

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 of Computer Science, Saarbruecken, Germany

Matthew G. Parker (Ed.)

Cryptography and Coding

12th IMA International Conference
Cryptography and Coding 2009, Cirencester, UK
December 15-17, 2009, Proceedings



Springer

Volume Editor

Matthew G. Parker
University of Bergen
The Selmer Centre
Department of Informatics
P.O. Box 7800
N-5020, Bergen, Norway
E-mail: matthew.parker@ii.uib.no

Library of Congress Control Number: 2009939840

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743

ISBN-10 3-642-10867-9 Springer Berlin Heidelberg New York

ISBN-13 978-3-642-10867-9 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2009
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12807253 06/3180 5 4 3 2 1 0

Preface

The 12th in the series of IMA Conferences on Cryptography and Coding was held at the Royal Agricultural College, Cirencester, December 15–17, 2009. The program comprised 3 invited talks and 26 contributed talks. The contributed talks were chosen by a thorough reviewing process from 53 submissions. Of the invited and contributed talks, 28 are represented as papers in this volume. These papers are grouped loosely under the headings: Coding Theory, Symmetric Cryptography, Security Protocols, Asymmetric Cryptography, Boolean Functions, and Side Channels and Implementations.

Numerous people helped to make this conference a success. To begin with I would like to thank all members of the Technical Program Committee who put a great deal of effort into the reviewing process so as to ensure a high-quality program. Moreover, I wish to thank a number of people, external to the committee, who also contributed reviews on the submitted papers. Thanks, of course, must also go to all authors who submitted papers to the conference, both those rejected and accepted. The review process was also greatly facilitated by the use of the Web-submission-and-review software, written by Shai Halevi of IBM Research, and I would like to thank him for making this package available to the community.

The invited talks were given by Frank Kschischang, Ronald Cramer, and Alexander Pott, and two of these invited talks appear as papers in this volume. A particular thanks goes to these invited speakers, each of whom is well-known, not only for being a world-leader in their field, but also for their particular ability to communicate their expertise in an enjoyable and stimulating manner.

I would like to thank Amy Marsh and all those at the IMA. In particular I would like to thank Amy for her cheerful efficiency in dealing with the many administrative aspects of the conference and gently reminding me of things that needed to be done. It was a pleasure to work with her. We are grateful for the sponsorship provided by Vodafone for the conference, and I would also like to thank all at Springer for their hard work in publishing these proceedings.

December 2009

Matthew G. Parker

Cryptography and Coding

12th IMA International Conference Proceedings

Royal Agricultural College, Cirencester, UK
December 15–17, 2009

Program Committee

Steve Babbage	Vodafone Group Services Ltd.
Mohammed Benissa	University of Sheffield, UK
Pascale Charpin	INRIA Rocquencourt, Paris, France
Liqun Chen	Hewlett-Packard, USA
Carlos Cid	Royal Holloway, University of London, UK
Tuvi Etzion	Technion, Israel
Bahram Honary	Lancaster University, UK
Jon-Lark Kim	University of Louisville, USA
Gohar Kyureghyan	University of Magdeburg, Germany
Gary McGuire	University College Dublin, Ireland
Alfred Menezes	University of Waterloo, Canada
David Naccache	Ecole Normale Supérieure, France
Matthew G. Parker	University of Bergen, Norway (Program Chair)
Matt Robshaw	Orange Labs, Paris, France
Ana Salagean	Loughborough University, UK
Hans Georg Schaathun	University of Surrey, UK
Michael Scott	Dublin City University, Ireland
Amin Shokrollahi	EPFL, Lausanne, Switzerland
Nigel Smart	University of Bristol, UK
Patrick Solé	Telecom ParisTech, Paris, France
Frederik Vercauteren	K. U. Leuven, Belgium
Guilin Wang	University of Birmingham, UK
Kyeongcheol Yang	Pohang University of Science and Technology, South Korea
Gilles Zemor	University of Bordeaux, France

Steering Committee

Bahram Honary	Lancaster University, UK
Chris Mitchell	Royal Holloway, University of London, UK
Kenny Paterson	Royal Holloway, University of London, UK
Fred Piper	Royal Holloway, University of London, UK
Nigel Smart	University of Bristol, UK
Steven Galbraith	Auckland University, New Zealand

External Reviewers

Anne Canteaut	Atefeh Mashatan	Gregory Neven
Martin Albrecht	Pierrick Gaudry	Elisabeth Oswald
Karim Belabas	Faruk Goeloglu	Christophe Petit
Nick Bone	Yun Kyung Han	Thomas Peyrin
Sebastien Canard	Florian Hess	Eric Schost
Anne Canteaut	Christine Kelley	Yannick Seurin
Srdjan Capkun	Kyung-Joong Kim	Deian Stefan
Guilhem Castagnos	Markulf Kohlweiss	Michael Vielhaber
Wouter Castryck	Francoise Levy-dit-Vehel	Pascal Vontobel
Herv Chabanne	Fagen Li	Bogdan Warinschi
Jin-Ho Chung	Carlos Aguilar Melchor	Vitaly Skatchek
Yi Deng		

Table of Contents

Coding Theory

Subspace Codes	1
<i>Azadeh Khaleghi, Danilo Silva, and Frank R. Kschischang</i>	
On Linear Programming Decoding on a Quantized Additive White Gaussian Noise Channel	22
<i>Eirik Rosnes</i>	
Codes as Modules over Skew Polynomial Rings	38
<i>Delphine Boucher and Felix Ulmer</i>	
On Higher Weights and Code Existence	56
<i>Hans Georg Schaathun</i>	
Mass Formula for Even Codes over \mathbb{Z}_8	65
<i>Koichi Betsumiya, Rowena Alma L. Betty, and Akihiro Munemasa</i>	
On the Classification of Self-dual \mathbb{Z}_k -Codes	78
<i>Masaaki Harada and Akihiro Munemasa</i>	
On Linear Codes from Maximal Curves	91
<i>Stefania Fanali</i>	

Symmetric Cryptography

On Linear Cryptanalysis with Many Linear Approximations	112
<i>Benoît Gérard and Jean-Pierre Tillich</i>	
Bivium as a Mixed-Integer Linear Programming Problem	133
<i>Julia Borghoff, Lars R. Knudsen, and Mathias Stolpe</i>	
Security of Cyclic Double Block Length Hash Functions	153
<i>Ewan Fleischmann, Michael Gorski, and Stefan Lucks</i>	
Another Glance at Double-Length Hashing	176
<i>Onur Özen and Martijn Stam</i>	
Geometric Ideas for Cryptographic Equation Solving in Even Characteristic	202
<i>Sean Murphy and Maura B. Paterson</i>	

Security Protocols

Provably Secure Code-Based Threshold Ring Signatures	222
<i>Léonard Dallot and Damien Vergnaud</i>	
A New Protocol for the Nearby Friend Problem	236
<i>Sanjit Chatterjee, Koray Karabina, and Alfred Menezes</i>	
Distributing the Key Distribution Centre in Sakai–Kasahara Based Systems	252
<i>Martin Geisler and Nigel P. Smart</i>	
Key Predistribution Schemes and One-Time Broadcast Encryption Schemes from Algebraic Geometry Codes	263
<i>Hao Chen, San Ling, Carles Padró, Huaxiong Wang, and Chaoping Xing</i>	
Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes	278
<i>Nuttapong Attrapadung and Hideki Imai</i>	
Certificate-Free Attribute Authentication	301
<i>Dalia Khader, Liqun Chen, and James H. Davenport</i>	

Asymmetric Cryptography

Comparing with RSA	326
<i>Julien Cathalo, David Naccache, and Jean-Jacques Quisquater</i>	
Double-Exponentiation in Factor-4 Groups and Its Applications	336
<i>Koray Karabina</i>	
Oracle-Assisted Static Diffie-Hellman Is Easier Than Discrete Logarithms	351
<i>Antoine Joux, Reynald Lercier, David Naccache, and Emmanuel Thomé</i>	
An Improvement to the Gaudry-Schost Algorithm for Multidimensional Discrete Logarithm Problems	368
<i>Steven Galbraith and Raminder S. Ruprai</i>	

Boolean Functions

On Designs and Multiplier Groups Constructed from Almost Perfect Nonlinear Functions	383
<i>Yves Edel and Alexander Pott</i>	
A New Family of Hyper-Bent Boolean Functions in Polynomial Form	402
<i>Sihem Mesnager</i>	

The Rayleigh Quotient of Bent Functions	418
<i>Lars Eirik Danielsen, Matthew G. Parker, and Patrick Solé</i>	
Side Channels and Implementations	
Cache Timing Analysis of LFSR-Based Stream Ciphers	433
<i>Gregor Leander, Erik Zenner, and Philip Hawkes</i>	
Optimal Recovery of Secret Keys from Weak Side Channel Traces	446
<i>Werner Schindler and Colin D. Walter</i>	
Practical Zero-Knowledge Proofs for Circuit Evaluation	469
<i>Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi</i>	
Author Index	495

Subspace Codes

Azadeh Khaleghi, Danilo Silva, and Frank R. Kschischang

Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario M5S 3G4, Canada
{azalea,danilo,frank}@comm.utoronto.ca

Abstract. This paper is a survey of bounds and constructions for subspace codes designed for the injection metric, a distance measure that arises in the context of correcting adversarial packet insertions in linear network coding. The construction of lifted rank-metric codes is reviewed, along with improved constructions leading to codes with strictly more codewords. Algorithms for encoding and decoding are also briefly described.

1 Introduction

Let \mathbb{F}_q be the finite field of size q , and let \mathbb{F}_q^n denote the vector space of n -tuples over \mathbb{F}_q . The set of all subspaces of \mathbb{F}_q^n , called the *projective space* of order n over \mathbb{F}_q , is denoted $\mathcal{P}_q(n)$. The set of all k -dimensional subspaces of \mathbb{F}_q^n , called a *Grassmannian*, is denoted $\mathcal{G}_q(n, k)$, where $0 \leq k \leq n$. Obviously $\mathcal{P}_q(n) = \bigcup_{k=0}^n \mathcal{G}_q(n, k)$.

A (*subspace*) code \mathcal{C} is a nonempty collection of subspaces of \mathbb{F}_q^n , i.e., a nonempty subset of $\mathcal{P}_q(n)$. Unlike classical coding theory, where each codeword is a vector, here each codeword of \mathcal{C} is itself an entire space of vectors. A code in which each codeword has the same dimension, i.e., a code contained within a single Grassmannian, is called a *constant-dimension* code.

As in classical coding theory, it is important to define a distance measure between codewords. One possible distance measure between two spaces U and V in $\mathcal{P}_q(n)$ —though not the metric of main interest in this paper—is the so-called *subspace metric*

$$d_S(U, V) \triangleq \dim(U) + \dim(V) - 2\dim(U \cap V),$$

introduced in the context of error- and erasure-correction in linear network coding [1]. The measure that will be of main interest here, however, is the *injection distance* $d(U, V)$, introduced in the later paper [2], and given by

$$d(U, V) \triangleq \max\{\dim(U), \dim(V)\} - \dim(U \cap V).$$

This function is indeed a metric on $\mathcal{P}_q(n)$ [2]. The injection distance and the subspace distance are closely related, as

$$d(U, V) = \frac{1}{2}d_S(U, V) + \frac{1}{2}|\dim(V) - \dim(U)|, \quad \forall U, V \in \mathcal{P}_q(n). \quad (1)$$

In fact, the two metrics are equivalent when U and V have the same dimension, i.e., if $\dim(U) = \dim(V)$ then $d_S(U, V) = 2d(U, V)$. Denote by $U + V$ the sum of U and V , i.e., let $U + V = \{u + v : u \in U, v \in V\}$. The relation $\dim(U + V) = \dim(U) + \dim(V) - \dim(U \cap V)$ gives the alternative expressions

$$\begin{aligned} d(U, V) &= \dim(U + V) - \min\{\dim(U), \dim(V)\} \text{ and} \\ d_S(U, V) &= 2\dim(U + V) - \dim(U) - \dim(V) \\ &= \dim(U + V) - \dim(U \cap V) \end{aligned}$$

for two metrics.

The minimum distance between distinct codewords in a code \mathcal{C} is denoted as $d(\mathcal{C})$ if the injection metric is used and as $d_S(\mathcal{C})$ if the subspace metric is used, i.e.,

$$d(\mathcal{C}) \triangleq \min_{U, V \in \mathcal{C}: U \neq V} d(U, V) \text{ and } d_S(\mathcal{C}) \triangleq \min_{U, V \in \mathcal{C}: U \neq V} d_S(U, V).$$

It follows from (1) that

$$d(\mathcal{C}) \geq \frac{1}{2}d_S(\mathcal{C}), \quad (2)$$

with equality if (but not only if) \mathcal{C} is a constant dimension code.

A code $\mathcal{C} \subseteq \mathcal{P}_q(n)$ is called an $(n, d)_q$ code if $d(\mathcal{C}) = d$, and is called an $(n, d, k)_q$ code if, additionally, $\mathcal{C} \subseteq \mathcal{G}_q(n, k)$. Similarly, \mathcal{C} is called an $(n, d)_q^S$ code if $d_S(\mathcal{C}) = d$. The latter notation follows the convention, used throughout this paper, that if a concept is defined for the injection metric, then the analogous concept for the subspace metric is denoted by a superscript S. We will, however, have no occasion to refer to an $(n, d, k)_q^S$ code, since such a code is an $(n, d/2, k)_q$ code. We denote by $A_q(n, d)$ and $A_q(n, d, k)$ the sizes of a largest $(n, d)_q$ code and a largest $(n, d, k)_q$ code, respectively.

Subspace codes turn out to be the natural objects in several applications, such as noncoherent linear network coding [1, 2, 3, 4, 5] and linear authentication [6, 7]. For linear authentication, it is shown in [6, Theorem 4.1] that every $(n, d, k)_q$ code \mathcal{C} is an $[n, |\mathcal{C}|, n - k, d]$ linear authentication code over \mathbb{F}_q , and vice-versa. For network coding, it is shown in [2, Theorem 20] that an $(n, d)_q$ code can correct any t corrupt packets injected in a noncoherent linear network coding system with rank deficiency ρ if and only if $d > 2t + \rho$. Thus, the packet-error correction capability of a subspace code for network coding is completely characterized in terms of the injection distance. Historically, the subspace distance appeared earlier in this context [1], but it can only provide a correction guarantee (not the converse), which can be seen from (2).

This paper surveys the existing literature on constructions of $(n, d)_q$ and $(n, d, k)_q$ codes, as well as upper and lower bounds on $A_q(n, d)$ and $A_q(n, d, k)$. Usually, results for general subspace codes are based on previous results for constant-dimension codes. In view of (2), results for the subspace metric may also be useful and are reviewed as well.

The remainder of the paper is organized as follows. Section 2 establishes some useful notation and reviews properties of rank metric codes. Section 3 discusses

bounds on $A_q(n, d)$, $A_q(n, d, k)$ and $A_q^S(n, d)$. Section 4 reviews existing constructions of general and constant-dimension subspace codes. Section 5 briefly describes encoding and decoding methods for subspace codes. The paper ends in Section 6 with some concluding remarks and a list of open problems.

2 Preliminaries

2.1 Notation and Basic Facts

Let $\mathbb{N} = \{0, 1, 2, \dots\}$. If \mathcal{A} is a finite set, let $|\mathcal{A}|$ denote its cardinality.

We will often need to refer to vectors and matrices with components from \mathbb{F}_q . If $v = (v_1, \dots, v_n)$ is a vector \mathbb{F}_q^n , let $\text{supp}(v) = \{i \in \{1, \dots, n\} : v_i \neq 0\}$ denote its support and let $\text{wt}(v) = |\text{supp}(v)|$ denote its Hamming weight.

Let $\mathbb{F}_q^{m \times n}$ denote the set of all $m \times n$ matrices over \mathbb{F}_q . For concreteness, a vector in \mathbb{F}_q^n will be considered as an element of $\mathbb{F}_q^{1 \times n}$, i.e., as a row vector. The $m \times n$ all-zero matrix and the $n \times n$ identity matrix are denoted by $\mathbf{0}_{m \times n}$ and $I_{n \times n}$, respectively, where the subscripts may be omitted when there is no risk of confusion.

Let $X \in \mathbb{F}_q^{m \times n}$ be an $m \times n$ matrix. If \mathcal{S} is a nonempty subset of $\{1, \dots, m\}$, then $X_{\mathcal{S}}$ is the submatrix of X consisting of the rows indexed by \mathcal{S} (in increasing order). If X is nonzero, then its reduced row echelon form (RREF) is denoted as $\text{rref}(X)$. Associated with a nonzero X is a vector $\text{prof}(X) \in \{0, 1\}^n$, called the *profile vector* of X , in which $\text{supp}(\text{prof}(X))$ is the set of column positions of the leading ones in the rows of $\text{rref}(X)$. If $X = \mathbf{0}$, then we set $\text{prof}(X)$ to the zero vector.

The row space of a matrix X is denoted as $\langle X \rangle$. If $X \in \mathbb{F}_q^{m \times n}$ then $\langle X \rangle \in \mathcal{P}_q(n)$. The rank of X is denoted as $\text{rank}(X)$ and, of course, $\text{rank}(X) = \dim(\langle X \rangle)$. More generally, if $X \in \mathbb{F}_q^{n \times m}$ and $Y \in \mathbb{F}_q^{N \times m}$, then

$$\left\langle \begin{bmatrix} X \\ Y \end{bmatrix} \right\rangle = \langle X \rangle + \langle Y \rangle;$$

therefore,

$$\text{rank} \begin{bmatrix} X \\ Y \end{bmatrix} = \dim(\langle X \rangle + \langle Y \rangle).$$

Note that $\text{wt}(\text{prof}(X)) = \text{rank}(X)$.

Associated with a vector space $U \in \mathcal{G}_q(n, k)$, $k > 0$, is a unique $k \times n$ matrix X_U in RREF (i.e., with $X_U = \text{rref}(X_U)$) having the property that $\langle X_U \rangle = U$. With a slight abuse of notation we extend the `prof` function to vector spaces by defining

$$\text{prof}(U) \triangleq \text{prof}(X_U),$$

where $\text{prof}(U)$ is the zero vector if $\dim(U) = 0$. Given any binary profile vector $b \in \{0, 1\}^n$, the so-called *Schubert cell* [8] in $\mathcal{P}_q(n)$ corresponding to b is the set

$$\mathcal{S}_q(b) = \text{prof}^{-1}(b) = \{U \in \mathcal{P}_q(n) : \text{prof}(U) = b\}.$$

If $\text{wt}(b) = k$, then $\mathcal{S}_q(b) \subseteq \mathcal{G}_q(n, k)$. Thus binary profile vectors (in general) induce a *partition* of $\mathcal{P}_q(n)$ into 2^n distinct Schubert cells, while binary profile vectors of weight k (in particular) induce a partition of $\mathcal{G}_q(n, k)$ into $\binom{n}{k}$ Schubert cells. These partitions will become useful in Section 4.

Associated with $\mathcal{G}_q(n, k)$ is a distance-regular graph (called a Grassmann graph) whose vertices correspond to the elements of $\mathcal{G}_q(n, k)$ and where two vertices are adjacent if the corresponding subspaces intersect in a space of dimension $k - 1$ [9]. The Grassmannian $\mathcal{G}_q(n, k)$ also forms an association scheme, the so-called q -Johnson scheme [10, Ch. 30], in which two spaces are i th associates if they intersect in a space of dimension $k - i$, or, equivalently, if they are separated by graph distance i in the Grassmann graph. When restricted to $\mathcal{G}_q(n, k)$, the injection distance $d(\cdot, \cdot)$ corresponds to the graph distance in the corresponding Grassmann graph.

It is well known that the cardinality of the Grassmannian $\mathcal{G}_q(n, k)$ is given by the *Gaussian coefficient*

$$\begin{bmatrix} n \\ k \end{bmatrix}_q = \prod_{i=0}^{k-1} \frac{(q^n - q^i)}{(q^k - q^i)}.$$

The subscript q will be omitted when there is no possibility of confusion. Note that $\begin{bmatrix} n \\ k \end{bmatrix} = \begin{bmatrix} n \\ n-k \end{bmatrix}$ and $\begin{bmatrix} n \\ 0 \end{bmatrix} = \begin{bmatrix} n \\ n \end{bmatrix} = 1$.

Let $V \in \mathcal{G}_q(n, k)$ be a fixed vector space of dimension k , and let $N_q(n, k, j, \ell)$ denote the number of elements $W \in \mathcal{G}_q(n, j)$ with the property that $V \cap W \in \mathcal{G}_q(n, \ell)$. We have

$$N_q(n, k, j, \ell) = q^{(k-\ell)(j-\ell)} \begin{bmatrix} k \\ \ell \end{bmatrix} \begin{bmatrix} n-k \\ j-\ell \end{bmatrix}. \quad (3)$$

To see this, observe that the space U of intersection can be chosen in $\begin{bmatrix} k \\ \ell \end{bmatrix}$ ways. This subspace can be extended to a j -dimensional subspace in

$$\frac{(q^n - q^k)(q^n - q^{k+1})(q^n - q^{k+2}) \cdots (q^n - q^{k+j-\ell-1})}{(q^j - q^\ell)(q^j - q^{\ell+1})(q^j - q^{\ell+2}) \cdots (q^j - q^{j-1})} = q^{(j-\ell)(k-\ell)} \begin{bmatrix} n-k \\ j-\ell \end{bmatrix}$$

ways, since we can extend U by adjoining any of the $q^n - q^k$ vectors not in V , then adjoining any of the $q^n - q^{k+1}$ vectors not in the resulting $(k+1)$ -space, etc., but any specific choice is in an equivalent class of size $(q^j - q^\ell)(q^j - q^{\ell+1}) \cdots (q^j - q^{j-1})$.

The quantity $N_q(n, k, j, \ell)$ is very useful. For example, $N_q(n, n, k, k) = \begin{bmatrix} n \\ k \end{bmatrix}$ (the number of k -subspaces of an n -space, i.e., $|\mathcal{G}_q(n, k)|$), $N_q(n, k, j, k) = \begin{bmatrix} n-k \\ j-k \end{bmatrix}$ (the number of j -dimensional spaces containing the k -space V), $N_q(n, k, k, k-i) = q^{i^2} \begin{bmatrix} k \\ i \end{bmatrix} \begin{bmatrix} n-k \\ i \end{bmatrix}$ (the number of k -spaces at injection distance i from the k -space V), etc.

