stefan 141
Kuwakado, Hidenori 213
- Lauter, Kristin 45
Loftus, Jake 45
- McLaughlin, James 120
Mella, Silvia 28
Mendel, Florian 141
Mennink, Bart 232
Mesnager, Sihem 1, 85
- Nad, Tomislav 141
Naehrig, Michael 45
Nakano, Rintaro 324
Nielsen, Jesper Buus 270
- Pagter, Jakob I. 270
Patey, Alain 85
Phan, Raphael C.-W. 16
Pol, Joop van de 290
- Ramanna, Somindu C. 344
Renner, Soline 193
- Sălăgean, Ana 16
Sarkar, Palash 344
Schläffer, Martin 141
Schrek, Julien 99
Shikata, Junji 324
Smart, Nigel P. 290
Susella, Ruggero 28
- Tessaro, Stefano 65
Tibouchi, Mehdi 158
- Vivek, Srinivas 173
- Wilson, David A. 65
- Zapalowicz, Jean-Christophe 158
Zémor, Gilles 193