Let us also mention here two additional properties of the Gaussian coefficient [11]

$$\begin{bmatrix} m \\ n \end{bmatrix} \begin{bmatrix} n \\ t \end{bmatrix} = \begin{bmatrix} m \\ t \end{bmatrix} \begin{bmatrix} m-t \\ n-t \end{bmatrix}, \quad t \leq n \leq m, \quad (4)$$

and [1, Lemma 5]

$$q^{i(n-i)} \leq \binom{n}{i} \leq h(q)q^{i(n-i)}, \quad (5)$$

where $h(q) = \prod_{j=0}^{\infty} \frac{1}{1 - q^{-j}}$. It is shown in [1] that $h(q)$ decreases monotonically with q , approaching $q/(q-1)$ for large q . The series for $h(q)$ converges rapidly; the following table lists $h(q)$ for various values of q .

q	2	3	4	5	7	8	9	11	16	32	64	128	256
$h(q)$	3.46	1.79	1.45	1.32	1.20	1.16	1.14	1.11	1.07	1.03	1.02	1.01	1.004

2.2 Rank-Metric Codes

For matrices $X, Y \in \mathbb{F}_q^{n \times m}$, the *rank distance* is defined as

$$d_R(X, Y) \triangleq \text{rank}(Y - X).$$

As observed in [11], the rank distance is indeed a metric. A *rank-metric code* $\mathcal{C} \subseteq \mathbb{F}_q^{n \times m}$ is a matrix code (i.e., a nonempty set of matrices) used in the context of the rank metric. We use $d_R(\mathcal{C})$ to denote the minimum rank distance of \mathcal{C} . The Singleton bound for the rank metric [11, 12] (see also [3, 13, 14]) states that

$$|\mathcal{C}| \leq q^{\max\{n,m\}(\min\{n,m\}-d+1)}$$

for every code $\mathcal{C} \subseteq \mathbb{F}_q^{n \times m}$ with $d_R(\mathcal{C}) = d$. Codes that achieve this bound are called *maximum-rank-distance* (MRD) codes and linear MRD codes are known to exist for all choices of parameters q, n, m and $d \leq \min\{n, m\}$ [11].

Gabidulin codes [11] are an important class of MRD codes, described as follows. Without loss of generality, assume $n \leq m$ (otherwise consider the transposed version of the following argument). Let \mathbb{F}_{q^m} be an extension field of \mathbb{F}_q , and let $\theta: \mathbb{F}_{q^m} \rightarrow \mathbb{F}_q^m$ be a vector space isomorphism, where the elements in \mathbb{F}_q^m are regarded as row vectors. Let $\mathbb{F}_{q,m}^n[x]$ denote the set of linearized polynomials, i.e., all polynomials of the form $f(x) = \sum_{i=0}^{n-1} f_i x^{q^i}$, where $f_i \in \mathbb{F}_{q^m}$. Let $\alpha_1, \dots, \alpha_n \in \mathbb{F}_{q^m}$ be elements that are linearly independent when regarded as vectors in \mathbb{F}_q^m , and let $0 \leq d \leq n$.

A *Gabidulin code* $\mathcal{C} \subseteq \mathbb{F}_q^{n \times m}$ is defined as

$$\mathcal{C} = \left\{ c \in \mathbb{F}_q^{n \times m} : c = [\theta(f(\alpha_1)), \dots, \theta(f(\alpha_n))]^T, f(x) \in \mathbb{F}_{q,m}^{(n-d+1)}[x] \right\}.$$

It is shown in [11] that such a code has $d_R(\mathcal{C}) = d$, so it is indeed an MRD code.

Given a rank-metric code $\mathcal{C} \subseteq \mathbb{F}_q^{n \times m}$, a minimum-rank-distance decoder for \mathcal{C} takes a matrix $r \in \mathbb{F}_q^{n \times m}$ and returns a codeword $c \in \mathcal{C}$ that minimizes the rank distance $d_R(c, r)$. It is easy to see that, if $d_R(c, r) < d_R(\mathcal{C})/2$ for some $c \in \mathcal{C}$, then c is the unique solution to the above problem. A bounded-distance decoder for \mathcal{C} returns $c \in \mathcal{C}$ if $d_R(c, r) < d_R(\mathcal{C})/2$, or declares a failure if no such codeword can be found. For Gabidulin codes, very efficient bounded-distance decoders exist; see, e.g., [3, 11].

3 Bounds

In this section, we consider bounds on $A_q(n, d, k)$, $A_q(n, d)$, and $A_q^S(n, d)$. Since

$$A_q(n, d, k) = A_q(n, d, n - k), \quad (6)$$

when dealing with $A_q(n, d, k)$, we may safely assume $k \leq n/2$.

3.1 Upper Bounds on $A_q(n, d, k)$

Sphere-Packing Bound: The simplest upper bound that can be obtained for $A_q(n, d, k)$ is the sphere-packing bound, which follows from the fact that the Grassmann graph corresponding to $\mathcal{G}_q(n, k)$ is distance-regular. First, we need the concept of a sphere in $\mathcal{G}_q(n, k)$.

For $V \in \mathcal{G}_q(n, k)$, let $\mathcal{B}_V(t, k) \triangleq \{U \in \mathcal{G}_q(n, k) : \mathbf{d}(V, U) \leq t\}$ be the set of all subspaces of dimension k at injection distance at most t from V , a set that we regard as a sphere in $\mathcal{G}_q(n, k)$ of radius t with center V . For any $V \in \mathcal{G}_q(n, k)$, the size of $\mathcal{B}_V(t, k)$ is [1]

$$|\mathcal{B}_V(t, k)| = \sum_{i=0}^t q^{i^2} \begin{bmatrix} k \\ i \end{bmatrix} \begin{bmatrix} n-k \\ i \end{bmatrix}, \quad (7)$$

which follows easily from (3). Note that the size of a sphere in $\mathcal{G}_q(n, k)$ is independent of its center. For convenience, define $B(t, k) \triangleq |\mathcal{B}_V(t, k)|$.

The following sphere-packing bound for $A_q(n, d, k)$ is given in [1].

Theorem 1 (Sphere-packing bound)

$$A_q(n, d, k) \leq \frac{\begin{bmatrix} n \\ k \end{bmatrix}}{B(\lfloor (d-1)/2 \rfloor, k)}.$$

Singleton Bound. In [1] a puncturing operation in $\mathcal{G}_q(n, k)$ is defined that reduces by one the dimension of the ambient space and the dimension of each subspace in $\mathcal{G}_q(n, k)$. According to this puncturing operation, a punctured code obtained by puncturing an $(n, d, k)_q$ code is itself an $(n-1, d', k-1)_q$ code, where $d' \geq d-1$. If an $(n, d, k)_q$ code is punctured $d-1$ times repeatedly, an $(n-d+1, d'', k-d+1)_q$ code (with $d'' \geq 1$) is obtained, which may have size no greater than $|\mathcal{G}_q(n-d+1, k-d+1)|$. Thus the following Singleton-type bound is established [1].

Theorem 2 (Singleton bound)

$$A_q(n, d, k) \leq |\mathcal{G}_q(n-d+1, k-d+1)| = \begin{bmatrix} n-d+1 \\ k-d+1 \end{bmatrix} = \begin{bmatrix} n-d+1 \\ n-k \end{bmatrix}.$$

We note that from (5) it follows that

$$A_q(n, d, k) \leq h(q)q^{(n-k)(k-d+1)}. \quad (8)$$

It is observed in [1] that this bound is always stronger than the sphere-packing bound of Theorem 1 for nontrivial codes.

Anticode Bound. Since $\mathcal{G}_q(n, k)$ is an association scheme, the anticode bound of Delsarte [15] can be applied. Let \mathcal{C} be an $(n, d, k)_q$ code. Then Delsarte's bound implies that

$$|\mathcal{C}| \leq \frac{|\mathcal{G}_q(n, k)|}{|\mathcal{A}|},$$

where $\mathcal{A} \subseteq \mathcal{G}_q(n, k)$ is any set with maximum distance $d - 1$ (called an *anticode*).

Note that, for all $U, V \in \mathcal{G}_q(n, k)$, $d(U, V) \leq d - 1$ if and only if $\dim(U \cap V) \geq k - d + 1$. Thus, we can take \mathcal{A} as a set in which any two elements intersect in a space of dimension at least $k - d + 1$. From the results of Frankl and Wilson [16], it follows that, for $k \leq n/2$, the maximum value of $|\mathcal{A}|$ is equal to $\binom{n-k+d-1}{d-1}$. Hence, we have the following bound.

Theorem 3 (Anticode bound)

$$A_q(n, d, k) \leq \frac{\binom{n}{k}}{\binom{n-k+d-1}{d-1}} = \frac{\binom{n}{k-d+1}}{\binom{k}{k-d+1}}.$$

The equality in this theorem follows by observing from (4) that $\binom{n}{k} \binom{k}{k-d+1} = \binom{n}{k-d+1} \binom{n-k+d-1}{d-1}$. Applying (5) yields (8).

It is easy to observe that Delsarte's bound also implies the sphere-packing bound as a special case, since a sphere $\mathcal{B}_V(\lfloor(d-1)/2\rfloor, k)$ is (by the triangle inequality) an anticode of maximum distance $d - 1$. However, a sphere is not an optimal anticode in $\mathcal{G}_q(n, k)$, and therefore the bound of Theorem 3 is always tighter for nontrivial codes.

The bound in Theorem 3 was first obtained by Wang, Xing and Safavi-Naini in [6] using a different argument. The proof that Theorem 3 follows from Delsarte's bound is due to Etzion and Vardy [17].

As observed in [7], the anticode bound is always stronger than the Singleton bound for non-trivial codes in $\mathcal{G}_q(n, k)$.

Johnson-Type Bounds. Associated with an $(n, d, k)_q$ code \mathcal{C} is a binary constant weight code of length $q^n - 1$, weight $q^k - 1$, and minimum Hamming distance $2q^k(1 - q^{-d})$, having $|\mathcal{C}|$ codewords. This binary code has codewords that form the rows of the $|\mathcal{C}| \times (q^n - 1)$ incidence matrix between codewords of \mathcal{C} and the nonzero vectors of \mathbb{F}_q^n . The classical Johnson bound on binary constant weight codes (e.g., see [18]) immediately implies the following bound on $A_q(n, d, k)$.

Theorem 4 ([7])

$$A_q(n, d, k) \leq \frac{q^k(1 - q^{-d})(q^n - 1)}{(q^k - 1)^2 - (q^n - 1)(q^k - 1) + q^k(1 - q^{-d})(q^n - 1)}.$$

Now let \mathcal{C} be an $(n, d, k)_q$ code with $A_q(n, d, k)$ codewords. For any subspace $U \in \mathcal{G}_q(n, n-1)$ of dimension $n-1$, let \mathcal{C}_U be the set of codewords of \mathcal{C} contained entirely in U . Clearly \mathcal{C}_U is an $(n-1, d, k)_q$ code, and so cannot have cardinality

greater than $A_q(n-1, d, k)$. If we now form the summation of such cardinalities, ranging over all possible U , we obtain

$$\sum_{U \in \mathcal{G}_q(n, n-1)} |\mathcal{C}_U| = \left(\frac{q^{n-k} - 1}{q - 1} \right) A_q(n, d, k) \leq \left(\frac{q^{n-1} - 1}{q - 1} \right) A_q(n-1, d, k),$$

where the first equality follows from the fact that each codeword of \mathcal{C} will appear as a codeword in exactly $(q^{n-k} - 1)/(q - 1)$ of the \mathcal{C}_U 's. This argument yields the following theorem [17].

Theorem 5 ([17])

$$A_q(n, d, k) \leq \frac{q^n - 1}{q^{n-k} - 1} A_q(n-1, d, k)$$

Applying (6) results in the following.

Theorem 6 ([7, 17])

$$A_q(n, d, k) \leq \frac{q^n - 1}{q^k - 1} A_q(n-1, d, k-1)$$

Theorems 5 and 6 may be iterated to give an upper bound for $A_q(n, d, k)$. However, as in the classical case of the Johnson space, the order in which the two bounds should be iterated in order to get the tightest bound is unclear. By iterating Theorem 6 with itself, the following bound is established in [7, 17].

Theorem 7 ([7, 17])

$$A_q(n, d, k) \leq \left\lfloor \frac{q^n - 1}{q^k - 1} \left\lfloor \frac{q^{n-1} - 1}{q^{k-1} - 1} \cdots \left\lfloor \frac{q^{n-k+d} - 1}{q^d - 1} \right\rfloor \cdots \right\rfloor \right\rfloor.$$

It is shown in [7] that Theorem 5 improves on the anticode bound.

Ahlswede and Aydinian Bound. Let D be a nonempty subset of $\{1, \dots, n\}$ and let $\mathcal{C} \subseteq \mathcal{G}_q(n, k)$ be a code. If, for all $U, V \in \mathcal{C}$, with $U \neq V$, we have $d(U, V) \in D$, then we say that \mathcal{C} is a code with distances in D . The following Lemma is given in [19].

Lemma 1 ([19]). *Let $\mathcal{C}_D \subseteq \mathcal{G}_q(n, k)$ be a code with distances from a set D . Then, for a nonempty subset $\mathcal{B} \subseteq \mathcal{G}_q(n, k)$ there exists a code $\mathcal{C}_D^*(\mathcal{B}) \subseteq \mathcal{B}$ with distances from D such that*

$$\frac{|\mathcal{C}_D^*(\mathcal{B})|}{|\mathcal{B}|} \geq \frac{|\mathcal{C}_D|}{\binom{n}{k}},$$

where, if $|\mathcal{C}_D^*| = 1$, then \mathcal{C}_D^* is a code with distances from D by convention.

In particular when \mathcal{C}_D is an $(n, d, k)_q$ code and \mathcal{B} is an anticode of maximum distance $d - 1$, then $|\mathcal{C}_D^*(\mathcal{B})| = 1$ and Delsarte's anticode bound on $\mathcal{G}_q(n, k)$ is obtained.

Using Lemma 1 Ahlswede and Aydinian obtain the following bound:

Theorem 8 ([19]). *For integers $0 \leq t \leq d \leq k$, $k - t \leq m \leq n$,*

$$A_q(n, d, k) \leq \frac{\binom{n}{k} A_q(m, d - t, k - t)}{\sum_{i=0}^t q^{i(m-i)} \binom{m}{k-i} \binom{n-m}{i}}$$

It is shown in [19] that for $t = 0$ and $m = n - 1$, Theorem 8 gives Theorem 5.

3.2 Upper Bounds on $A_q(n, d)$ and $A_q^S(n, d)$

A Simple Bound. The simplest upper bound in $A_q(n, d)$ follows immediately from the observation that every subspace code is a union of constant-dimension codes, and hence

$$A_q(n, d) \leq \sum_{k=0}^n A_q(n, d, k)$$

Etzion-Vardy LP Bound. Etzion and Vardy derive in [17] the following linear programming bound for $A_q^S(n, 3)$.

Theorem 9 ([17]). *Let $f^* = \max(\sum_{i=0}^n D_i)$ subject to the following linear constraints:*

$$\frac{q^{n-i+1} - 1}{q - 1} D_{i-1} + D_i + \frac{q^{i+1} - 1}{q - 1} D_{i+1} \leq \binom{n}{i} \quad (9)$$

and $D_i \leq A_q(n, 2, i)$, for all $i = 0, 1, \dots, n$, where $D_{-1} = D_{n+1} = 0$ by convention. Then

$$A_q^S(n, 3) \leq f^*.$$

Ahlswede-Aydinian LP Bound. Ahlswede and Aydinian establish the following linear programming bound for $A_q(n, d)$ in [19].

Theorem 10 ([19]). *For integers $1 \leq d \leq \frac{n}{2}$, let*

$$f(n, d, q) = \max(\sum_{i=0}^n f_i)$$

subject to the following linear constraints:

$$f_i \in \mathbb{N} \text{ for } i = 0, 1, \dots, n.$$

$$f_0 = f_n = 1, f_k = f_{n-k} = 0 \text{ for } k = 1, \dots, d$$

$$f_{-j} = f_{n+j} = 0 \text{ for } j = 1, \dots, d \text{ (by convention)}$$

$$f_k + \frac{1}{d+1} \sum_{i=1}^d (d+1-i) \left(f_{k-i} \binom{n-k+i}{n-k} + f_{k+i} \binom{k+i}{k} \right) \leq \binom{n}{k} \text{ and,}$$

$$f_k \leq A_q(n, d+1, k) \text{ for } k = 0, \dots, n.$$

Then,

$$A_q(n, d) \leq f(n, d, q).$$

3.3 Lower Bounds

In this section, we give the counterparts of the Gilbert-Varshamov lower bound for $A_q(n, d, k)$, $A_q(n, d)$ and $A_q^S(n, d)$. We start with the sizes of spheres in $\mathcal{P}_q(n)$. Recall that the size of a sphere in $\mathcal{G}_q(n, k)$ was given in (7).

For $V \in \mathcal{P}_q(n)$, let $\mathcal{B}_V(t) \triangleq \{U \in \mathcal{P}_q(n) : d(V, U) \leq t\}$ be the sphere of radius t centered at V in $\mathcal{P}_q(n)$. For any $V \in \mathcal{P}_q(n)$, the size of $\mathcal{B}_V(t)$ can be computed using (3) as [20]

$$|\mathcal{B}_V(t)| = \sum_{r=0}^t q^{r^2} \begin{bmatrix} k \\ r \end{bmatrix} \begin{bmatrix} n-k \\ r \end{bmatrix} + \sum_{j=1}^r q^{r(r-j)} \left(\begin{bmatrix} k \\ r \end{bmatrix} \begin{bmatrix} n-k \\ r-j \end{bmatrix} + \begin{bmatrix} n-k \\ r \end{bmatrix} \begin{bmatrix} k \\ r-j \end{bmatrix} \right) \quad (10)$$

where $k = \dim(V)$. Note that the size of a sphere in $\mathcal{P}_q(n)$ does not depend on the specific subspace at its center, but does depend on its dimension. For convenience, we use the notation $B_k(t) \triangleq |\mathcal{B}_V(t)|$, where $k = \dim(V)$.

We can also define the analogous concept of a sphere under the subspace distance, which is denoted as $\mathcal{B}_V^S(t)$ for $V \in \mathcal{P}_q(n)$. It is shown in [17] that

$$|\mathcal{B}_V^S(t)| = \sum_{r=0}^t \sum_{j=0}^r q^{j(r-j)} \begin{bmatrix} n-k \\ r-j \end{bmatrix} \begin{bmatrix} k \\ j \end{bmatrix} \begin{bmatrix} n \\ k \end{bmatrix} \quad (11)$$

where $k = \dim(V)$. Similarly as above, we use the notation $B_k^S(t) \triangleq |\mathcal{B}_V^S(t)|$.

Let Ω be a general metric space with distance metric denoted by δ . Let $\mathcal{B}_\alpha(t) \triangleq \{\beta \in \Omega : \delta(\alpha, \beta) \leq t\}$ be a sphere of radius t centered at α in Ω . Every maximal code \mathcal{C} of minimum distance d must satisfy

$$\sum_{c \in \mathcal{C}} |\mathcal{B}_c(d-1)| \geq |\Omega|. \quad (12)$$

Since the size $B(t, k)$ of a sphere $\mathcal{B}_V(t, k)$ in $\mathcal{G}_q(n, k)$ is independent of V , when Ω is replaced with $\mathcal{G}_q(n, k)$ and $\delta(\cdot, \cdot)$ with the injection metric, (12) results in the following Gilbert-Varshamov bound on $A_q(n, d, k)$.

Theorem 11 ([1])

$$A_q(n, d, k) \geq \frac{|\mathcal{G}_q(n, k)|}{B(d-1, k)}. \quad (13)$$

Since by (11), the size of a sphere $\mathcal{B}_V(t)$ in $\mathcal{P}_q(n)$ depends on $\dim(V)$, the approach of Theorem 11 is not suitable for the derivation of a Gilbert-Varshamov bound in $\mathcal{P}_q(n)$.

As pointed out in [17], the appropriate framework for a Gilbert-Varshamov bound in spaces where the size of a sphere depends upon the location of its center

is given by Tolhuizen [21]. Let $\overline{B}(t) \triangleq \frac{1}{|\Omega|} \sum_{\alpha \in \Omega} |\mathcal{B}_\alpha(t)|$ denote the “average size” of a sphere of radius t in Ω . Tolhuizen showed in [21] that the maximum size of a code $C \subseteq \Omega$ of minimum distance d is at least $\frac{|\Omega|}{\overline{B}(d-1)}$. Using this result along with (10), Khaleghi and Kschischang [20] and independently Gadouleau and Yan [14] obtain the following Gilbert-Varshamov bound for $A_q(n, d)$:

Theorem 12 ([14, 20]). $A_q(n, d) \geq \frac{|\mathcal{P}_q(n)|^2}{\sum_{k=0}^n \binom{n}{k} B_k(d-1)}.$

Earlier, Etzion and Vardy [17] had already established the following Gilbert-Varshamov bound on $A_q^S(n, d)$:

Theorem 13 ([17]). $A_q^S(n, d) \geq \frac{|\mathcal{P}_q(n)|^2}{\sum_{k=0}^n \binom{n}{k} B_k^S(d-1)}$

where $B_k^S(d-1)$ is given by (11).

Unlike the case of classical coding theory in the Hamming metric, the best lower bounds on $A_q(n, d)$ and $A_q(n, k, k)$ result from code constructions, the subject of the next section.

4 Constructions

4.1 Lifted Rank-Metric Codes

In this section, we describe the simplest construction of asymptotically good subspace codes, which uses rank-metric codes as building blocks. This construction was first proposed in [6], and then rediscovered in [1] for the special case where the rank-metric code is a Gabidulin code. The construction was later explained in [3, 22] in the context of the subspace/injection distance. The latter description is reviewed below.

For a matrix $X \in \mathbb{F}_q^{k \times m}$, let the subspace $\Lambda(X) \triangleq \langle [I_{k \times k} \ X] \rangle \in \mathcal{G}_q(k+m, k)$ be called the *lifting* of X . Similarly, for a matrix code $\mathcal{C} \subseteq \mathbb{F}_q^{k \times m}$, let the subspace code $\Lambda(\mathcal{C}) \triangleq \{\Lambda(X), X \in \mathcal{C}\}$ be called the *lifting* of \mathcal{C} . Since every subspace corresponds to a unique matrix in RREF, we have that the mapping $X \rightarrow \Lambda(X)$ is injective, and therefore $|\Lambda(\mathcal{C})| = |\mathcal{C}|$. Note that $\Lambda(\mathcal{C})$ is a constant-dimension code, i.e., $\Lambda(\mathcal{C}) \subseteq \mathcal{G}_q(k+m, k)$.

Lemma 2 (Lifting Lemma [3]). *For all $X, X' \in \mathbb{F}_q^{k \times m}$ and all $\mathcal{C} \subseteq \mathbb{F}_q^{k \times m}$,*

$$\begin{aligned} d(\Lambda(X), \Lambda(X')) &= d_R(X, X'), \\ d(\Lambda(\mathcal{C})) &= d_R(\mathcal{C}). \end{aligned}$$

Proof. We have

$$\begin{aligned}
d(\Lambda(X), \Lambda(X')) &= \dim(\Lambda(X) + \Lambda(X')) - \min\{\dim(\Lambda(X)), \dim(\Lambda(X'))\} \\
&= \text{rank} \begin{bmatrix} I & X \\ I & X' \end{bmatrix} - k \\
&= \text{rank} \begin{bmatrix} I & X \\ 0 & X' - X \end{bmatrix} - k \\
&= \text{rank}(X' - X).
\end{aligned}$$

The second statement immediately follows from the first.

Lemma 2 shows that a subspace code constructed by lifting inherits the distance properties of its underlying rank-metric code.

In particular, let $\mathcal{C} \subseteq \mathbb{F}_q^{k \times (n-k)}$ be an MRD code with $d_R(\mathcal{C}) = d$ and, without loss of generality, let $k \leq n - k$. Then $\Lambda(\mathcal{C})$ is an (n, d, k) code with cardinality

$$|\Lambda(\mathcal{C})| = q^{(n-k)(k-d+1)}. \quad (14)$$

Note that (14) gives a lower bound on $A_q(n, d, k)$. Comparing with the upper bound of (8), we see that the ratio of the upper and lower bounds is a constant depending only on q , thus demonstrating that this construction yields asymptotically optimal codes.

Optimizing k in (14), we obtain

$$A_q(n, d) \geq q^{\lceil \frac{n}{2} \rceil (\lfloor \frac{n}{2} \rfloor - d + 1)}.$$

We now mention a particular way of constructing lifted rank-metric codes. When $m \geq 2k$ it is convenient to construct an MRD code $\mathcal{C} \subseteq \mathbb{F}_q^{k \times m}$ as a Cartesian product of simpler MRD codes. Let $m_1, \dots, m_r \geq k$ be such that $\sum_{i=1}^r m_i = m$, and let $\mathcal{C}_i \subseteq \mathbb{F}_q^{k \times m_i}$, $i = 1, \dots, r$, be MRD codes with minimum rank distance d . Then, it is easy to see that the Cartesian product $\mathcal{C} = \mathcal{C}_1 \times \dots \times \mathcal{C}_r$ is also an MRD code with $d_R(\mathcal{C}) = d$, where a specific element (X_1, \dots, X_r) in the Cartesian product is interpreted as the $k \times m$ matrix $[X_1 \ X_2 \ \dots \ X_r]$. Clearly, we have $|\mathcal{C}| = \prod_{i=1}^r q^{m_i(k-d+1)} = q^{m(k-d+1)}$. Note the importance of choosing $m_i \geq k$ for the resulting code to be MRD. Now, since $d_R(\mathcal{C}) = d$, it follows that $\Lambda(\mathcal{C})$ is a $(k+m, k, d)_q$ code.

4.2 Padded Codes

Padded codes are a set of subspace codes in $\mathcal{G}_q(n, k)$ obtained as a union of lifted product rank-metric codes. Let $n = (r+1)k + s$, where $r, s \in \mathbb{N}$ and $s < k$. Let $\mathcal{C} \subseteq \mathbb{F}_q^{k \times k}$, and $\mathcal{C}' \subseteq \mathbb{F}_q^{k \times (k+s)}$ be rank-metric codes of minimum rank-distance d .

Define a padded code as $\Omega = \bigcup_{i=0}^{r-1} \Omega_i$, where

$$\Omega_i = \left\{ \langle \overbrace{[\mathbf{0}_{k \times k} \cdots \mathbf{0}_{k \times k}]}^i \ I_{k \times k} \ c_{i+1} \cdots c_r \rangle \right\}$$

with $c_j \in \mathcal{C}$ for $j = 1, \dots, r-1$, and $c_r \in \mathcal{C}'$.

It is clear that $\mathbf{d}(\Omega_i) = d$. Now, let $j < i \leq r - 1$, and consider $U = \langle X \rangle \in \Omega_i$ and $V = \langle Y \rangle \in \Omega_j$, where $X = [\overbrace{\mathbf{0}_{k \times k} \cdots \mathbf{0}_{k \times k}}^j I_{k \times k} c_{i+1} \cdots c_r]$ and $Y = [\overbrace{\mathbf{0}_{k \times k} \cdots \mathbf{0}_{k \times k}}^i I_{k \times k} c_{i+1} \cdots c_r]$. Since $j < i$, $I_{k \times k}$ in X and Y are not aligned. Therefore,

$$\dim(U + V) = \text{rank} \begin{bmatrix} X \\ Y \end{bmatrix} = 2k,$$

and $\mathbf{d}(U, V) = \dim(U + V) - k = k \geq d$. Thus we obtain $\mathbf{d}(\Omega) = d$.

When $\mathcal{C} = \mathcal{C}'$ are Gabidulin codes, we obtain a special case of the construction in [23]. If in addition $\mathbf{d}_R(\mathcal{C}) = k$, then the construction above results in the “spread codes” of [24] and [25].

4.3 Lifted FD Codes

In [26], Etzion and Silberstein provide a multi-level construction for codes in $\mathcal{P}_q(n)$. The basic idea of this construction is to generalize the lifting construction to Schubert cells (as defined in Section 2) so that a lifted rank-metric code is contained completely within any given cell. A code can then be constructed by taking a union of such lifted rank-metric codes in suitably well-separated Schubert cells. We now give a detailed description of this construction.

For a subspace $V \in \mathcal{P}_q(n)$ and a nonsingular matrix $T \in \mathbb{F}_q^{n \times n}$, define $VT \triangleq \{vT, v \in V\}$ (which is a subspace isomorphic to V). Given any binary vector b of length n and weight k , define $P(b)$ as the $n \times n$ permutation matrix such that $P(b)_{\text{supp}(b)} = [I_{k \times k} \mathbf{0}_{k \times (n-k)}]$ and $P(b)_{\text{supp}(\bar{b})} = [\mathbf{0}_{(n-k) \times k} I_{(n-k) \times (n-k)}]$. Multiplication of a matrix $[X \ Y]$, where X is $k \times k$ and Y is $k \times (n-k)$, by $P(b)^{-1}$ on the right results in a matrix in which the columns are permuted. Specifically, the columns of X appear in columns indexed by $\text{supp}(b)$, and columns of Y appear in columns indexed by $\text{supp}(\bar{b})$, and the order of the columns within each submatrix is preserved.

Now, let b be a binary vector of length n and weight k . For a matrix $X \in \mathbb{F}_q^{k \times (n-k)}$, define the generalized lifting, $\Lambda_b(X)$, of X with respect to b as

$$\Lambda_b(X) \triangleq \Lambda(X)P(b)^{-1} = \langle [I \ X] P(b)^{-1} \rangle.$$

Since $\text{rank}([I \ X] P(b)^{-1}) = k$, we observe that $\Lambda_b(X)$ is a k -dimensional subspace of \mathbb{F}_q^n . Similarly, for a matrix code $\mathcal{C} \subseteq \mathbb{F}_q^{k \times (n-k)}$, let

$$\Lambda_b(\mathcal{C}) \triangleq \{ \Lambda_b(c), c \in \mathcal{C} \}.$$

Note that the lifting $\Lambda(\cdot)$ defined in Section 4.1 is a special case of $\Lambda_b(\cdot)$, namely, $\Lambda(X) = \Lambda_b(X)$ where $b = (1, \dots, 1, 0, \dots, 0)$.

The generalized lifting of a matrix code does not generally lead to a subspace code confined to a single Schubert cell. However, if the matrix code is suitably constrained in a manner depending on b , then its image will indeed be confined to the Schubert cell $\mathcal{S}_q(b)$ corresponding to b . The particular constraints are described as follows.

Let $Q = [Q_{ij}]$ be the $n \times n$ upper triangular matrix with $Q_{ij} = 1$ if $j \geq i$ and $Q_{ij} = 0$ otherwise. Given a binary profile vector b of length n and weight k , regarded as an element of $\mathbb{Z}^{1 \times n}$, define the vector $c(b) \in \mathbb{Z}^{1 \times n}$ via

$$c(b) \triangleq bQP(b).$$

Then, the generalized lifting $\Lambda_b(X)$ of a matrix $X = [x_{ij}] \in \mathbb{F}_q^{k \times (n-k)}$ is guaranteed to be in the Schubert cell corresponding to b provided that

$$\text{for } 1 \leq i \leq k, 1 \leq j \leq n-k, i > c(b)_{j+k} \text{ implies that } x_{ij} = 0. \quad (15)$$

For example, suppose $n = 8$ and $k = 3$, and let $b = (0, 0, 1, 0, 1, 0, 0, 1)$. Then,

$$P(b) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } c(b) = (1, 2, 3, 0, 0, 1, 2, 2).$$

Let $X \in \mathbb{F}_q^{3 \times 5} = [x_{ij}]$. Observe that

$$[I \ X] P(b)^{-1} = \begin{bmatrix} x_{11} & x_{12} & 1 & x_{13} & 0 & x_{14} & x_{15} & 0 \\ x_{21} & x_{22} & 0 & x_{23} & 1 & x_{24} & x_{25} & 0 \\ x_{31} & x_{32} & 0 & x_{33} & 0 & x_{34} & x_{35} & 1 \end{bmatrix}.$$

Clearly this matrix is in RREF and hence $\text{prof}([I \ X] P(b)^{-1}) = b$ if

$$x_{11} = x_{21} = x_{31} = x_{12} = x_{22} = x_{32} = x_{23} = x_{33} = x_{34} = x_{35} = 0.$$

These conditions are precisely those implied by (15).

Now let b be a binary vector of length n and weight k . Let $\mathcal{C} \subseteq \mathbb{F}_q^{k \times (n-k)}$ be a rank-metric code with $d_R(\mathcal{C}) = d$ in which each codeword satisfies (15). We refer to such a code as an FD(b) code, where FD stands for “Ferrers’ Diagram” [26]. Clearly, $\Lambda_b(\mathcal{C})$ consists of subspaces in the Schubert-cell corresponding to b , and by Lemma 2 we have that $d(\Lambda_b(\mathcal{C})) = d$, and $d_S(\Lambda_b(\mathcal{C})) = 2d$. The code $\Lambda_b(\mathcal{C})$ is referred to as a lifted FD(b) code.

In [20, 27] a construction for FD(b) codes is presented, where a code \mathcal{C}_b is obtained as a subcode of a linear MRD code with a further set of linear constraints ensuring that each codeword in \mathcal{C}_b satisfies (15). The following theorem gives a lower bound on the cardinality of these codes.

Theorem 14 ([20, 27]). *For a binary vector b of length n with $\text{wt}(b) = k > 0$, let \mathcal{C}_b be an FD(b) code of minimum rank-distance d , obtained via the construction presented in [20, 27]. We have*

$$|\mathcal{C}_b| \geq q^{w(b) - \max\{\mu(b), \eta(b)\}(d-1)},$$

where $w(b) = \sum_{i>k} c(b)_i$, $\mu(b) = \max\{c(b)_i : i > k\}$ and $\eta(b) = \text{wt}(c(b)) - k$.

We now consider the minimum distance between elements in distinct Schubert cells. Let u and v be two distinct binary vectors of length n and having weights k and k' respectively, and let $u \wedge v$ denote the logical AND of u and v , i.e., the binary vector in which $(u \wedge v)_i = u_i v_i$. Let U and V be arbitrary vector spaces in the Schubert cells $\mathcal{S}_q(u)$ and $\mathcal{S}_q(v)$, respectively. The following lower bound on $d(U, V)$ is given in [20]:

Theorem 15 ([20]). $d(U, V) \geq d_a(u, v)$, where $d_a(u, v) = \max\{\text{wt}(u), \text{wt}(v)\} - \text{wt}(u \wedge v)$ is a metric known as the asymmetric distance between u and v .

Proof. Clearly $\dim(U) = \text{wt}(u)$ and $\dim(V) = \text{wt}(v)$. Let $w = u \wedge v$ and observe that $\dim(U \cap V) \leq \text{wt}(w)$. Thus,

$$\dim(U) - \dim(U \cap V) \geq \text{wt}(u) - \text{wt}(w).$$

Similarly,

$$\dim(V) - \dim(U \cap V) \geq \text{wt}(v) - \text{wt}(w).$$

Taking the $\max\{\cdot, \cdot\}$ of both equations we obtain

$$\begin{aligned} d(U, V) &\geq \max\{\text{wt}(u), \text{wt}(v)\} - \text{wt}(w) \\ &= d_a(u, v). \end{aligned}$$

Earlier, Etzion and Silberstein [26] had given the following theorem:

Theorem 16 ([26]). $d_S(U, V) \geq d_H(u, v)$

Proof. Let $N(u, v) = \text{wt}(u) - \text{wt}(v)$, and $N(v, u) = \text{wt}(v) - \text{wt}(u)$. In a manner similar to the proof of Theorem 15 we have,

$$\begin{aligned} N(u, v) &= \text{wt}(u) - \text{wt}(w) \\ &= \dim(U) - \text{wt}(w) \\ &\leq \dim(U) - \dim(U \cap V) \end{aligned}$$

Similarly $N(v, u) \leq \dim(V) - \dim(U \cap V)$, thus we have

$$d_H(u, v) = N(u, v) + N(v, u) \leq \dim(U) + \dim(V) - 2 \dim(U \cap V) = d_S(U, V).$$

Note that both lower bounds are achieved with equality when U and V correspond to lifted all-zero codewords.

Finally let \mathcal{A} be a binary code of length n . For every element $b \in \mathcal{A}$, let \mathcal{C}_b be a FD(b) code. Then

$$\Omega = \bigcup_{b \in \mathcal{A}} \Lambda_b(\mathcal{C}_b)$$

is a subspace code.

If \mathcal{A} has minimum asymmetric distance d and each FD(b) code is designed to have minimum rank-distance d , then Ω is guaranteed to have minimum injection distance d . Similarly, if \mathcal{A} has minimum Hamming distance $2d$ and each FD(b)

code is designed to have minimum rank-distance d , then Ω is guaranteed to have minimum subspace distance $2d$. These codes are the lifted Ferrer's diagram rank-metric codes of [20, 26] designed for the injection and subspace distance respectively. We refer to such codes as lifted FD codes.

It is interesting to observe that this construction included the padded codes of Section 4.2 as a special case. In particular, let $\mathcal{P} \subseteq \{0, 1\}^n$ be a set of constant-weight binary vectors of weight $k \leq n$ such that,

$$\text{for all } v \in \mathcal{P}, v = (\overbrace{0, 0, \dots, 0}^i, \overbrace{1, 1, \dots, 1}^k, \overbrace{0, 0, \dots, 0}^{n-k-i}).$$

Let $\mathcal{C} \subseteq \mathbb{F}_q^{k \times k}$ be a rank-metric code of minimum rank-distance d . Then $\{\Lambda_v(\mathcal{C}) : v \in \mathcal{P}\}$ is a padded code in $\mathcal{G}_q(n, k)$ with minimum injection distance d .

Notice that in this construction a naive choice for \mathcal{A} would be one with a high information rate. However, a high information rate would only result in a large number of selected Schubert cells, and does not necessarily guarantee a high overall rate for the resulting $(n, d)_q$ code. This is due to the fact that the rate of a lifted FD code depends on the rate of its underlying FD(b) codes, which in turn by Theorem 14 depend on the particular choices of b .

In [26] constant-weight lexicodes are used to select well-separated Schubert cells in the Grassmannian. In [20, 27] a scoring function is defined, which given a minimum distance d , calculates for every $b \in \{0, 1\}^n$ the bound of Theorem 14. In order to construct \mathcal{A} , a standard greedy algorithm is used that maintains a list of available profile vectors $\mathcal{A} \subseteq \{0, 1\}^n$, (with \mathcal{A} initialized to $\{0, 1\}^n$). At each step an available vector with the highest score is added to \mathcal{A} , and vectors within asymmetric distance d of b are made unavailable. The algorithm proceeds until $\mathcal{A} = \emptyset$. Results obtained from this algorithm are tabulated in [27].

4.4 Codes Obtained by Integer Linear Programming

In [28] Kohnert and Kurtz view the construction of constant-dimension subspace codes as an optimization problem involving integer variables.

Let \mathcal{C} be an $(n, d, k)_q$ code so that for all $U, V \in \mathcal{C}$ we have $\mathsf{d}(U, V) = k - \dim(U \cap V) \geq d$. The code construction problem is equivalent to finding a set of N subspaces $\mathcal{C} = \{V_1, V_2, \dots, V_N\} \in \mathcal{G}_q(n, k)$ such that for all $i, j \in \{1, 2, \dots, N\}$, $V_i, V_j \in \mathcal{C}$, we have $\dim(V_i \cap V_j) \leq k - d$. This means that no pair of subspaces in \mathcal{C} intersect in a $(k - d + 1)$ -space in $\mathcal{P}_q(n)$. Let $M \in \mathbb{F}_2^{[n \choose k-d+1] \times [n \choose k]}$ be an incidence matrix defined as follows:

$$M_{W,V} := \begin{cases} 1 & \text{if } W \subseteq V, \\ 0 & \text{otherwise.} \end{cases}$$

Let x be a binary vector of length $[n \choose k]$. The code construction problem may be viewed as the following optimization problem:

$$\text{maximize} \sum_{i=1}^{[n \choose k]} x_i, \text{ subject to } Mx \leq \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Let \mathcal{S} be an ordered set obtained by taking the subspaces in $\mathcal{G}_q(n, k)$ in some arbitrary order. Then, if x is the solution to the above optimization problem, we may construct a subspace code $\mathcal{C} \subseteq \mathcal{G}_q(n, k)$ of minimum distance d , by taking the subspaces in \mathcal{S} indexed by $\text{supp}(x)$.

It is possible to significantly reduce the size of the problem by prescribing a group of automorphisms for the code, and then using the induced symmetry to reduce the number of equations. See [28] for details.

5 Encoding and Decoding

Let $\Omega \in \mathcal{P}_q(n)$ be a subspace code with $\mathsf{d}(\Omega) = d$. Throughout this section, let $t = \lfloor (d-1)/2 \rfloor$. In this section, we consider two problems related to the use of Ω for error control in noncoherent linear network coding. The *encoding problem* is how to efficiently map an integer in $\{0, \dots, |\Omega| - 1\}$ into a codeword of Ω (and back). The *decoding problem* is how to efficiently find a codeword of Ω that is closest (in injection distance) to a given subspace $U \in \mathcal{P}_q(n)$. More specifically, we focus on a *bounded-distance decoder*, which returns a codeword $V \in \Omega$ if V is the unique codeword that satisfies $\mathsf{d}(V, U) \leq t$ and returns a failure otherwise.

5.1 Encoding Lifted FD Codes

Let $\Omega = \bigcup_{b \in \mathcal{A}} \Lambda_b(\mathcal{C}_b)$ be an $(n, d)_q$ lifted FD-code constructed as described in Section 4.3, and suppose that \mathcal{A} is given $\{b_1, b_2, \dots, b_{|\mathcal{A}|}\}$. Let $c_1 = 0$, and, for $2 \leq i \leq |\mathcal{A}|$, let $c_i = \sum_{j=1}^{i-1} |\mathcal{C}_{b_j}|$.

Codewords are numbered starting at zero. To map an integer m in the range $\{0, \dots, |\Omega| - 1\}$ to a codeword: (a) find the largest index i such that $c_i \leq m$, (b) map the integer $m - c_i$ to a codeword of \mathcal{C}_{b_i} (using an encoder for the corresponding rank-metric code), which can then be lifted to the corresponding subspace. Note that $0 \leq m_i < |\mathcal{C}_{b_i}|$. Conversely, the j th codeword of \mathcal{C}_{b_i} maps back to the message $m = c_i + j$. Assuming efficient encoding of the underlying rank-metric codes, the main complexity of the encoding algorithm, given m , is to determine the corresponding c_i , which can be done using a binary search in time at worst proportional to $\log |\mathcal{A}|$.

5.2 Decoding Lifted Gabidulin Codes

Let $\mathcal{C} \subseteq \mathbb{F}_q^{k \times m}$ be a Gabidulin code with $\mathsf{d}_R(\mathcal{C}) = d$. Recall that $\Lambda(\mathcal{C})$ is a $(k+m, d, k)_q$ code.

A bounded-distance decoder for $\Lambda(\mathcal{C})$ is a function $\text{dec}: \mathcal{P}_q(n) \rightarrow \mathcal{C} \cup \{\varepsilon\}$ such that $\text{dec}(U) = c$ for all $U \in \mathcal{B}_{\Lambda(c)}(t)$ and all $c \in \mathcal{C}$, and such that $\text{dec}(U) = \varepsilon$ for all other U .

Let us first point out that decoding of $\Lambda(\mathcal{C})$ is not a straightforward application of rank-distance decoding. To see this, let $A \in \mathbb{F}_q^{\ell \times k}$, $y \in \mathbb{F}_q^{\ell \times (n-k)}$ and $Y = [A \ y]$

be such that $\langle Y \rangle = U$ is the received subspace. If $\ell = k$ and A is nonsingular, then

$$\mathbf{d}(\Lambda(c), U) = \mathbf{d}_R(c, A^{-1}y)$$

and therefore decoding of Ω reduces to rank-distance decoding of \mathcal{C} . In general, however, A may not be invertible, in which case the argument above does not hold.

Several algorithms have been proposed for implementing the function $\text{dec}(\cdot)$. The first such algorithm was proposed by Kötter and Kschischang in [1] and is a version of Sudan's "list-of-1" decoding for Gabidulin codes. The time complexity of the algorithm is $O((k+m)^2m^2)$ operations in \mathbb{F}_q . A faster algorithm was proposed in [3] which is a generalization of the standard ("time-domain") decoding algorithm for Gabidulin codes. The complexity of this algorithm is $O(dm^3)$ operations in \mathbb{F}_q . As shown in [29], the algorithm in [3] can significantly benefit from the use of optimal (or low-complexity) normal bases, further reducing the decoding complexity to $(11t^2 + 13t + m)m^2/2$ multiplications in \mathbb{F}_q (and a similar number of additions). Finally, a transform-domain decoding algorithm was proposed in [22, 29], which is slightly faster than that of [3] for low-rate codes.

As we will see, a bounded-distance decoder for a lifted Gabidulin code can be used as a black box for decoding many other subspace codes. For instance, consider $\mathcal{C}^r = \mathcal{C} \times \cdots \times \mathcal{C}$, the r th Cartesian power of a Gabidulin code $\mathcal{C} \subseteq \mathbb{F}_q^{k \times m}$. Recall that $\Lambda(\mathcal{C}^r)$ is a $(k+rm, d, k)_q$ code, where $d = \mathbf{d}_R(\mathcal{C})$. Let $\text{dec}(\cdot)$ be a bounded-distance decoder for $\Lambda(\mathcal{C})$. Then a bounded-distance decoder for $\Lambda(\mathcal{C}^r)$ can be obtained as the map $\mathcal{P}_q(k+rm) \rightarrow \mathcal{C}^r \cup \{\varepsilon\}$ given by

$$U \mapsto \begin{cases} [\hat{c}_1 \cdots \hat{c}_r] = \hat{c} & \text{if } \hat{c}_i \neq \varepsilon, i = 1, \dots, r, \text{ and } \mathbf{d}(\Lambda(\hat{c}), U) \leq t \\ \varepsilon & \text{otherwise} \end{cases}$$

where $\hat{c}_i = \text{dec}(\langle [A y_i] \rangle)$, $i = 1, \dots, r$, and $A \in \mathbb{F}_q^{\ell \times k}$ and $y_1, \dots, y_r \in \mathbb{F}_q^{\ell \times m}$ are such that $\langle [A y_1 \cdots y_r] \rangle = U$. In other words, we can decode $\Lambda(\mathcal{C}^r)$ by decoding each Cartesian component individually (using the same matrix A on the left) and then checking whether the resulting subspace codeword is within the bounded distance from U .

5.3 Decoding Lifted FD Codes

Let \mathcal{A} be a binary code with $\mathbf{d}_a(\mathcal{A}) \geq d$. For all $b \in \mathcal{A}$, let \mathcal{C}_b be a b -FD code with $\mathbf{d}_R(\mathcal{C}_b) \geq d$. Let

$$\Omega = \bigcup_{b \in \mathcal{A}} \Lambda_b(\mathcal{C}_b).$$

Recall that Ω is an $(n, d)_q$ code.

A bounded-distance decoder for Ω is a function $\text{dec}: \mathcal{P}_q(n) \rightarrow (\mathcal{A} \times \bigcup_{b \in \mathcal{B}} \mathcal{C}_b) \cup \{\varepsilon\}$ such that $\text{dec}(U) = (b, c)$ for all $U \in \mathcal{B}_{\Lambda_b(c)}(t)$, all $c \in \mathcal{C}_b$, and all $b \in \mathcal{A}$, and such that $\text{dec}(U) = \varepsilon$ for all other U .

We will show that we can efficiently decode Ω , provided that we have efficient decoders for \mathcal{A} and for each \mathcal{C}_b , $b \in \mathcal{A}$. The basic procedure was proposed in

[26] for the decoding in the subspace metric. Here we adapt it for the injection metric.

Let us first consider the decoding of $\Lambda_b(\mathcal{C}_b)$, for some $b \in \mathcal{A}$. Let $c \in \mathcal{C}_b$ and $U \in \mathcal{P}_q(n)$. Recall that $\Lambda_b(c) = \Lambda(c)P(b)^{-1}$. Since $P(b)$ is a nonsingular linear transformation, and therefore preserves dimensions, we have

$$\mathsf{d}(\Lambda_b(c), U) = \mathsf{d}(\Lambda_b(c)P(b), UP(b)) = \mathsf{d}(\Lambda(c), UP(b)).$$

It follows that bounded-distance decoding of $\Lambda_b(\mathcal{C}_b)$ can be performed by first computing $\hat{c} = \text{dec}(UP(b))$, and then returning (b, \hat{c}) unless $\hat{c} = \varepsilon$.

Now, consider the decoding of Ω . Let $U \in \mathcal{P}_q(n)$ be such that $\mathsf{d}(V, U) \leq t$, for some (unique) $V \in \Omega$. The first step is to compute the profile vector b corresponding to the Schubert cell containing V . Let b' denote the profile vector corresponding to the Schubert cell containing U . Since by Theorem 15

$$\mathsf{d}_{\mathbf{a}}(b, b') \leq \mathsf{d}(V, U) \leq t,$$

it follows that b can be found by inputting b' to a bounded-asymmetric-distance decoder for \mathcal{A} . Then the actual $c \in \mathcal{C}_b$ such that $V = \Lambda_b(c)$ can be found by using the decoder for \mathcal{C}_b described above.

6 Conclusions and Open Questions

Subspace codes represent an intriguing domain in which to carry out basic investigations of coding theory.

From a practical standpoint, at least for applications in network coding, the main problems appear to be solved, as constant-dimension lifted rank-metric codes contain close to the maximum possible number of codewords (at least on a logarithmic scale), and efficient encoding and decoding algorithms have been developed. It is unlikely that codes with marginally larger codebooks (even though they exist) will justify the additional complexity needed to process them.

From a mathematical standpoint, however, much remains open. For example, what are the optimal codes of minimum distance 2 or 3? Can existing constructions be improved? For example, the construction of lifted FD codes, which relies on a partitioning of $\mathcal{P}_q(n)$ into Schubert cells, can be regarded as a form of generalized concatenation. Are there other partitioning schemes that, for example, result in subsets of increasing minimum distance? Are there interesting subspace codes that can be constructed as orbits of a group action on vector spaces? Finally, are there additional applications of subspace codes beyond those of network coding and linear authentication?

References

1. Kötter, R., Kschischang, F.R.: Coding for errors and erasures in random network coding. *IEEE Trans. Inf. Theory* 54(8), 3579–3591 (2008)
2. Silva, D., Kschischang, F.R.: On metrics for error correction in network coding. *IEEE Trans. Inf. Theory* (to appear 2009)

3. Silva, D., Kschischang, F.R., Kötter, R.: A rank-metric approach to error control in random network coding. *IEEE Trans. Inf. Theory* 54(9), 3951–3967 (2008)
4. Silva, D., Kschischang, F.R.: Universal secure network coding via rank-metric codes. *IEEE Trans. Inf. Theory* (2008) (submitted for publication)
5. Katti, S., Katabi, D., Balakrishnan, H., Médard, M.: Symbol-level network coding for wireless mesh networks. In: *ACM SIGCOMM*, Seattle, WA (August 2008)
6. Wang, H., Xing, C., Safavi-Naini, R.: Linear authentication codes: bounds and constructions. *IEEE Trans. Inf. Theory* 49(4), 866–872 (2003)
7. Xia, S.T., Fu, F.W.: Johnson type bounds on constant dimension codes. *Designs, Codes and Cryptography* 50(2), 163–172 (2009)
8. Borel, A.: Linear Algebraic Groups. In: *Grad. Texts Math.*, 2nd edn., vol. 126. Springer, Heidelberg (1991)
9. Brouwer, A.E., Cohen, A.M., Neumaier, A.: *Distance-Regular Graphs*. Springer, New York (1989)
10. van Lint, J.H., Wilson, R.M.: *A Course in Combinatorics*, 2nd edn. Cambridge University Press, Cambridge (2001)
11. Gabidulin, E.M.: Theory of codes with maximum rank distance. *Probl. Inform. Transm.* 21(1), 1–12 (1985)
12. Delsarte, P.: Bilinear forms over a finite field, with applications to coding theory. *J. of Comb. Theory. Series A* 25, 226–241 (1978)
13. Loidreau, P.: Étude et optimisation de cryptosystèmes à clé publique fondés sur la théorie des codes correcteurs. Ph.d. dissertation, École Polytechnique, Paris, France (May 2001)
14. Gadouleau, M., Yan, Z.: Packing and covering properties of cdcs and subspace codes. *IEEE Trans. on Inform. Theory* (2008) (Submitted)
15. Delsarte, P.: An algebraic approach to association schemes of coding theory. *Philips J. Res.*, 1–97 (1973)
16. Frankl, P., Wilson, R.: The Erdős-Ko-Rado Theorem for Vector Spaces. *Journal of Combinatorial Theory* 43, 228–236 (1986)
17. Etzion, T., Vardy, A.: Error-correcting codes in projective space. In: *Proc. IEEE Int. Symp. Information Theory*, Toronto, Canada, July 6–11, pp. 871–875 (2008)
18. MacWilliams, F.J., Sloane, N.J.A.: *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam (1977)
19. Ahlswede, R., Aydinian, H.: On error control for random network coding. In: *IEEE Workshop on Network Coding, Theory and Applications* (2009)
20. Khaleghi, A., Kschischang, F.R.: Projective space codes for the injection metric. In: *Proc. 11th Canadian Workshop Inform. Theory*, Ottawa, May 13–15, pp. 9–12 (2009)
21. Tolhuizen, L.M.G.M.: The generalized Gilbert-Varshamov bound is implied by Túran's theorem. *IEEE Trans. Inf. Theory* 43(5), 1605–1606 (1997)
22. Silva, D.: Error Control for Network Coding. PhD thesis, University of Toronto, Toronto, Canada (2009)
23. Skachek, V.: Recursive code construction for random networks. Submitted for Publication (2008)
24. Manganelli, F., Gorla, E., Rosenthal, J.: Spread codes and spread decoding in network coding. In: *Proc. IEEE Int. Symp. Information Theory*, Toronto, Canada, July 6–11, pp. 881–885 (2008)

25. Gabidulin, E., Bossert, M.: Codes for network coding. In: Proc. IEEE Int. Symp. Information Theory, Toronto, Canada, July 6-11, pp. 867–870 (2008)
26. Etzion, T., Silberstein, N.: Error-correcting codes in projective spaces via rank-metric codes and Ferrers diagrams. *IEEE Trans. Inf. Theory* 55(7), 2909–2919 (2009)
27. Khaleghi, A.: Projective space codes for the injection metric. Master's thesis, University of Toronto, Toronto, Canada (2009)
28. Kohnert, A., Kurz, S.: Construction of large constant dimension codes with a prescribed minimum distance. Mathematical Methods in Computer Science: Essays in Memory of Thomas Beth, 31–42 (2008)
29. Silva, D., Kschischang, F.R.: Fast encoding and decoding of Gabidulin codes. In: Proc. IEEE Int. Symp. Information Theory, Seoul, Korea, June 28-July 3, pp. 2858–2862 (2009)

On Linear Programming Decoding on a Quantized Additive White Gaussian Noise Channel

Eirik Rosnes

The Selmer Center, Dept. of Informatics, University of Bergen
N-5020 Bergen, Norway
`eirik@ii.uib.no`

Abstract. In this work, we consider the pairwise error probability (PEP) of a linear programming (LP) decoder for a general binary linear code as formulated by Feldman *et al.* (IEEE Trans. Inf. Theory, March 2005) on a quantized additive white Gaussian noise (AWGN) channel. With a quantized AWGN (QAWGN) channel, we mean a channel where we first compute log-likelihood ratios as for an AWGN channel and then quantize them. Let \mathbf{H} be a parity-check matrix of a binary linear code and consider LP decoding based on \mathbf{H} . The output of the LP decoder is always a *pseudo-codeword*, of some *pseudo-weight*, where the definition of pseudo-weight is specific to the underlying channel model. In this work, we give a definition of pseudo-weight for a QAWGN channel based on an asymptotic (high signal-to-noise ratio) analysis of the PEP. Note that with maximum-likelihood decoding, the parameters of the quantization scheme, i.e., the quantization levels and the corresponding quantization region thresholds, that minimize the PEP of wrongly decoding to a non-zero codeword \mathbf{c} when the all-zero codeword is transmitted is independent of the specific codeword \mathbf{c} . However, this is not the case with LP decoding based on a parity-check matrix \mathbf{H} , which means that the quantization scheme needs to be optimized for the given \mathbf{H} . As a case study, we consider the well-known (3, 5)-regular (155, 64, 20) Tanner code and estimate its minimum QAWGN pseudo-weight with 3 and 5 levels of quantization, in which the quantization scheme is optimized to maximize the minimum QAWGN pseudo-weight.

1 Introduction

From the mid 1990's major research efforts have been devoted to the construction and performance analysis of capacity-approaching low-complexity codes, in particular turbo and low-density parity-check (LDPC) codes [1, 2, 3]. The existence of computationally efficient decoding algorithms for these capacity-approaching codes is the number one reason for their popularity. The performance of many of the most efficient graph-based algorithms, e.g., message-passing algorithms and decoding based on linear programming (LP) is crucially dependent on the existence of an efficient representation of the code in terms of a graphical model.

For the binary erasure channel (BEC), it is well-known that iterative belief-propagation (BP) decoding of LDPC codes can be characterized exactly in terms of *stopping sets* [4] in the chosen Tanner graph representation of the code. Thus, finding a Tanner graph containing no non-empty stopping sets of size less than the minimum distance of the code is important for improved performance under iterative BP decoding. Note that a binary linear code has many different parity-check matrices and hence many different Tanner graphs [5]. This reflects the property of message-passing algorithms that both the complexity and the performance are functions of the structure of the chosen parity-check matrix for the code.

Feldman *et al.* [6] recently considered the use of LP to decode binary linear codes. The obvious polytope for LP decoding is the convex hull of all codewords, in which case LP decoding is equivalent to maximum-likelihood (ML) decoding. However, the convex hull has a description complexity that is exponential in the codeword length for a general binary linear code. Thus, Feldman *et al.* [6] proposed a relaxed polytope which contains all valid codewords as vertices, but also additional non-codeword vertices. The vertices of the relaxed polytope are basically what the authors called *pseudo-codewords* in [6].

Recently, expressions for the *pseudo-weight* on the BEC, the binary symmetric channel (BSC), and the additive white Gaussian noise (AWGN) channel have been derived [7,8]. The pseudo-weight is the equivalent of the Hamming weight on the BEC and the BSC, and the equivalent of one quarter of the squared Euclidean distance on the AWGN channel when LP decoding and *not* ML decoding is performed. In a recent work [9], the pairwise error probability (PEP) of LP decoding on the independent Rayleigh flat-fading channel was considered, and it was shown that the PEP of wrongly decoding to a non-zero pseudo-codeword ω when the all-zero codeword is transmitted, behaves asymptotically as $K(\omega) \cdot (E_s/N_0)^{-|\chi(\omega)|}$, where $\chi(\omega)$ is the support set of ω , i.e., the set of non-zero coordinates, E_s/N_0 is the average signal-to-noise ratio (SNR), and $K(\omega)$ is a constant independent of the SNR.

In this work, however, we consider a quantized AWGN (QAWGN) channel. With a QAWGN channel, we mean a channel where we first computed log-likelihood ratios (LLRs) as for an AWGN channel and then quantize them. In particular, we give a definition of pseudo-weight for a QAWGN channel based on an asymptotic (high SNR) analysis of the PEP. Note that the effect of quantization and thresholding on the performance of the LP decoder has been treated in [10]. In fact, in [10], it was shown that for certain classes of LDPC codes (in particular, the Tanner graph of the LDPC code should be a good *expander*) and large enough SNR, it is advantageous to truncate the LLRs before passing them to the LP decoder.

2 Preliminaries

A binary linear code \mathcal{C} of length n and dimension k is a k -dimensional subspace of $\{0, 1\}^n$. The code can be specified as the null space of a rank $(n - k)$, $r \times n$

binary parity-check matrix \mathbf{H} where $r \geq n - k$. The Tanner graph $T = T(\mathbf{H})$ of \mathbf{H} is a bipartite graph composed of *left* (bit or variable) and *right* (check) vertices (or nodes) corresponding to codeword bits and parity-check constraints, respectively. All edges in T are incident on a left vertex and a right vertex, and there is an edge between the i th bit vertex and the j th check vertex if and only if the i th entry in the j th row of \mathbf{H} is equal to one.

A stopping set is a subset of the codeword positions such that the subgraph of the Tanner graph induced by the corresponding bit vertices has no check nodes of degree one. It is well-known that on the BEC iterative BP decoding is successful if and only if the set of erased positions does not contain a non-empty stopping set [4].

To construct a degree- q cover of $T(\mathbf{H})$, denoted by $T^{(q)}(\mathbf{H})$, we first make q identical copies of $T(\mathbf{H})$. Any permutation of the edges of the q copies of the Tanner graph $T(\mathbf{H})$ such that any edge e with left vertex $v = v(e)$ and right vertex $c = c(e)$ before the permutation, is incident on the left to one distinct vertex of the q copies of v , and on the right to one distinct vertex of the q copies of c after the permutation, will give a valid cover of $T(\mathbf{H})$. The corresponding cover code is denoted by $\mathcal{C}^{(q)}$ and is a function of both \mathbf{H} and the edge permutations. Define

$$\omega_l(\mathbf{c}^{(q)}) = \frac{|\{j : c_l^{(j)} = 1\}|}{q}, \quad l = 0, \dots, n-1$$

where $\mathbf{c}^{(q)} = (c_0^{(0)}, \dots, c_{n-1}^{(0)}, \dots, c_0^{(q-1)}, \dots, c_{n-1}^{(q-1)})$ is a codeword from $\mathcal{C}^{(q)}$. In this notation, $c_l^{(j)}$ corresponds to the j th copy of the l th bit vertex in the Tanner graph. Then, $\boldsymbol{\omega} = \boldsymbol{\omega}(\mathbf{c}^{(q)}) = (\omega_0(\mathbf{c}^{(q)}), \dots, \omega_{n-1}(\mathbf{c}^{(q)}))$ is a (graph-cover) pseudo-codeword [7]. We remark that a different definition of pseudo-codewords based on computation trees appeared in [8]. In this work, however, we will use the definition based on finite graph covers. The support set of a pseudo-codeword, as defined above, is a stopping set, and for any stopping set there is at least one pseudo-codeword (derived from a codeword in a suitable finite graph cover) with support set equal to the given stopping set [7, 11, 12]. Finally, note that the AWGN pseudo-weight of a non-zero pseudo-codeword $\boldsymbol{\omega}$ is defined as [7, 8]

$$w^{\text{AWGN}}(\boldsymbol{\omega}) = \frac{\left(\sum_{l=0}^{n-1} \omega_l\right)^2}{\sum_{l=0}^{n-1} \omega_l^2}. \quad (1)$$

Also, in the following, let $\kappa = \kappa(\boldsymbol{\omega})$ be the number of different non-zero values in the non-zero coordinates of the pseudo-codeword $\boldsymbol{\omega}$.

2.1 LP Decoding Based on the Parity-Check Matrix

Let the indices of the neighboring bit vertices of the j th check vertex in $T(\mathbf{H})$ constitute the set N_j and let

$$E_j = \{S : S \subseteq N_j \text{ and } |S| \text{ is even}\}.$$

Define the polytope

$$\mathcal{Q}_j = \left\{ (\mathbf{y}, \boldsymbol{\gamma}_j) \in [0, 1]^n \times [0, 1]^{2^{|N_j|}-1} : \sum_{S \in E_j} \gamma_{j,S} = 1 \text{ and } y_i = \sum_{S \in E_j : i \in S} \gamma_{j,S}, \forall i \in N_j \right\}$$

for all $j, j = 0, \dots, r-1$. Next, define the following projection onto the \mathbf{y} variables

$$\dot{\mathcal{Q}}_j = \left\{ \mathbf{y} \in [0, 1]^n : \exists \boldsymbol{\gamma}_j \in [0, 1]^{2^{|N_j|}-1} \text{ and } (\mathbf{y}, \boldsymbol{\gamma}_j) \in \mathcal{Q}_j \right\}$$

and the polytope

$$\dot{\mathcal{Q}} = \dot{\mathcal{Q}}_0 \cap \dot{\mathcal{Q}}_1 \cap \dots \cap \dot{\mathcal{Q}}_{r-1}. \quad (2)$$

LP decoding (on a binary-input memoryless channel) of binary linear codes as formulated by Feldman *et al.* [6, 13] can be described by the linear program

$$\text{minimize } \sum_{l=0}^{n-1} \lambda_l y_l \text{ subject to } \mathbf{y} \in \dot{\mathcal{Q}} \quad (3)$$

where

$$\lambda_l = \log \left(\frac{\Pr\{r_l | c_l = 0\}}{\Pr\{r_l | c_l = 1\}} \right), \quad l = 0, \dots, n-1,$$

c_l is the l th codeword bit, and r_l is the l th component of the received vector. If instead of $\dot{\mathcal{Q}}$ we use the convex hull of \mathcal{C} , then solving the linear program in (3) is equivalent to ML decoding.

The notion of a *proper* and \mathcal{C} -*symmetric* polytope was introduced in [6, 13], where the authors proved that the probability of error of LP decoding is independent of the transmitted codeword on a binary-input output-symmetric memoryless channel when the underlying code is linear¹ and the polytope is proper and \mathcal{C} -symmetric. A polytope $\dot{\mathcal{Q}}$ is proper if $\dot{\mathcal{Q}} \cap \{0, 1\}^n = \mathcal{C}$, and \mathcal{C} -symmetric if for any $\mathbf{y} \in \dot{\mathcal{Q}}$ and $\mathbf{c} \in \mathcal{C}$ it holds that $|\mathbf{y} - \mathbf{c}| \in \dot{\mathcal{Q}}$, where the notation $|\cdot|$ with a vector argument means the component-wise absolute value.

The set of points from the relaxed polytope $\dot{\mathcal{Q}}$, as defined in (2), where all entries in all points are rational numbers is equal to the set of all pseudo-codewords of all finite graph covers of the Tanner graph. Furthermore, all vertices of $\dot{\mathcal{Q}}$ have rational entries [7].

In a practical communication system, the LLRs are quantized. Let $\tilde{\lambda}_l$ be the quantized version of λ_l . Then, the decoding problem in (3) is modified to

$$\text{minimize } \sum_{l=0}^{n-1} \tilde{\lambda}_l y_l \text{ subject to } \mathbf{y} \in \dot{\mathcal{Q}}. \quad (4)$$

¹ We remark that the way the polytope $\dot{\mathcal{Q}}$ is defined in (2) the code is assumed to be linear in the setup.

3 PEP of LP Decoding on a QAWGN Channel

The PEP $\Pr\{\mathbf{c} \rightarrow \boldsymbol{\omega}\}$ of decoding to the pseudo-codeword $\boldsymbol{\omega}$ (within $\dot{\mathcal{Q}}$) when \mathbf{c} is transmitted (assuming LP decoding on a quantized binary-input memoryless channel) is

$$\Pr\{\mathbf{c} \rightarrow \boldsymbol{\omega}\} = \Pr \left\{ \sum_{l=0}^{n-1} (\omega_l - c_l) \tilde{\lambda}_l \leq 0 \right\}.$$

We will assume that the quantization scheme is symmetric, i.e., if λ_l is quantized to L , then $-\lambda_l$ is quantized to $-L$, where L is some real number. This implies that the corresponding binary-input memoryless channel with the quantized LLRs as output is output-symmetric. Thus, we can assume (without loss of generality, since the polytope in (2) is proper and \mathcal{C} -symmetric, and the channel is output-symmetric) the transmission of the all-zero codeword. When transmitting the all-zero codeword, the PEP $\Pr\{\mathbf{0} \rightarrow \boldsymbol{\omega}\}$ becomes

$$\Pr\{\mathbf{0} \rightarrow \boldsymbol{\omega}\} = \Pr \left\{ \sum_{l=0}^{n-1} \omega_l \tilde{\lambda}_l \leq 0 \right\}.$$

With two levels of quantization, we get the BSC. A pseudo-weight definition for the BSC appeared in [8]. Let t be the minimum number of coordinates in a pseudo-codeword $\boldsymbol{\omega}$ such that the sum of any t largest components in $\boldsymbol{\omega}$ is at least $(\sum_{l=0}^n \omega_l)/2$. Then, the BSC pseudo-weight of $\boldsymbol{\omega}$ is $2t$ if this sum of any t largest components in $\boldsymbol{\omega}$ is exactly $(\sum_{l=0}^n \omega_l)/2$, and $2t-1$, otherwise, i.e., if the sum exceeds $(\sum_{l=0}^n \omega_l)/2$ [8]. In the following, we will generalize the BSC definition of pseudo-weight to a QAWGN channel with $2M+1$ levels of quantization, where M is a positive integer. The approach considers the PEP at high SNR and is an extension of the derivation from the BSC case.

3.1 Three Levels of Quantization

With three levels of quantization, we get what is commonly referred to as a binary symmetric error and erasure channel (BSEC). In particular, we have

$$\tilde{\lambda}_l = \begin{cases} L, & \text{if } \Gamma \leq \lambda_l \\ 0, & \text{if } -\Gamma \leq \lambda_l < \Gamma \\ -L, & \text{otherwise} \end{cases}$$

for some non-negative real values Γ and L that may depend on the SNR and the underlying code. When the all-zero codeword is transmitted, the LLR λ_l is Gaussian distributed with mean $2/\sigma^2$ and variance $4/\sigma^2$ independently of the binary modulation, i.e., we have the same distribution both with binary antipodal signaling ($0 \rightarrow 1$ and $1 \rightarrow -1$) and with the mapping $0 \rightarrow -1$ and $1 \rightarrow 1$, from which it follows that

$$p^+ = \Pr\{\lambda_l \geq \Gamma \mid \mathbf{0} \text{ sent}\} = 1 - Q\left(\frac{-\Gamma\sigma^2 + 2}{2\sigma}\right) \geq 1 - \frac{1}{2} \exp\left(-\frac{(-\Gamma\sigma^2 + 2)^2}{8\sigma^2}\right)$$

$$p^- = \Pr\{\lambda_l < -\Gamma \mid \mathbf{0} \text{ sent}\} = Q\left(\frac{\Gamma\sigma^2 + 2}{2\sigma}\right) \leq \frac{1}{2} \exp\left(-\frac{(\Gamma\sigma^2 + 2)^2}{8\sigma^2}\right)$$

assuming that $-\Gamma\sigma^2 + 2 \geq 0$ in the first inequality, where

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp(-t^2/2) dt$$

is the Gaussian Q -function, and σ is the standard deviation of the AWGN. Setting $\Gamma = \alpha/\sigma^2$, we get

$$\begin{aligned} p^+ &= 1 - Q\left(\frac{-\alpha + 2}{2\sigma}\right) \geq 1 - \frac{1}{2} \exp\left(-\frac{\alpha^2 - 4\alpha + 4}{4} \cdot E_s/N_0\right) \\ p^- &= Q\left(\frac{\alpha + 2}{2\sigma}\right) \leq \frac{1}{2} \exp\left(-\frac{\alpha^2 + 4\alpha + 4}{4} \cdot E_s/N_0\right) \end{aligned}$$

where $E_s/N_0 = 1/(2\sigma^2)$ is the SNR.

Let $\boldsymbol{\omega}' = (\omega'_0, \dots, \omega'_{n-1})$ be a vector of length n with the same components as $\boldsymbol{\omega}$, but in non-increasing order. Define

$$f(\epsilon) = \begin{cases} \omega'_0, & \text{if } \epsilon \in (0, 1] \\ \omega'_1, & \text{if } \epsilon \in (1, 2] \\ \vdots \\ \omega'_{n-1}, & \text{if } \epsilon \in (n-1, n] \end{cases}$$

and $F(\epsilon) = \int_{\epsilon'=0}^{\epsilon} f(\epsilon') d\epsilon'$. Now, let

$$S =$$

$$\{(d_{-1}, d_0) : F(d_{-1}) \geq F(m) - F(d_{-1} + d_0), 0 \leq d_{-1}, d_0 \leq m \text{ and } d_{-1} + d_0 \leq m\}$$

where $m = |\chi(\boldsymbol{\omega})|$ and $\chi(\cdot)$ denotes the support set of its vector argument. Furthermore, define \tilde{S} as the maximum subset of S with the property that if $(d_{-1}, d_0) \in \tilde{S}$, then $(d, d_0) \notin S \forall d < d_{-1}$ and $(d_{-1}, d) \notin S \forall d < d_0$. With these definitions, a necessary condition for a non-zero probability for the event

$$\left\{ \sum_{l=0}^{n-1} \omega_l \tilde{\lambda}_l \leq 0 \right\}$$

is that \tilde{S} is non-empty. Since $(0, m) \in \tilde{S}$ and $(\lceil F^{-1}(F(m)/2) \rceil, 0) \in \tilde{S}$, \tilde{S} is always non-empty. Now, the PEP of decoding to the pseudo-codeword $\boldsymbol{\omega}$ when the all-zero codeword is transmitted can be written as

$$\Pr\{\mathbf{0} \rightarrow \boldsymbol{\omega}\} = \sum_{(d_{-1}, d_0) \in S} K(\boldsymbol{\omega}, d_{-1}, d_0) \cdot (p^-)^{d_{-1}} (1 - p^- - p^+)^{d_0} (p^+)^{m - d_{-1} - d_0}$$

where $K(\boldsymbol{\omega}, d_{-1}, d_0)$ is upper-bounded by the multinomial coefficient

$$\binom{m}{d_{-1}, d_0, m - d_{-1} - d_0} = \frac{m!}{d_{-1}! d_0! (m - d_{-1} - d_0)!}. \quad (5)$$

The exact value of the constant $K(\boldsymbol{\omega}, d_{-1}, d_0)$ depends on the number of times we can select d_{-1} non-zero coordinates $i_0, \dots, i_{d_{-1}-1}$ from $\boldsymbol{\omega}$, and then an additional $m - d_{-1} - d_0$ non-zero coordinates $i_{d_{-1}}, \dots, i_{m-d_0-1}$ such that

$$\sum_{l=0}^{d_{-1}-1} \omega_{i_l} - \sum_{l=d_{-1}}^{m-d_0-1} \omega_{i_l} \geq 0$$

assuming that $(d_{-1}, d_0) \in S$. In general, this number depends on $\boldsymbol{\omega}$ and is upper-bounded by the multinomial coefficient in (5) with equality when $\kappa = 1$. Since $p^+ \rightarrow 1$ as E_s/N_0 tends to infinity, we can define, based on the expressions above, a BSEC pseudo-weight as follows.

Definition 1. *The BSEC pseudo-weight of a pseudo-codeword $\boldsymbol{\omega}$ (for a given α) is defined as*

$$w^{(1)}(\boldsymbol{\omega}; \alpha) = \min_{(d_{-1}, d_0) \in \tilde{S}} \frac{1}{4} ((\alpha^2 + 4\alpha + 4)d_{-1} + (\alpha^2 - 4\alpha + 4)d_0) \quad (6)$$

where $0 \leq \alpha \leq 2$, since $-\Gamma\sigma^2 + 2 \geq 0$ and Γ is non-negative.

We remark that the minimization in (6) can be restricted to \tilde{S} (instead of S), since the coefficients of d_{-1} and d_0 both are non-negative. In Definition 1, the upper index (1) on the BSEC pseudo-weight is related to the number of quantization levels. In the general case, an upper index of (M) is used with $2M + 1$ quantization levels. Note that Definition 1 is based on an asymptotic (high SNR) argument. In fact, as the SNR tends to infinity, the PEP will approach $K(\boldsymbol{\omega}) \cdot \exp(-w^{(1)}(\boldsymbol{\omega}; \alpha) \cdot E_s/N_0)$, where $K(\boldsymbol{\omega})$ is a constant independent of the SNR, and the BSEC pseudo-weight is defined such that the exponent in the expression for the asymptotic PEP matches the exponent in the corresponding expression for the PEP for the AWGN channel. In the special case of $\alpha = 0$, we have two quantization levels and the BSEC pseudo-weight reduces to

$$w^{(1)}(\boldsymbol{\omega}; 0) = \min_{(d_{-1}, d_0) \in \tilde{S}} (d_{-1} + d_0) = \lceil F^{-1}(F(m)/2) \rceil$$

which is about half the BSC pseudo-weight $w^{\text{BSC}}(\boldsymbol{\omega})$, as defined in [8], where

$$w^{\text{BSC}}(\boldsymbol{\omega}) = \begin{cases} 2 \cdot \lceil F^{-1}(F(m)/2) \rceil, & \text{if } F^{-1}(F(m)/2) \text{ is an integer value} \\ 2 \cdot \lceil F^{-1}(F(m)/2) \rceil - 1, & \text{otherwise.} \end{cases}$$

The reason for this factor of two is that we consider a definition of pseudo-weight based on the PEP at high SNRs, while the BSC pseudo-weight definition in [8] is based on the ability to correct all error patterns with at most $\lceil w_{\min}^{\text{BSC}} / 2 \rceil - 1$ flips, where w_{\min}^{BSC} is the minimum BSC pseudo-weight, i.e., the minimum of $w^{\text{BSC}}(\boldsymbol{\omega})$ over all pseudo-codewords $\boldsymbol{\omega}$ different from the all-zero codeword.

Lemma 1. *The BSEC pseudo-weight $w^{(1)}(\omega; \alpha)$ with an optimal value for α is lower-bounded by half the BSC pseudo-weight, where an optimal value for α is an α -value that maximizes $w^{(1)}(\omega; \alpha)$.*

Proof. Let $\alpha = \hat{\alpha}$ denote an optimal value for α . Now,

$$w^{(1)}(\omega; \hat{\alpha}) \geq w^{(1)}(\omega; 0) = \lceil F^{-1}(F(m)/2) \rceil = \lceil w^{\text{BSC}}(\omega)/2 \rceil \geq w^{\text{BSC}}(\omega)/2$$

from which the result follows.

We remark that from the proof of Lemma 1, it follows that the minimum BSEC pseudo-weight (over all pseudo-codewords different from the all-zero codeword) optimized over α is lower-bounded by half the minimum BSC pseudo-weight. Also, analogous to the BSC case, one can make statements on the number of correctable errors and erasures with LP decoding on the BSEC. In fact, with LP decoding on the BSEC, one can correct t errors and e erasures if $F(t) < F(m) - F(t + e)$ for all non-zero pseudo-codewords ω , where $m = |\chi(\omega)|$. In particular, with $e = 0$ erasures, this statement reduces to the statement above for the BSC.

Example 1. When $\kappa = 1$, i.e., when the number of different non-zero values in the non-zero coordinates of the pseudo-codeword is one, $\tilde{S} = \{(m/2, 0), (m/2 - 1, 2), \dots, (1, m - 2), (0, m)\}$ when m is even, and $\tilde{S} = \{((m + 1)/2, 0), ((m - 1)/2, 1), ((m - 3)/2, 3), \dots, (1, m - 2), (0, m)\}$ when m is odd. In Fig. 1, we have plotted

$$g(\alpha; d_{-1}, d_0) = (\alpha^2 + 4\alpha + 4)d_{-1} + (\alpha^2 - 4\alpha + 4)d_0 \quad (7)$$

as a function of α for all $(d_{-1}, d_0) \in \tilde{S}$ when $\kappa = 1$ and $m = 10$. From the figure, we observe that the optimal value for α (the α -value that maximizes the BSEC pseudo-weight $w^{(1)}(\omega; \alpha)$) is where all the curves intersect. Since $(m/2, 0) \in \tilde{S}$ and $(0, m) \in \tilde{S}$, it follows that an optimal value for α is a solution to the equation

$$(\alpha^2 + 4\alpha + 4)m/2 = (\alpha^2 - 4\alpha + 4)m$$

which reduces to $\alpha^2 - 12\alpha + 4 = 0$. The unique solution to this equation within the range $[0, 2]$ is $\alpha = 6 - 4\sqrt{2}$. In fact, this result holds in general for any m (both even and odd). The corresponding BSEC pseudo-weight is $(12 - 8\sqrt{2})m \approx 0.6863m$.

Example 2. Consider the pseudo-codeword $\omega = (1, 1/2, 1/10, 1, 1/2, 1/10, 1, 1/2)$. For this pseudo-codeword, $\tilde{S} = \{(0, 8), (1, 4), (2, 1), (3, 0)\}$ and the BSEC pseudo-weight with the optimal choice of quantization is 3.7513. Note that the four functions $g(\alpha; d_{-1}, d_0)$ with $(d_{-1}, d_0) \in \tilde{S}$ (see (7) in Example 1 for a definition), do not all intersect for a specific α -value, which means that the picture is not as clear as when $\kappa = 1$. In fact, the optimal α -value is $4 - 2\sqrt{3}$ which corresponds to the intersection of $g(\alpha; 1, 4)$ and $g(\alpha; 2, 1)$. The AWGN pseudo-weight (computed using (1)) is 5.8594, and the BSEC pseudo-weight with two levels of quantization ($\alpha = 0$) is 3.

From Examples 1 and 2, we observe that the optimal value for α depends on the actual pseudo-codeword. However, when $\kappa = 1$, the optimal value for α is independent of the pseudo-codeword, as shown in Example 1.

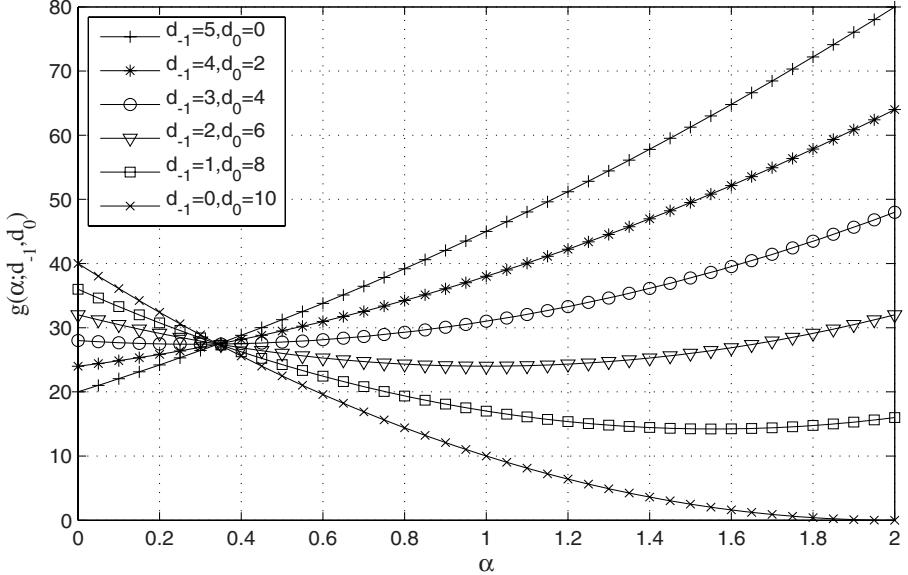


Fig. 1. The function $g(\alpha; d_{-1}, d_0) = (\alpha^2 + 4\alpha + 4)d_{-1} + (\alpha^2 - 4\alpha + 4)d_0$ versus α for all $(d_{-1}, d_0) \in \tilde{S}$ when $\kappa = 1$ and $m = 10$

3.2 $2M + 1$ Levels of Quantization

With $2M + 1$ levels of quantization, we have

$$\tilde{\lambda}_l = \begin{cases} L_M, & \text{if } \Gamma_M \leq \lambda_l \\ \vdots \\ L_1, & \text{if } \Gamma_1 \leq \lambda_l < \Gamma_2 \\ 0, & \text{if } -\Gamma_1 \leq \lambda_l < \Gamma_1 \\ -L_1, & \text{if } -\Gamma_2 \leq \lambda_l < -\Gamma_1 \\ \vdots \\ -L_M, & \text{if } \lambda_l < -\Gamma_M \end{cases}$$

for some non-negative real values $\Gamma_1, \dots, \Gamma_M$ and L_1, \dots, L_M , where $M \geq 1$, $\Gamma_M \geq \Gamma_{M-1} \geq \dots \geq \Gamma_1 \geq 0$, and $L_M \geq L_{M-1} \geq \dots \geq L_1 \geq 0$, that may depend on the SNR and the underlying code. Define

$$S_M = \left\{ (d_{-M}, \dots, d_{-1}, d_1, \dots, d_M) : L_M \sum_{l=0}^{d_{-M}-1} \omega'_l + \dots + L_1 \sum_{l=\sum_{i=2}^M d_{-i}}^{\sum_{i=1}^M d_{-i}-1} \omega'_l \right. \\ \left. \geq L_M \sum_{l=0}^{d_M-1} \omega''_l + \dots + L_1 \sum_{l=\sum_{i=2}^M d_i}^{\sum_{i=1}^M d_i-1} \omega''_l \right\}$$

under the conditions that $0 \leq d_{-i}, d_i \leq m$, $i = 1, \dots, M$, and $\sum_{i=1}^M d_{-i} + \sum_{i=1}^M d_i \leq m$, where $\boldsymbol{\omega}'' = (\omega_0'', \dots, \omega_{m-1}'')$ is a vector of length m with the non-zero components of $\boldsymbol{\omega}$ in a non-decreasing order. With this definition, a necessary condition for a non-zero probability for the event

$$\left\{ \sum_{l=0}^{n-1} \omega_l \tilde{\lambda}_l \leq 0 \right\}$$

is that S_M is non-empty. Since $(0, \dots, 0, 0, \dots, 0) \in S_M$, S_M is always non-empty. The probability that we have d_{-i} , $i = 1, \dots, M-1$, events of type $\{-\Gamma_{i+1} \leq \lambda_l < -\Gamma_i\}$ is $p_{-i}^{d_{-i}}$, d_i , $i = 1, \dots, M-1$, events of type $\{\Gamma_i \leq \lambda_l < \Gamma_{i+1}\}$ is $p_i^{d_i}$, d_{-M} events of type $\{\lambda_l < -\Gamma_M\}$ is $p_{-M}^{d_{-M}}$, d_M events of type $\{\Gamma_M \leq \lambda_l\}$ is $p_M^{d_M}$, and finally, $m - \sum_{i=1}^M d_{-i} - \sum_{i=1}^M d_i$ events of type $\{-\Gamma_1 \leq \lambda_l < \Gamma_1\}$ is $p_0^{m - \sum_{i=1}^M d_{-i} - \sum_{i=1}^M d_i}$, where

$$p_i = \begin{cases} Q\left(\frac{\Gamma_M \sigma^2 + 2}{2\sigma}\right), & i = -M \\ Q\left(\frac{\Gamma_{-i} \sigma^2 + 2}{2\sigma}\right) - Q\left(\frac{\Gamma_{-i+1} \sigma^2 + 2}{2\sigma}\right), & i = -M+1, \dots, -1 \\ Q\left(\frac{-\Gamma_1 \sigma^2 + 2}{2\sigma}\right) - Q\left(\frac{\Gamma_1 \sigma^2 + 2}{2\sigma}\right), & i = 0 \\ Q\left(\frac{-\Gamma_{i+1} \sigma^2 + 2}{2\sigma}\right) - Q\left(\frac{-\Gamma_i \sigma^2 + 2}{2\sigma}\right), & i = 1, \dots, M-1 \\ 1 - Q\left(\frac{-\Gamma_M \sigma^2 + 2}{2\sigma}\right), & i = M. \end{cases} \quad (8)$$

Define \tilde{S}_M as the maximum subset of S_M with the property that if

$$(d_{-M}, \dots, d_{-j}, \dots, d_{-1}, d_1, \dots, d_{M-1}, d_M) \in \tilde{S}_M$$

then

$$(d_{-M}, \dots, d, \dots, d_{-1}, d_1, \dots, d_{M-1}, \hat{d}) \notin S_M$$

for any $\hat{d} > d_M$ and $d < d_{-j}$ such that $d + \hat{d} = d_{-j} + d_M$, where $j = 1, \dots, M$, if

$$(d_{-M}, \dots, d_{-1}, d_1, \dots, d_j, \dots, d_{M-1}, d_M) \in \tilde{S}_M$$

then

$$(d_{-M}, \dots, d_{-1}, d_1, \dots, d, \dots, d_{M-1}, \hat{d}) \notin S_M$$

for any $\hat{d} > d_M$ and $d < d_j$ such that $d + \hat{d} = d_j + d_M$, where $j = 1, \dots, M-1$, and finally, if

$$(d_{-M}, \dots, d_{-1}, d_1, \dots, d_{M-1}, d_M) \in \tilde{S}_M$$

then

$$(d_{-M}, \dots, d_{-1}, d_1, \dots, d_{M-1}, d) \notin S_M$$

for any $d > d_M$. Setting $\Gamma_i = \alpha_i/\sigma^2$, $i = 1, \dots, M$, we can define (in the same way as for the special case of $M = 1$), based on the probabilities in (8), a QAWGN pseudo-weight as follows.

Definition 2. The QAWGN pseudo-weight of a pseudo-codeword ω (for given values of L_1, \dots, L_{M-1} and $\alpha_1, \dots, \alpha_M$) is defined as

$$\begin{aligned} w^{(M)}(\omega; L_1, \dots, L_{M-1}, \alpha_1, \dots, \alpha_M) \\ = \min_{(d_{-M}, \dots, d_{-1}, d_1, \dots, d_M) \in \tilde{S}_M} \frac{1}{4} \left(\sum_{i=1}^M (\alpha_i^2 + 4\alpha_i + 4)d_{-i} + \sum_{i=1}^M (\alpha_i^2 - 4\alpha_i + 4)d_{i-1} \right) \end{aligned} \quad (9)$$

where $d_0 = m - \sum_{i=1}^M d_{-i} - \sum_{i=1}^M d_i$ and $0 \leq \alpha_i \leq 2$, $i = 1, \dots, M$, since $-\Gamma_i \sigma^2 + 2 \geq 0$ and Γ_i is non-negative. Also, $\alpha_M \geq \alpha_{M-1} \geq \dots \geq \alpha_1$ and $L_M \geq L_{M-1} \geq \dots \geq L_1 \geq 0$.

We remark that the minimization in (9) can be restricted to \tilde{S}_M (instead of S_M), since the coefficients of $d_{-M}, \dots, d_{-1}, d_0, \dots, d_{M-1}$ are all non-negative. Furthermore, the value of L_M can be fixed to a real number greater than or equal to L_{M-1} due to scaling-invariance, i.e., the solution to the linear program in (4) remains the same even if the LLRs are scaled by any positive real constant. We remark that Definition 2 (as Definition 1 for three levels of quantization, i.e., for $M = 1$) is based on an asymptotic (high SNR) argument. In fact, as the SNR tends to infinity, the PEP will approach

$$K(\omega) \cdot \exp \left(-w^{(M)}(\omega; L_1, \dots, L_{M-1}, \alpha_1, \dots, \alpha_M) \cdot E_s / N_0 \right)$$

where $K(\omega)$ is a constant independent of the SNR. Note that the QAWGN pseudo-weight is defined such that the exponent in the expression for the asymptotic PEP matches the exponent in the corresponding expression for the PEP for the AWGN channel, which implies that the QAWGN pseudo-weight will approach the AWGN pseudo-weight as the number of quantization levels tends to infinity.

Lemma 2. The QAWGN pseudo-weight with optimal quantization levels and quantization thresholds is non-decreasing in the number of quantization levels, i.e., $\max_{L_1, \dots, L_{M-1}, \alpha_1, \dots, \alpha_M} w^{(M)}(\omega; L_1, \dots, L_{M-1}, \alpha_1, \dots, \alpha_M)$ is non-decreasing in $M \geq 1$.

Proof. Assume that $L_i = \hat{L}_i^{(M)}$, $i = 1, \dots, M-1$, and $\alpha_i = \hat{\alpha}_i^{(M)}$, $i = 1, \dots, M$, are optimal with the QAWGN pseudo-weight $w^{(M)}(\omega; L_1, \dots, L_{M-1}, \alpha_1, \dots, \alpha_M)$ with $2M+1$ levels of quantization. It follows that

$$\begin{aligned} & w^{(M+1)}(\omega; \hat{L}_1^{(M+1)}, \dots, \hat{L}_M^{(M+1)}, \hat{\alpha}_1^{(M+1)}, \dots, \hat{\alpha}_{M+1}^{(M+1)}) \\ & \geq w^{(M+1)}(\omega; \hat{L}_1^{(M)}, \dots, \hat{L}_{M-1}^{(M)}, \hat{L}_M^{(M)}, \hat{\alpha}_1^{(M)}, \dots, \hat{\alpha}_M^{(M)}, \hat{\alpha}_M^{(M)}) \\ & = w^{(M)}(\omega; \hat{L}_1^{(M)}, \dots, \hat{L}_{M-1}^{(M)}, \hat{\alpha}_1^{(M)}, \dots, \hat{\alpha}_M^{(M)}) \end{aligned}$$

from which the result follows.

From the proof of Lemma 2, it follows that the minimum QAWGN pseudo-weight (over all pseudo-codewords different from the all-zero codeword) with $2M + 1$ levels of quantization, optimized over the quantization parameters, is non-decreasing in $M \geq 1$.

Lemma 3. *The QAWGN pseudo-weight for any $M \geq 1$ with optimal quantization levels and quantization thresholds is upper-bounded by the BEC pseudo-weight.*

Proof. Since $(0, \dots, 0, 0, \dots, 0) \in \tilde{S}_M$, it follows from Definition 2 that the QAWGN pseudo-weight $w^{(M)}(\boldsymbol{\omega}; L_1, \dots, L_{M-1}, \alpha_1, \dots, \alpha_M)$ is upper-bounded by $1/4(\alpha_1^2 - 4\alpha_1 + 4)d_0 = 1/4(\alpha_1^2 - 4\alpha_1 + 4)m$. Optimizing over α_1 ($\alpha_1 = 0$ is an optimal α_1 -value) gives m , which is the BEC pseudo-weight.

Example 3. When $\kappa = 1$ and $M = 2$, the optimal value for α_1 can be shown to be a solution to the equation

$$x^4 + 8x^3 - 72x^2 + 672x - 112 = 0. \quad (10)$$

This equation results from the common intersection of the functions

$$g(\alpha_1, \alpha_2; d_{-2}, d_{-1}, d_1, d_2) = \sum_{i=1}^2 (\alpha_i^2 + 4\alpha_i + 4)d_{-i} + \sum_{i=1}^2 (\alpha_i^2 - 4\alpha_i + 4)d_{i-1}$$

with $(d_{-2}, d_{-1}, d_1, d_2) = (0, 0, 0, 0)$, $(m/2, 0, 0, m/2)$, and $(m/2 - 1, 1, 1, m/2 - 1)$, where $d_0 = m - d_{-2} - d_{-1} - d_1 - d_2$ and $m = |\chi(\boldsymbol{\omega})|$ is even. Note that $(0, 0, 0, 0)$, $(m/2, 0, 0, m/2)$, and $(m/2 - 1, 1, 1, m/2 - 1)$ are always in \tilde{S}_M , independently of the values for L_1, \dots, L_M . In more detail, setting $g(\alpha_1, \alpha_2; m/2, 0, 0, m/2) = g(\alpha_1, \alpha_2; m/2 - 1, 1, 1, m/2 - 1)$ and $g(\alpha_1, \alpha_2; 0, 0, 0, 0) = g(\alpha_1, \alpha_2; m/2, 0, 0, m/2)$, results in

$$\alpha_1^2 + 4\alpha_1 + 4 = 8\alpha_2 \text{ and } \alpha_2^2 + 4\alpha_2 + 4 = 2\alpha_1^2 - 8\alpha_1 + 8$$

respectively. Now, setting $\alpha_2 = (\alpha_1^2 + 4\alpha_1 + 4)/8$ (from the first equation) into the second equation, yields the equation in (10). The only solution of (10) between 0 and 2 is $x = 0.1697$, which means that the optimal value for α_1 is 0.1697. Furthermore, the optimal value for α_2 is $\alpha_2 = (\alpha_1^2 + 4\alpha_1 + 4)/8 = 0.5884$. An optimal value for L_1 with $L_2 = 1$ is 0.4, and the corresponding QAWGN pseudo-weight is 0.8375m. Note that different values for L_1 will give different sets \tilde{S}_M , and for the (α_1, α_2) pair corresponding to the intersection of the three functions $g(\alpha_1, \alpha_2; 0, 0, 0, 0)$, $g(\alpha_1, \alpha_2; m/2, 0, 0, m/2)$, and $g(\alpha_1, \alpha_2; m/2 - 1, 1, 1, m/2 - 1)$ to be the optimal pair (in terms of maximizing the QAWGN pseudo-weight), L_1 should be appropriately chosen.

Example 4. Consider the pseudo-codeword $\boldsymbol{\omega} = (1, 1/2, 1/10, 1, 1/2, 1/10, 1, 1/2)$. For this pseudo-codeword, the QAWGN pseudo-weight with the optimal choice of quantization with $M = 2$ (α_1 , α_2 , and L_1 need to be optimized) is 4.633. The AWGN pseudo-weight (computed using (1)) is 5.8594, and the BEC pseudo-weight with the optimal choice of quantization is 3.7513 (see Example 2).

4 Case Study

In this section, we present a case study of the $(3, 5)$ -regular $(155, 64, 20)$ Tanner code from [14]. The minimum BSC pseudo-weight for this code is 9 (which means that the minimum QAWGN pseudo-weight with two levels of quantization is 5) [15], and the minimum AWGN pseudo-weight is upper-bounded by 16.4037 [16]. The exact value for the minimum AWGN pseudo-weight is not known, but the upper bound is believed to be close to the true value. Also, the first few terms of the stopping set enumerator is known exactly for this code [17]. On the BEC, the BEC pseudo-weight is the size of the support set of the pseudo-codeword, which is a stopping set. Thus, the minimum BEC pseudo-weight is equal to the minimum stopping set size, or the stopping distance [5].

We have modified the pseudo-codeword/*instanton* search algorithm outlined in [15], developed for the BSC, to a QAWGN channel with $2M + 1$ levels of quantization. For instance, for the BSC, the search algorithm from [15] is initialized with a random LLR vector $\tilde{\lambda}$ containing the values 1 and -1 . The number of coordinates with value -1 in $\tilde{\lambda}$ should be large enough to make the LP decoder decode it into a pseudo-codeword different from the all-zero codeword. However, in the case with $2M + 1$ quantization levels, the coordinates of the initial LLR vector $\tilde{\lambda}$ should be initialized with the different possible quantization levels in a random fashion. As for the BSC, the number of coordinates with value different from $L_M = 1$ in $\tilde{\lambda}$ should be large enough to make the LP decoder decode it into a pseudo-codeword different from the all-zero codeword. Also, the concept of a *median* of a pseudo-codeword ω , as defined in [15], needs to be redefined.

Let

$$(\hat{d}_{-M}, \dots, \hat{d}_{-1}, \hat{d}_1, \dots, \hat{d}_M) \\ = \arg \min_{(d_{-M}, \dots, d_{-1}, d_1, \dots, d_M) \in \tilde{S}_M} \frac{1}{4} \left(\sum_{i=1}^M (\alpha_i^2 + 4\alpha_i + 4)d_{-i} + \sum_{i=1}^M (\alpha_i^2 - 4\alpha_i + 4)d_{i-1} \right).$$

Furthermore, let π be a coordinate permutation induced by the ordering of the components in ω from the highest to the lowest value, i.e., $\omega_{\pi(i)} = \omega'_i$, $i = 0, \dots, n - 1$. A *median* $\psi = \psi(\omega)$ of a non-zero pseudo-codeword ω is a $(2M + 1)$ -ary vector of length n with the property that

- $\psi_{\pi(i)} = -L_j$, $i = \sum_{l=0}^{M-j-1} \hat{d}_{-M+l}, \dots, \sum_{l=0}^{M-j} \hat{d}_{-M+l} - 1$ and $j = 1, \dots, M$,
- $\psi_{\pi(i)} = 0$, $i = \sum_{l=0}^{M-1} \hat{d}_{-M+l}, \dots, \sum_{l=0}^M \hat{d}_{-M+l} - 1$,
- $\psi_{\pi(i)} = L_{j-M+1}$, $i = \sum_{l=0}^j \hat{d}_{-M+l}, \dots, \sum_{l=0}^{j+1} \hat{d}_{-M+l} - 1$ and $j = M, \dots, 2M - 2$, and
- $\psi_i = L_M = 1$ for all other values of i .

Note that the LP decoder will fail to decode to the all-zero codeword when the quantized LLR vector $\tilde{\lambda}$ is equal to the median $\psi(\omega)$ of a non-zero pseudo-codeword ω , since $\sum_{l=0}^n \omega_l \tilde{\lambda}_l \leq 0$. Now, the algorithm proceeds as described in [15] by running the LP decoder several times in a row with an input that depends on the output at the previous iteration. With an input to the LP decoder, we

mean a quantized LLR vector. In general, the input at the current iteration is either the median of the pseudo-codeword at the output at the previous iteration, or a subset of it. The algorithm is initialized with a random LLR vector as described above, which makes it a random algorithm. We have run the algorithm a large number of times to increase the accuracy of the minimum QAWGN pseudo-weight estimation.

In Fig. 2, we have plotted the minimum BSEC pseudo-weight (estimated by the modified algorithm from [15]) as a function of the quantization parameter α . For small values of α (i.e., for values of α smaller than the optimal α -value), pseudo-codewords of support set size 44 and with $\kappa = 18$ have minimum BSEC pseudo-weight, while for larger values of α (i.e., for values of α larger than the optimal α -value), pseudo-codewords of support set size 18 and with $\kappa = 1$ have minimum BSEC pseudo-weight. We remark that the support sets of these pseudo-codewords with $\kappa = 1$ are minimum-size stopping sets [17]. This is due to the fact that when α increases, the BSEC looks more like an erasure channel, and the minimum BEC pseudo-weight is equal to the stopping distance.

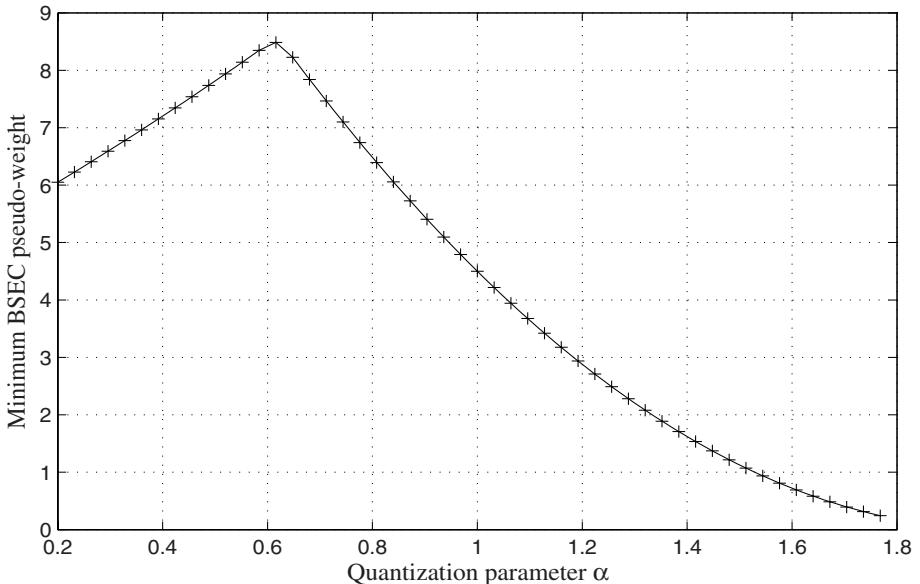


Fig. 2. Estimated minimum BSEC pseudo-weight for the (3, 5)-regular (155, 64, 20) Tanner code from [14] as a function of the quantization parameter α .

With 5 levels of quantization, the minimum QAWGN pseudo-weight is 11.33 (estimated by the modified algorithm from [15]) when optimal quantization parameters are used. The optimization is carried out over the parameters α_1 , α_2 , and L_1 . As stated above, the value of L_2 can be fixed to any real number greater than or equal to L_1 due to scaling-invariance. Note that when using the optimal

parameters for $\kappa = 1$, i.e., $\alpha_1 = 0.1697$, $\alpha_2 = 0.5884$, $L_1 = 0.4$, and $L_2 = 1$ (see Example 3), the minimum QAWGN pseudo-weight is as small as 8.37 (estimated by the modified algorithm from [15]).

5 Conclusion

In this work, we have analyzed the PEP of LP decoding for a general binary linear code on a QAWGN channel and given a definition of pseudo-weight for this channel based on an asymptotic (high SNR) analysis of the PEP. With ML decoding, the parameters of the quantization scheme, i.e., the quantization levels and the corresponding quantization region thresholds, that minimize the PEP of wrongly decoding to a non-zero codeword \mathbf{c} when the all-zero codeword is transmitted is independent of the specific codeword \mathbf{c} . With LP decoding based on a parity-check matrix \mathbf{H} , this is not true, and the quantization scheme needs to be optimized for the given \mathbf{H} . As an example, we studied the well-known (3, 5)-regular (155, 64, 20) Tanner code and estimated its minimum QAWGN pseudo-weight with 3 and 5 levels of quantization, in which the quantization scheme was optimized to maximize the minimum QAWGN pseudo-weight.

References

- Berrou, C., Glavieux, A., Thitimajshima, P.: Near Shannon limit error-correcting coding and decoding: Turbo-codes. In: Proc. IEEE Int. Conf. Commun. (ICC), Geneva, Switzerland, pp. 1064–1070 (May 1993)
- Gallager, R.G.: Low-density parity-check codes. IRE Trans. Inf. Theory 8(1), 21–28 (1962)
- MacKay, D.J.C.: Good error-correcting codes based on very sparse matrices. IEEE Trans. Inf. Theory 45(2), 399–431 (1999)
- Di, C., Proietti, D., Telatar, I.E., Richardson, T.J., Urbanke, R.L.: Finite-length analysis of low-density parity-check codes on the binary erasure channel. IEEE Trans. Inf. Theory 48(6), 1570–1579 (2002)
- Schwartz, M., Vardy, A.: On the stopping distance and the stopping redundancy of codes. IEEE Trans. Inf. Theory 52(3), 922–932 (2006)
- Feldman, J., Wainwright, M.J., Karger, D.R.: Using linear programming to decode binary linear codes. IEEE Trans. Inf. Theory 51(3), 954–972 (2005)
- Vontobel, P.O., Koetter, R.: Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes. IEEE Trans. Inf. Theory (to appear), <http://arxiv.org/abs/cs.IT/0512078/>
- Forney Jr., G.D., Koetter, R., Kschischang, F.R., Reznik, A.: On the effective weights of pseudocodewords for codes defined on graphs with cycles. In: Marcus, B., Rosenthal, J. (eds.) Codes, Systems, and Graphical Models. IMA Vol. Math. Appl., vol. 123, pp. 101–112. Springer, Heidelberg (2001)
- Rosnes, E.: On the pairwise error probability of linear programming decoding on independent Rayleigh flat-fading channels. IEEE Trans. Inf. Theory 55(7), 2942–2955 (2009)
- Feldman, J., Koetter, R., Vontobel, P.O.: The benefit of thresholding in LP decoding of LDPC codes. In: Proc. IEEE Int. Symp. Inf. Theory (ISIT), Adelaide, SA, Australia, pp. 307–311 (September 2005)

11. Kelley, C.A., Sridhara, D.: Pseudocodewords of Tanner graphs. *IEEE Trans. Inf. Theory* 53(11), 4013–4038 (2007)
12. Rosnes, E.: On the connection between finite graph covers, pseudo-codewords, and linear programming decoding of turbo codes. In: Proc. 4th Int. Symp. Turbo Codes & Related Topics, Munich, Germany (April 2006)
13. Feldman, J.: Decoding Error-Correcting Codes via Linear Programming. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA (2003)
14. Tanner, R.M., Sridhara, D., Fuja, T.: A class of group-structured LDPC codes. In: Proc. Int. Symp. Commun. Theory and Appl (ISCTA), Ambleside, UK (July 2001)
15. Chilappagari, S.K., Chertkov, M., Vasic, B.: Provably efficient instanton search algorithm for LP decoding of LDPC codes over the BSC. *IEEE Trans. Inf. Theory* (submitted for publication) (2008), <http://arxiv.org/abs/0808.2515/>
16. Chertkov, M., Stepanov, M.G.: An efficient pseudocodeword search algorithm for linear programming decoding of LDPC codes. *IEEE Trans. Inf. Theory* 54(4), 1514–1520 (2008)
17. Rosnes, E., Ytrehus, Ø.: An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices. *IEEE Trans. Inf. Theory* 55(9), 4167–4178 (2009)

Codes as Modules over Skew Polynomial Rings

Delphine Boucher and Felix Ulmer

Université de Rennes 1; IRMAR, CNRS, UMR 6625;
Université européenne de Bretagne

Abstract. In previous works we considered codes defined as *ideals* of quotients of *skew* polynomial rings, so called Ore rings of automorphism type. In this paper we consider codes defined as *modules* over skew polynomial rings, removing therefore some of the constraints on the length of the skew codes defined as ideals. The notion of BCH codes can be extended to this new approach and the skew codes whose duals are also defined as modules can be characterized. We conjecture that self-dual skew codes defined as modules must be constacyclic and prove this conjecture for the Hermitian scalar product and under some assumptions for the Euclidean scalar product. We found new [56, 28, 15], [60, 30, 16], [62, 31, 17], [66, 33, 17] Euclidean self-dual skew codes and new [50, 25, 14], [58, 29, 16] Hermitian self-dual skew codes over \mathbf{F}_4 , improving the best known distances for self-dual codes of these lengths over \mathbf{F}_4 .

1 Introduction

Starting from the finite field \mathbf{F}_q and an automorphism θ of \mathbf{F}_q , one defines a ring structure on the set:

$$R = \mathbf{F}_q[X, \theta] = \{a_n X^n + \dots + a_1 X + a_0 \mid a_i \in \mathbf{F}_q \text{ and } n \in \mathbf{N}\}.$$

The addition in R is defined to be the usual addition of polynomials and the multiplication is defined by the basic rule $X a = \theta(a) X$ ($a \in \mathbf{F}_q$) and extended to all elements of R by associativity and distributivity (cf. [1,9,10]). The ring R is a left and right Euclidean ring whose left and right ideals are principal [10].

In [3,5] we studied codes $(g)/(f) \subset R/(f)$ that are ideals of quotient rings. Those skew codes are completely defined by g and the degree of f , but the fact that the code has the structure of an R -ideal is linked to arithmetic properties of g . In particular g is a right divisor in R of a polynomial f which generates a two-sided ideal in R . This puts restrictions on the length of the code generated by g . In section 2 we will define skew codes $Rg/Rf \subset R/Rf$ that are R -submodules of a quotient module. Here f can be any left multiple of g and therefore any g is a valid generator. In section 3, we will show that these skew codes defined as modules enable to improve distances of skew codes defined as ideals and that the BCH approach can be generalized to skew codes defined as modules with prescribed distances. As there is no more restriction on the length of the code, we can construct BCH skew codes whose lengths could not be reached by BCH skew codes defined as ideals. Lastly, we will focus on self-dual skew codes defined

as modules. In [5] we showed that the (Euclidean and Hermitian) dual of a skew cyclic code defined as an ideal of the quotient ring $R/(X^n - 1)$ with generator polynomial g is again a skew cyclic code defined as an ideal of the quotient ring $R/(X^n - 1)$ and that the generator g^\perp of the dual code is determined by the factor h of the decomposition $hg = gh = X^n - 1$. This allows to use polynomial systems to compute self-dual codes over \mathbf{F}_q . In section 4, under the minor assumption that the constant term of g is $\neq 0$, we show that the dual of a skew code Rg/Rf is again a skew code defined as a module if and only if g is a left factor in R of some $X^n - c$ ($c \in \mathbf{F}_q - \{0\}$) and that the generator g^\perp of the dual code is determined by the right factor h of the decomposition $gh = X^n - c$ (here g is on the left). Using this result we are able to characterize all the skew codes defined as ideals (not necessarily cyclic) whose duals are again defined as ideals. We conjecture that self-dual skew codes defined as modules must be constacyclic and prove this conjecture for the Hermitian scalar product and under some assumptions for the Euclidean scalar product. Following [5], we use polynomial systems to compute self-dual skew codes over \mathbf{F}_4 and we find new [56, 28, 15], [60, 30, 16], [62, 31, 17], [66, 33, 17] Euclidean self-dual skew codes and new [50, 25, 14], [58, 29, 16] Hermitian self-dual skew codes over \mathbf{F}_4 , improving the best known distances for self-dual codes of these lengths over \mathbf{F}_4 .

2 Coding with Skew Polynomial Rings

2.1 Ideal θ -Codes

In [5] we defined codes as ideals of quotient rings of R . If $I = (f)$ is a two-sided ideal of R , then, in analogy to classical cyclic codes, we associate to an element $a(X) = \sum_{i=0}^{n-1} a_i X^i$ in $R/(f)$ the ‘word’ $a = (a_0, a_1, \dots, a_{n-1}) \in \mathbf{F}_q^n$.

Definition 1 (cf. [5]). *Let $f \in R$ be of degree n . If $I = (f)$ is a two-sided ideal of R , then an **ideal¹ θ -code** \mathcal{C} is a left ideal $(g)/(f) \subset R/(f)$, where $g \in R$ is a right divisor of f in R . Let us assume that the order of θ divides n , then*

1. *if $f = X^n - c$ with $c \in \mathbf{F}_q^\theta$, we call the ideal θ -code corresponding to the left ideal $(g)/(X^n - c) \subset R/(X^n - c)$ an **ideal θ -constacyclic code**;*
2. *if $f = X^n - 1$, we call the ideal θ -code corresponding to the left ideal $(g)/(X^n - 1) \subset R/(X^n - 1)$ an **ideal θ -cyclic code**.*

The length of the code is $n = \deg(f)$ and its dimension is $k = \deg(f) - \deg(g)$, we say that the code \mathcal{C} is of type $[n, k]_q$. If the distance of the code is d , then we say that the code \mathcal{C} is of type $[n, k, d]_q$.

An ideal θ -cyclic code \mathcal{C} has the following property ([3], Theorem 1)

$$(a_0, a_1, \dots, a_{n-1}) \in \mathcal{C} \quad \Rightarrow \quad (\theta(a_{n-1}), \theta(a_0), \theta(a_1), \dots, \theta(a_{n-2})) \in \mathcal{C}.$$

¹ In previous work we called those codes simply θ -codes, but we added *ideal* in the definition in order to distinguish those codes from the module codes which we will introduce in the next section.

If θ is not the identity, then the non commutative ring R is not a unique factorization ring and there are much more right factors of $f \in R$ than in the commutative case, leading to a huge number of linear codes that are not cyclic codes (cf. [3,5]).

EXAMPLE. Let α be a generator of the multiplicative group of \mathbf{F}_4 and θ the Frobenius automorphism given by $\theta(a) = a^2$. The polynomial $X^2 + \alpha^2 X + \alpha$ is a right divisor of $X^4 - 1 \in \mathbf{F}_4[X, \theta]$ so it generates a $[4, 2]_4$ ideal θ -cyclic code. Note that there are seven different monic right factors of degree two of $X^4 - 1$ in $\mathbf{F}_4[X, \theta]$ ([3], Example 2). ■

In the following we denote $\mathbf{F}_q^\theta \subset \mathbf{F}_q$ the fixed field of θ . In order to generate a two-sided ideal of R , a monic polynomial f must be of the form $X^t \tilde{f}$ where \tilde{f} is a monic polynomial belonging to the center $\mathbf{F}_q^\theta[X^m]$ of R , where m is the order of θ . If f is in the center of R , then we call the ideal θ -code, corresponding to the left ideal $(g)/(f) \subset R/(f)$, an **ideal θ -central code** (cf [5]).

The length of an ideal θ -code is determined by the degree of f , while the code itself is given by the generator matrix

$$G = \begin{pmatrix} g_0 & \cdots & g_{r-1} & g_r & 0 & \cdots & 0 \\ 0 & \theta(g_0) & \cdots & \theta(g_{r-1}) & \theta(g_r) & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & & & & & & \\ 0 & \cdots & 0 & \theta^{n-r-1}(g_0) & \cdots & \theta^{n-r-1}(g_{r-1}) & \theta^{n-r-1}(g_r) \end{pmatrix}$$

depending only on g and n .

The restriction on the length is that f has to be a multiple of the bound g :

Definition 2 [8]. An element $P \in \mathbf{F}_q[X, \theta]$ is **bounded** if the left ideal (P) contains a two-sided ideal (P^*) . The monic polynomial P^* of minimal degree is the **bound** of P .

The degree of the bound g^* of g can be bounded in terms of the degree of g and the order of θ ([5], Lemma 10) namely $\deg(g^*) \leq \lambda \cdot \deg(g)$ where λ is at most $m \cdot [\mathbf{F}_q : \mathbf{F}_q^\theta]$. Over \mathbf{F}_4 , we proved that $\lambda = 2$ when $\theta \neq id$ (lemma 11 of [5]).

Any polynomial $g \in R$ of degree r always generates an ideal θ -code of length n if $n \geq \lambda \cdot r$. But if $n < \lambda \cdot r$, then those polynomials $g \in R$ of degree r for which $n < \deg(g^*)$ do not generate an ideal θ -code of length n . This restriction is the motivation in the following section to generalize the notion of ideal θ -codes to module θ -codes.

EXAMPLE. Consider $g = X^3 + \alpha^2 X^2 + \alpha X + 1 \in \mathbf{F}_4[X, \theta]$. Its bound $g^* = X^6 + 1$ is of degree 6. Therefore the above matrix G obtained from the coefficients of g will generate an ideal θ -code only if the length of the code is at least 6, i.e. if there are at least 3 rows in the above matrix. The use of modules instead of ideals will allow to consider also generator matrices with fewer rows. ■

However, the bound of $g \in \mathbf{F}_4[X, \theta]$ can also be of degree $< 2 \cdot \deg(g)$:

EXAMPLE. Consider $g = X^4 + X^3 + \alpha^2 X^2 + X + \alpha \in \mathbf{F}_4[X, \theta]$. Its bound $g^* = X^6 + X^4 + X^2 + 1$ is of degree $6 < 2 \cdot 4$. Therefore from the above matrix G obtained from the coefficients of g we can generate an ideal θ -code $(g)/(X g^*) \subset R/(X g^*)$ of length 7 whose minimum distance 4 is the best known distance for $[7, 3]_4$ linear codes. ■

2.2 Module θ -Codes

The goal of this section is to define skew codes as modules instead of ideals.

In the following we will consider left R -modules $_R M$, where $_R M$ is an additive group with a left scalar multiplication $_R M \times R \rightarrow _R M$ given by $(m, r) \mapsto r \cdot m$. Since R is left and right Euclidean, all left ideals of R are principal of the form Rf and are examples of left R -modules as well as the quotients R/Rf . The fact that R is a left and right Euclidean ring also implies a similar structure theorem than for finitely generated abelian groups. From ([1], Theorem 3.3.6) we get:

Theorem 1. *A finitely generated right R -module is isomorphic to*

$$R/Rf_1 \oplus R/Rf_2 \oplus \dots \oplus R/Rf_s \oplus R^r$$

where s and r are non negative integers and the f_i are non units of R with the property that f_i is a right divisor of f_{i+1} for $i \in \{1, \dots, s-1\}$.

In particular a left R -module is irreducible if and only if the module is isomorphic to R/Rf where f is irreducible in R . Note that $Rf \subset Rg$ if and only if g is a right factor of f . If $f = hg$, then Rg/Rf is a submodule of R/Rf which is cyclic and generated as a left R -module by $g + Rf$.

If $f = hg \in R$, then Rg/Rf and R/Rh are isomorphic as R -modules. For $\ell \in R$ the module $R/R\ell$ can be identified with the set of possible remainders of a right division by ℓ in R and is therefore a \mathbf{F}_q -vector space of dimension $\deg(\ell)$. Therefore the left R -submodule $Rg/Rf \subset R/Rf$ is a \mathbf{F}_q -vector subspace of dimension $\deg(h) = \deg(f) - \deg(g)$ of the \mathbf{F}_q -vector space R/Rf of dimension $\deg(f)$. Since a vector subspace of a finite dimensional \mathbf{F}_q -vector space is a linear code over \mathbf{F}_q , we obtain the following generalization of ideal θ -codes (cf. [5]):

Definition 3. *Let $f \in R$ be of degree n . A **module θ -code** \mathcal{C} is a left R -submodule $Rg/Rf \subset R/Rf$ where g is a right divisor of f in R . Furthermore,*

1. *if $f = X^n - c$, with $c \in \mathbf{F}_q$, we call the module θ -code corresponding to the left R -module $Rg/Rf \subset R/Rf$ a **module θ -constacyclic code**;*
2. *if $f = X^n - 1$, we call the module θ -code corresponding to the left module $Rg/Rf \subset R/Rf$ a **module θ -cyclic code**.*

The length of the code is $n = \deg(f)$ and its dimension is $k = \deg(f) - \deg(g)$, we say that the code \mathcal{C} is of type $[n, k]_q$. If the distance of the code is d , then we say that the code \mathcal{C} is of type $[n, k, d]_q$.

As usual, we identify codewords with the list of coefficients of the remainder of a right division by f in R . The elements of Rg/Rf are then all left multiples of $g = g_r X^r + \cdots + g_1 X + g_0$ and are of the form

$$\left(\sum_{i=0}^{\deg(f)-\deg(g)-1} b_i X^i \right) \cdot g$$

This shows that the generator matrix of the corresponding module θ -code of length $n = \deg(f)$ is given by the matrix G in the previous section.

Note that the code is defined uniquely by the generator polynomial g whose leading coefficient can be supposed to be one. Therefore a module θ -code of type $[n, k] = [n, n - \deg(g)]$ is defined by the $\deg(g) - 1$ coefficients of the monic polynomial g , and there are $q^{\deg(g)-1}$ such codes.

Since the restriction linked to the degree of the bound g^* of g no longer exists, there are more module θ -codes than ideal θ -codes. In particular any polynomial $g \in R$ is a right divisor of some polynomial f of degree $n \geq \deg(g)$, so for any $g \in R$ and any $n \geq \deg(g)$ the matrix G generates a module θ -code.

EXAMPLE. The previous polynomial $g = X^3 + \alpha^2 X^2 + \alpha X + 1 \in \mathbf{F}_4[X, \theta]$ with bound $g^* = X^6 + 1$ generates, via the matrix

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

a $[5, 2]_4$ module θ -code Rg/Rf over \mathbf{F}_4 where f is a left multiple of g with degree 5. This module θ -code is not an ideal θ -code and its minimum distance 4 matches the best known distance for $[5, 2]_4$ linear codes. ■

3 Distance Improvements by Using Modules versus Ideals

The table 1 illustrates the gain of using module θ -codes instead of just ideal θ -codes. In the table, n is the length of the module θ -codes over $\mathbf{F}_4 = \mathbf{F}_2(\alpha)$ and corresponds to the degree of f . The integer r is the degree of g (therefore $n - r$ is the dimension of the code). An entry C_d indicates that the best known linear $[n, n - r]_4$ code is of minimal distance d and can be found within the family of cyclic codes. An entry C_d^θ indicates that the best known linear $[n, n - r]_4$ code is of minimal distance d and can be found within the family of ideal θ -cyclic codes (the entry C_{ds}^θ indicates that there exists such a code which is Euclidean self-dual). An entry θ_d indicates that the best known linear $[n, n - r]_4$ code is of minimal distance d and can be found within the family of ideal θ -codes. An entry M_d indicates that the best known linear $[n, n - r]_4$ code is of minimal distance d and can be found within the family of module θ -codes. A negative entry $-j$ indicates that the best module θ -code has a distance $d - j$, where d is the distance of the best known linear $[n, n - r]_4$ code.

Table 1. Codes over \mathbf{F}_4 constructed using $R = \mathbf{F}_4[X, \theta]$, where $\text{id} \neq \theta \in \text{Aut}(\mathbf{F}_4)$

$n \setminus r$	2	3	4	5	6	7	8	9	10	11
3	M_3									
4	C_{3s}^θ	C_4								
5	-1	M_4	M_5							
6	C_2	C_4	C_4^θ	C_6						
7	θ_2	θ_3	θ_4	M_5	M_7					
8	C_2	C_3^θ	C_{4s}^θ	C_5^θ	C_6^θ	C_8				
9	θ_2	θ_3	θ_4	M_5	M_6	M_7	M_9			
10	C_2	θ_3	C_4^θ	C_5^θ	C_6^θ	θ_6	θ_8	C_{10}		
11	θ_2	θ_3	θ_4	θ_4	M_6	M_6	M_7	M_8	M_{11}	
12	C_2	θ_3	θ_4	C_4	C_{6s}^θ	C_6^θ	C_7^θ	C_8^θ	C_9^θ	C_{12}
13	θ_2	θ_3	θ_4	θ_4	θ_5	M_6	M_7	M_8	M_9	M_{10}
14	C_2	C_3^θ	C_4^θ	C_4	C_5^θ	C_{6s}^θ	C_7^θ	M_8	M_9	C_{10}^θ
15	θ_2	-1	-1	θ_4	θ_5	-1	-1	M_8	M_8	-1
16	C_2	-1	-1	C_4^θ	-1	-1	-1	C_8^θ	M_9	
17	θ_2	-1	-1	θ_4	-1	-1	-1	-1	M_8	M_9
18	C_2	-1	θ_3	θ_4	-1	-1	C_6^θ	-1	C_8^θ	-1
19	θ_2	-1	θ_3	θ_4	-1	-1	θ_6	θ_7	M_8	θ_8
20	C_2	-1	θ_3	θ_4	-1	-1	θ_6	C_7^θ	C_8^θ	C_8^θ
21	θ_2	-1	θ_3	θ_4	-1	θ_5	-1	-1	θ_7	θ_8
22	C_2	θ_2	θ_3	θ_4	θ_4	θ_5	-1	C_6^θ	C_7^θ	C_8

In the part of the table below the diagonal staircase, there is no restriction on the generators of an ideal θ -code due to the bound. Therefore module θ -codes will not improve ideal θ -codes in this lower part of the table.

EXAMPLE. The polynomial $g = X^9 + \alpha X^8 + X^7 + X^5 + \alpha^2 X^4 + \alpha X^2 + X + 1$ generates a module θ -code of any length ≥ 9 . As its bound is $X^{18} + X^{16} + X^{14} + X^{12} + X^{10} + X^6 + 1$, the module θ -code of length 14 generated by g is not an ideal θ -code. Its minimum distance is 8, which is the best known distance for $[14, 5]_4$ linear codes. Furthermore there is no $[14, 5]_4$ ideal θ -code reaching this best distance. In the table this code corresponds to the entry M_8 at row 14 and column $9 = 14 - 5$. ■

In [3,6] the BCH approach is generalized to the non-commutative case to construct codes of arbitrary length and prescribed distance. In the following, we show that this approach can be extended to the module θ -codes. The difference with the work in [3,6] is that the use of module θ -codes allows to remove the restriction on the length of the codes in terms of the bound of g ([5], Definition 9). We get the following definition derived from definition 5 of [6]:

Definition 4. Let $\theta \in \text{Aut}(\mathbf{F}_q)$ be given by $a \mapsto a^{q_0}$, δ be a positive integer, $q = q_0^t$ and β belong to a field extension $\mathbf{F}_{q_0^s}$ of $\mathbf{F}_q = \mathbf{F}_{q_0^t}$. A BCH module θ -code over \mathbf{F}_q with parameters δ and β of length n is a module θ -code of length n generated by the monic skew polynomial $g \in \mathbf{F}_q[X, \theta]$ of smallest degree such that g is right divisible in $\mathbf{F}_{q_0^s}[X, \theta]$ by $X - \beta^i$ for $i \in \{0, \dots, \delta - 1\}$.

For the construction of the generator polynomial g of a BCH module θ -code we can use the algorithm given in [6] Section 4. The decoding algorithm described in [3,6] also allows to decode skew BCH module codes. The proof of Proposition 2 of [6] can be adapted word for word in order to obtain

Proposition 1. Let \mathcal{C} be a BCH module θ -code with the notations of the above definition. If $n \leq (q_0 - 1) \cdot s$ and β is of order $q_0^s - 1$ then \mathcal{C} has minimum distance at least δ .

The next proposition improves the previous one for codes defined over $\mathbf{F}_q = \mathbf{F}_{2^t}$, showing that β does not need to be a generator of the field extension \mathbf{F}_{2^s} :

Proposition 2. Let \mathcal{C} be a BCH module θ -code with the notations of definition (cf. 4) and $q_0 = 2$. If the order of β is at least $2^n - 1$ then \mathcal{C} has a minimum distance at least δ .

PROOF. Let m be the order of β . Following the proof of proposition 2 of [6] or the proposition 2 of [3], the minimum distance of \mathcal{C} is at least δ if and only if for all $j < i < n$, $\beta^{(2^i - 2^j)/(2-1)} \neq 1$. Let us assume $\beta^{2^i - 2^j} = 1$, then the order m of β divides $2^i - 2^j$; as m divides $2^s - 1$, it cannot divide 2^j , so there exists $l < n$ such that m divides $2^l - 1$. As $l < n$ we get $m < 2^n - 1$. So if $m \geq 2^n - 1$ then the minimum distance of \mathcal{C} is at least δ . ■

The following examples show that there are more BCH module θ -codes than BCH ideal θ -codes.

EXAMPLE. Using modules we construct a $[10, 4, 6]_4$ BCH module θ -code (best possible distance). This code is obtained using the element $\beta = a^{11} \in \mathbf{F}_{2^{12}}$ of order $2^{12} - 1$ (where a is a generator of the multiplicative group of $\mathbf{F}_{2^{12}}$ used by MAGMA) by imposing a distance 2. The resulting generator polynomial is $g = X^6 + \alpha^2 X^5 + \alpha X^4 + \alpha X^2 + X + \alpha^2$ (where $\mathbf{F}_4 = \mathbf{F}_2(\alpha)$). The bound of g is $g^* = X^{12} + 1$, showing that the smallest length of an ideal θ -code with generator polynomial g is 12. This code improves the BCH ideal θ -codes as there exists no $[10, 4]$ BCH ideal θ -code constructed from $\mathbf{F}_{2^{12}}$ ([6]). ■

EXAMPLE. Using modules we construct the module θ -codes $[8, 3, 5]_4$ and $[8, 2, 6]_4$ respectively (best possible distances) obtained from $\mathbf{F}_{2^{12}} = \mathbf{F}_2(a)$ (where a is a generator of the multiplicative group of $\mathbf{F}_{2^{12}}$ used by MAGMA) by imposing a distance 2. No code with such length and dimension can be constructed using ideal θ -codes ([6])

1. In order to construct a $[8, 3, 5]_4$ module θ -code we used $\beta = a^{25} \in \mathbf{F}_{2^{12}}$ of order $819 \neq 2^{12} - 1$, $819 \geq 2^8 - 1$ to obtain $g = X^5 + \alpha^2 X^4 + \alpha^2 X^2 + \alpha X + \alpha^2 \in \mathbf{F}_4[X, \theta]$ whose bound is $g^* = X^{10} + X^8 + X^6 + X^4 + X^2 + 1$.

2. In order to construct a $[8, 2, 6]_4$ module θ -code, we used $\beta = a \in \mathbf{F}_{2^{12}}$ to obtain $g = X^6 + X^5 + \alpha^2 X^4 + X^3 + \alpha X^2 + \alpha^2 X + \alpha^2 \in \mathbf{F}_4[X, \theta]$ whose bound is $g^* = X^{12} + 1$. \blacksquare

4 Self-dual Module θ -Codes

The Euclidean and Hermitian dual of an ideal θ -cyclic code $(g)/(X^n - 1)$ is an ideal θ -cyclic code whose generator polynomial is determined by the factor h in the decomposition $X^n - 1 = hg = gh$ (cf. [5]). This allowed us in [5] to characterize self-dual codes by polynomial equations satisfied by the coefficients of g . However, if an ideal θ -code is not θ -cyclic then its dual may not be an ideal θ -code and until now we were not able to characterize those ideal θ -central codes who are not ideal θ -cyclic and whose duals are ideal θ -central codes.

In the following we give a characterization of the module θ -codes whose duals are module θ -codes for the Euclidean and Hermitian scalar products. It enables us to characterize those ideal θ -central codes whose duals are ideal θ -central codes. We also conjecture that a self-dual θ -code is necessarily a module θ -constacyclic code and prove this conjecture for the Euclidean scalar product under some assumptions and for the Hermitian scalar product. In particular, the self-dual module θ -codes are not necessarily ideal θ -codes.

Like in [5] we derive polynomial equations which characterize self-dual module θ -codes for both scalar products. Thanks to a refinement in the resolution of these polynomial equations we were able to find new $[56, 28, 15]_4$, $[60, 30, 16]_4$, $[62, 31, 17]_4$, $[66, 33, 17]_4$ Euclidean self-dual codes and new $[50, 25, 14]_4$, $[58, 29, 16]_4$ Hermitian self-dual codes which improve the best previously known distances for these codes. It turns out that all these codes are ideal θ -cyclic codes.

4.1 Dual for the Euclidean Scalar Product

The Euclidean dual \mathcal{C}^\perp of a code \mathcal{C} of \mathbf{F}_q^n is the set of words which are orthogonal to the code's words relatively to the Euclidean scalar product. We characterize those module θ -codes whose duals are module θ -codes, extending the corresponding result of [5] for ideal θ -cyclic codes.

In the following we will assume that the constant term of the generator polynomial g is $\neq 0$. This is not a strong restriction since if g is right divisible by X^s , then the resulting ideal θ -code has s coordinates which are always zeros and the resulting code is of little interest if $s > 0$ (cf. [5], Proposition 13).

Theorem 2 : Euclidean dual of a module θ -code. *Let $k \leq n$ be integers, $g \in \mathbf{F}_q[X, \theta]$ of degree $n - k$ with constant term $\neq 0$ and \mathcal{C} be the module θ -code of length n generated by g . The Euclidean dual \mathcal{C}^\perp of \mathcal{C} is a module θ -code generated by a polynomial of degree k with constant term $\neq 0$ if and only if there exist $h \in \mathbf{F}_q[X, \theta]$ and c in $\mathbf{F}_q - \{0\}$ such that $g h = X^n - c$ (here g is on the left).*

In this case the generator polynomial of \mathcal{C}^\perp is given by : $g^\perp = \sum_{i=0}^k \theta^i (h_{k-i}) X^i$ and g^\perp is a left divisor of $X^n - \theta^{k-n} (\frac{1}{c}) \in \mathbf{F}_q[X, \theta]$.

Note: In the examples we will test the existence of the above $\hbar \in \mathbf{F}_q[X, \theta]$ and c in $\mathbf{F}_q - \{0\}$ by verifying that the remainder of the left division of $X^n \in \mathbf{F}_q[X, \theta]$ by g is a non zero constant.

PROOF. Suppose that $X^n - g\hbar = c \neq 0$ in $\mathbf{F}_q[X, \theta]$ and define $g^\perp := \sum_{i=0}^k \theta^i(\hbar_{k-i}) X^i$. As $g_0\hbar_0 = c \neq 0$ and $g_0 \neq 0$, we must have $\hbar_0 \neq 0$. Therefore g^\perp is a polynomial of degree k and generates a module θ -code $\tilde{\mathcal{C}}$ of length n and dimension $n - k$. We now prove that $\tilde{\mathcal{C}} = \mathcal{C}^\perp$ by showing that the words of \mathcal{C} and $\tilde{\mathcal{C}}$ are orthogonal:

For $i_0 \in \{0, \dots, k-1\}$, $i_1 \in \{0, \dots, n-k-1\}$ we have $\langle X^{i_0} g, X^{i_1} g^\perp \rangle$

$$\begin{aligned} &= \left\langle \sum_{i=0}^{n-k} \theta^{i_0}(g_i) X^{i+i_0}, \sum_{i=0}^k \theta^{i_1}(\theta^i(\hbar_{k-i})) X^{i+i_1} \right\rangle \\ &= \left\langle \sum_{i=0}^{n-k} \theta^{i_0}(g_i) X^{i+i_0}, \sum_{i=i_1-i_0}^{i_1-i_0+k} \theta^{i+i_0}(\hbar_{k-i+i_1-i_0}) X^{i+i_0} \right\rangle \\ &= \sum_{i=\max(0, i_1-i_0)}^{\min(n-k, i_1-i_0+k)} \theta^{i_0}(g_i) \theta^{i+i_0}(\hbar_{k-i+i_1-i_0}) \\ &= \theta^{i_0} \left(\sum_{i=\max(0, l-k)}^{\min(n-k, l)} g_i \theta^i(\hbar_{l-i}) \right) \quad (\text{where } l = k + i_1 - i_0 \in \{1, \dots, n-1\}) \\ &= \theta^{i_0}((g\hbar)_l) \quad (\text{here } (g\hbar)_l \text{ denotes the coefficient of } X^l \text{ in } g\hbar) \\ &= 0 \quad (\text{because } g\hbar = X^n - c) \end{aligned}$$

Conversely, suppose that \mathcal{C}^\perp is a module θ -code generated by a polynomial \tilde{g} with constant term $\neq 0$. Define \hbar as $\hbar = \sum_{i=0}^k \theta^{i-k}(\tilde{g}_{k-i}) X^i \in \mathbf{F}_q[X, \theta]$. Since the constant term of \tilde{g} is $\neq 0$, the polynomial \hbar is of degree k . Then for all $i_0 \in \{0, \dots, k\}$, $i_1 \in \{0, \dots, n-k\}$,

$$\begin{aligned} 0 &= \langle X^{i_0} g, X^{i_1} \tilde{g} \rangle \\ &= \left\langle \sum_{i=0}^{n-k} \theta^{i_0}(g_i) X^{i+i_0}, \sum_{i=0}^k \theta^{i_1}(\tilde{g}_i) X^{i+i_1} \right\rangle \\ &= \left\langle \sum_{i=0}^{n-k} \theta^{i_0}(g_i) X^{i+i_0}, \sum_{i=i_1-i_0}^{k+i_1-i_0} \theta^{i_1}(\tilde{g}_{i-i_1+i_0}) X^{i+i_0} \right\rangle \\ &= \sum_{i=\max(0, i_1-i_0)}^{\min(n-k, i_1-i_0+k)} \theta^{i_0}(g_i) \theta^{i_1}(\tilde{g}_{i-i_1+i_0}) \end{aligned}$$

So for all $l \in \{1, \dots, n-1\}$ ($l = i_1 - i_0 + k$)

$$0 = \theta^{i_0} \left(\sum_{i=\max(0, l-k)}^{\min(n-k, l)} g_i \theta^{l-k}(\tilde{g}_{i+k-l}) \right) = \theta^{i_0} \left(\sum_{i=\max(0, l-k)}^{\min(n-k, l)} g_i \theta^i(\hbar_{l-i}) \right)$$

$$0 = \sum_{i=\max(0,l-k)}^{\min(n-k,l)} g_i \theta^i(\hbar_{l-i}) \quad (\text{the coefficient of } X^l \text{ in } g\hbar)$$

This shows that $g\hbar$ is of the form $X^n - c$ with $c \in \mathbf{F}_q$. Since $g_0\hbar_0 = g_0\theta^{-k}(\tilde{g}_k) \neq 0$ we have $c \neq 0$.

For the last assertion, denote $\mathbf{F}_q(X, \theta)$ the right field of fractions of $\mathbf{F}_q[X, \theta]$ and X^{-1} the inverse of X . We have $aX^{-1} = X^{-1}\theta(a)$ and $\sum_{i=0}^n a_i X^i \mapsto \sum_{i=0}^n X^{-i} a_i$ is an anti-morphism of $\mathbf{F}_q(X, \theta)$ (cf. proof of Lemma 17 in [5]). If $g\hbar = X^n - c$ then

$$X^k \varphi(\hbar) \varphi(g) X^{n-k} = X^k (1/X^n - c) X^{n-k} = 1 - \theta^k(c) X^n \in \mathbf{F}_q[X, \theta].$$

As $g^\perp = X^k \varphi(\hbar)$, we obtain that g^\perp is a left divisor of $1 - \theta^k(c) X^n \in \mathbf{F}_q[X, \theta]$ so it is a left divisor of $-(1 - \theta^k(c) X^n) \theta^{-n+k}(\frac{1}{c}) = X^n - \theta^{k-n}(\frac{1}{c})$. ■

When a module θ -code is generated by a polynomial satisfying the conditions of the theorem, one can deduce a nice expression for the parity check matrix of the code.

Corollary 1 : Parity check matrix. Let $k \leq n$ be integers, let $g \in \mathbf{F}_q[X, \theta]$ be of degree $n - k$ with constant term $\neq 0$. If there exist $c \in \mathbf{F}_q - \{0\}$ and $\hbar \in \mathbf{F}_q[X, \theta]$ such that $g\hbar = X^n - c$, then the parity check matrix of the module θ -code \mathcal{C} of length n generated by g is:

$$H = \begin{pmatrix} \hbar_k & \dots & \theta^{k-1}(\hbar_1) & \theta^k(\hbar_0) & 0 & \dots & 0 \\ 0 & \theta(\hbar_k) & \dots & \dots & \theta^{k+1}(\hbar_0) & \dots & 0 \\ 0 & \ddots & \ddots & & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \theta^{n-k-1}(\hbar_k) & \dots & \theta^{n-2}(\hbar_1) & \theta^{n-1}(\hbar_0) \end{pmatrix}$$

PROOF. The parity check matrix of \mathcal{C} is the matrix of the dual \mathcal{C}^\perp . As g satisfies the conditions of theorem 2, \mathcal{C}^\perp is a module θ -code with generator polynomial $\hbar_k + \dots + \theta^{k-1}(\hbar_1) X^{k-1} + \theta^k(\hbar_0) X^k$. ■

EXAMPLE

- Consider the polynomial $g = X^2 + \alpha X + 1 \in \mathbf{F}_4[X, \theta]$, where $\mathbf{F}_4 = \mathbf{F}_2(\alpha)$. Since $X^3 - g(X + \alpha) = \alpha \neq 0$ (the remainder of the left division of X^3 by g is a constant $\neq 0$), the proposition shows that the dual of the $[3, 1]_4$ module θ -code \mathcal{C} generated by g is a module θ -code generated by $g^\perp = 1 + \alpha^2 X$. The parity check matrix of \mathcal{C} is:

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

- Consider the polynomial $g = X^4 + \alpha^3 \in \mathbf{F}_8[X, \theta]$, where $\mathbf{F}_8 = \mathbf{F}_2(\alpha)$, $\alpha^3 + \alpha + 1 = 0$ and θ is the Frobenius automorphism. The polynomial g generates a $[8, 4]_8$ module θ -code \mathcal{C} which is not an ideal θ -code as the degree of its

bound, $X^{12} + 1$, is greater than 8. Since $X^8 - g(X^4 + \alpha^5) = \alpha \neq 0$, the dual of the code \mathcal{C} is a module θ -code generated by : $1 + \theta^4(\alpha^5)X^4 = 1 + \alpha^3X^4$ and the parity check matrix of \mathcal{C} is deduced from it. ■

In [5], we prove that the dual of an ideal θ -cyclic code is θ -cyclic. Using the previous theorem, we will now extend this result to ideal θ -constacyclic code (see also [4], Theorem 4.4 for skew codes defined as quotient ideals of $GR(4, 2)[X, \theta]$). In the proof we use that for a central element $X^n - c$ the right factor g in $hg = X^n - c$ also give a left factor $gh = X^n - c$ as needed in the theorem :

Corollary 2 : Euclidean dual of an ideal θ -cyclic code. *The dual code of an ideal θ -constacyclic code (with constant term of the generator $\neq 0$) is an ideal θ -constacyclic code.*

PROOF. The generator polynomial g of an ideal θ -constacyclic code of length n is a right divisor of $X^n - c \in R = \mathbf{F}_q[X, \theta]$, where $c \in \mathbf{F}_q^\theta$ and n is a multiple of the order of θ . Since $X^n - c$ belongs to the center of R , the polynomial g is also a left divisor of $X^n - c \in R$. According to theorem 2, the code \mathcal{C}^\perp is a module θ -code whose generator polynomial g^\perp is a left divisor of $X^n - \theta^{k-n}(\frac{1}{c}) = X^n - \frac{1}{c}$. Since $X^n - 1/c$ belongs to the center of R , g^\perp is also a right divisor of $X^n - 1/c$. The central polynomial $X^n - \frac{1}{c}$ generates a two-sided ideal of R , showing that \mathcal{C}^\perp is an ideal θ -constacyclic code. ■

The dual of an ideal θ -central code is not always an ideal θ -central code (examples 5 and 20 of [5]). The above proposition allows to characterize the ideal θ -central codes whose duals are again θ -central codes.

Corollary 3 : Euclidean dual of an ideal θ -central code. *Let $k \leq n$ be integers, let \mathcal{C} be an ideal θ -central code of length n generated by a polynomial g of degree $n - k$ and constant term $\neq 0$. The code \mathcal{C}^\perp is an ideal θ -central code if and only if*

1. $\exists \hbar \in \mathbf{F}_q[X, \theta]$ and a non zero constant c such that $X^n - c = g\hbar$ (here g is a left factor and c may not belong to \mathbf{F}_q^θ);
2. the degree of the bound $(g^\perp)^*$ of $g^\perp = \sum_{i=0}^k \theta^i(\hbar_{k-i}) X^i$ is $\leq n$.

In this case the ideal θ -central code \mathcal{C}^\perp is generated by g^\perp .

PROOF. A module θ -code with generator g^\perp and length n is a central θ -code if and only if the degree of the bound of g^\perp is $\leq n$. The result now follows from theorem 2. ■

EXAMPLE

1. The polynomial $g = X^3 + X^2 + X + \alpha \in \mathbf{F}_4[X, \theta]$ generates an ideal θ -central code (which is not θ -cyclic) of length 12 (examples 5 and 20 in [5]). Since the remainder $X^2 + \alpha^2 X + \alpha$ of the left division of X^{12} by g is not a constant, the code \mathcal{C}^\perp is not a module θ -code and therefore also not an ideal θ -central code.

2. The polynomial $g = X^2 + \alpha \in \mathbf{F}_4[X, \theta]$ generates a module θ -code \mathcal{C} of length 4. The bound of g is $g^* = X^4 + X^2 + 1$, showing that this code is an ideal θ -central code which is not θ -cyclic. Since the remainder α^2 of the left (and right in this case) division of X^4 by g is a non zero constant, the above proposition shows that the dual code is a module θ -code generated by $g^\perp = X^2 + \alpha^2$. As the bound of g^\perp is $X^4 + X^2 + 1$ is of degree ≤ 4 , the code \mathcal{C}^\perp is an ideal θ -central code.
3. The polynomial $g = X^2 + \alpha \in \mathbf{F}_4[X, \theta]$ also generates an ideal θ -central code of length 8 which is not θ -cyclic. Since the remainder α of the left (and right in this case) division of X^8 by g is a non zero constant, the above proposition shows that the dual code is a module θ -code generated by $g^\perp = X^6 + \alpha^2 X^4 + \alpha X^2 + 1$. As the bound of $g^\perp = X^{12} + X^{10} + X^6 + X^2 + 1$ is of degree > 8 , the code \mathcal{C}^\perp is a module θ -code which is not an ideal θ -central code. ■

A code is said to be self-dual if it is equal to its dual. Following [5], we characterize Euclidean self-dual module θ -codes with a system of polynomial equations.

Corollary 4 : Euclidean self-dual module θ -codes. *Let k be an integer and $g = \sum_{i=0}^k g_i X^i \in \mathbf{F}_q[X, \theta]$ be monic of degree k and $g_0 \neq 0$. Denote \mathcal{C} the module θ -code of length $2k$ generated by g . The code \mathcal{C} is Euclidean self-dual if and only if*

$$\forall l \in \{1, \dots, k\}, \quad \sum_{i=0}^l \theta^{k-l}(g_i) g_{i+k-l} = 0 \quad (1)$$

PROOF. The module θ -code \mathcal{C} is self-dual if, and only if, \mathcal{C}^\perp is a module θ -code whose (monic) generator polynomial g^\perp is equal to g . According to theorem 2, \mathcal{C}^\perp is a module θ -code if and only if there exist $c \in \mathbf{F}_q - \{0\}$ and $\hbar \in \mathbf{F}_q[X, \theta]$ such that $g\hbar = X^{2k} - c$ and its (monic) generator polynomial is $g^\perp = \sum_{i=0}^k \theta^i(\hbar_{k-i})/\theta^k(\hbar_0) X^i$. So the code \mathcal{C} is self-dual if, and only if, there exist \hbar and c such that $g\hbar = X^{2k} - c$ where $\forall i \in \{0, \dots, k\}$, $\theta^i(\hbar_{k-i})/\theta^k(\hbar_0) = g_i$ i.e. $\hbar_i = \theta^i(c/g_0) \theta^{i-k}(g_{k-i})$. This is equivalent to :

$$\exists c \in \mathbf{F}_q - \{0\}, \left(\sum_{i=0}^k g_i X^i \right) \left(\sum_{i=0}^k \theta^i \left(\frac{c}{g_0} \right) \theta^{i-k}(g_{k-i}) X^i \right) = X^{2k} - c$$

$$\Leftrightarrow \forall l \in \{1, \dots, 2k-1\}, \sum_{i=\max(0, l-k)}^{\min(k, l)} g_i \theta^i \left(\theta^{l-i} \left(\frac{c}{g_0} \right) \theta^{l-i-k}(g_{k-l+i}) \right) = 0$$

$$\Leftrightarrow \forall l \in \{1, \dots, 2k-1\}, \sum_{i=\max(0, l-k)}^{\min(k, l)} g_i \theta^l \left(\frac{c}{g_0} \right) \theta^{l-k}(g_{k-l+i}) = 0$$

$$\Leftrightarrow \forall l \in \{1, \dots, 2k-1\}, \sum_{i=\max(0, l-k)}^{\min(k, l)} \theta^k(g_i) \theta^l(g_{i-(l-k)}) = 0$$

To conclude, it suffices to notice a symmetry in this system of equations, which enables to consider only the k first equations. ■

EXAMPLE. Over \mathbf{F}_4 the (Euclidean) self-dual module θ -codes of length 4 are generated by the polynomials $X^2 + g_1 X + g_0$ where g_0 and g_1 satisfy the equations

$$\begin{cases} \theta(g_0)g_1 + \theta(g_1) = 0 \\ g_0^2 + g_1^2 + 1 = 0 \end{cases}$$

i.e. $g_1(g_0^2 + g_1) = 0$ and $g_0^2 + g_1^2 = 1$. We get three polynomials $X^2 + 1$, $X^2 + \alpha^2 X + \alpha$ and $X^2 + \alpha X + \alpha^2$. It turns out that the codes they generate are ideal θ -cyclic codes. ■

In [5] we computed self-dual ideal θ -codes over \mathbf{F}_4 with length ≤ 40 using Groebner bases. Thanks to a further simplification of the system (1) and, following a suggestion of F. Chyzak and B. Salvy, an exhaustive search instead of Gröbner bases, we could perform the computation of Euclidean self-dual codes over \mathbf{F}_4 of length ≤ 66 . We found two non equivalent (with respect to the monomial action) Euclidean self-dual $[56, 28, 15]_4$ codes which improve the best known distance (14, [7]) for self-dual codes of this length over \mathbf{F}_4 ; five non equivalent (with respect to the monomial action) Euclidean self-dual $[60, 30, 16]_4$ codes (improving the previous distance 15); one (up to equivalence with respect to the monomial action) Euclidean self-dual $[62, 31, 17]_4$ code (improving the distance 16) and one (up to equivalence with respect to the monomial action) $[66, 33, 17]_4$ Euclidean self-dual code (improving the distance 16). We give a generator polynomial for each of these cases in the table 2.

It turns out that all these Euclidean self-dual module θ -codes we have computed are ideal θ -cyclic codes and that we found no self-dual module θ -code which is not θ -constacyclic. We conjecture that an Euclidean self-dual module

Table 2. Euclidean self-dual module codes over \mathbf{F}_4 of lengths > 40 improving minimum distances

Length	A generator polynomial	Minimum Distance	Number of codes
56	$X^{28} + X^{26} + \alpha X^{24} + \alpha^2 X^{22} + \alpha X^{21} + X^{20} + X^{19} + \alpha^2 X^{18} + \alpha X^{17} + \alpha^2 X^{16} + \alpha X^{15} + X^{13} + \alpha^2 X^{12} + X^{11} + \alpha^2 X^{10} + \alpha X^9 + \alpha X^8 + X^7 + \alpha^2 X^6 + X^4 + \alpha X^2 + \alpha$	15	2
60	$X^{30} + X^{29} + \alpha X^{28} + \alpha X^{27} + X^{26} + X^{25} + X^{23} + X^{22} + X^{21} + \alpha^2 X^{20} + X^{16} + X^{15} + X^{14} + \alpha X^{10} + X^9 + X^8 + X^7 + X^5 + X^4 + \alpha^2 X^3 + \alpha^2 X^2 + X + 1$	16	5
62	$X^{31} + X^{29} + \alpha X^{27} + \alpha X^{26} + \alpha X^{25} + \alpha^2 X^{24} + X^{23} + \alpha^2 X^{21} + \alpha X^{20} + \alpha X^{18} + \alpha X^{17} + \alpha X^{16} + \alpha X^{15} + \alpha X^{14} + \alpha X^{13} + \alpha X^{11} + \alpha^2 X^{10} + X^8 + \alpha^2 X^7 + \alpha X^6 + \alpha X^5 + \alpha X^4 + X^2 + 1$	17	1
66	$X^{33} + X^{31} + \alpha^2 X^{29} + \alpha^2 X^{27} + X^{25} + X^{24} + X^{23} + \alpha X^{20} + X^{19} + X^{18} + X^{17} + X^{16} + X^{15} + X^{14} + \alpha X^{13} + X^{10} + X^9 + X^8 + \alpha^2 X^6 + \alpha^2 X^4 + X^2 + 1$	17	1

θ -code is a module θ -constacyclic code and prove this conjecture below when the order of θ divides the length of the code.

Proposition 3. *Let θ be an automorphism of order m of \mathbf{F}_q . Let \mathcal{C} be an Euclidean self-dual module θ -code over \mathbf{F}_q generated by a polynomial g of degree k with constant term $\neq 0$. If the length $2k$ of \mathcal{C} is a multiple of m then \mathcal{C} is an ideal θ -constacyclic code $(g)/(X^{2k} - c)$ with $c \in \{-1, 1\}$.*

Over \mathbf{F}_4 all Euclidean self-dual module θ -codes are ideal θ -constacyclic codes.

PROOF. Let the polynomial g of degree k be the generator of the Euclidean self-dual module θ -code \mathcal{C} . According to theorem 2, there exist a constant $c \neq 0$ and a polynomial $\hbar \in R$ such that $g\hbar = X^{2k} - c$. Since X^{2k} belongs to the center of R we obtain from $(g\hbar)g = X^{2k}g - cg$ that $g(X^{2k} - \hbar g) = cg$. We deduce that g is a left divisor of cg , showing that $cg = g\tilde{c}$ for some $\tilde{c} \in \mathbf{F}_q$. Since the constant term of g is $\neq 0$, comparing the constant terms on both sides shows that $c = \tilde{c}$ and we obtain $cg = gc$. Therefore $\forall i \in \{0, \dots, k\}$ we have $cg_i = g_i\theta^i(c)$ and as g has degree k we get $\theta^k(c) = c$.

From theorem 2 we obtain that the polynomial g^\perp is a left divisor of $X^{2k} - \theta^{k-2k}(1/c) = X^{2k} - \theta^{-k}(1/c)$. Since \mathcal{C} is Euclidean self-dual, the polynomial g^\perp is equal to $\theta^k(\hbar_0)g$. As $X^{2k} - \theta^{-k}(1/c)$ commutes with $\theta^k(\hbar_0)$ we get that g is a left divisor of $X^{2k} - \theta^{-k}(1/c)$. Since g is also a left divisor of $X^{2k} - c$, the polynomial g is a left divisor of the difference of the two polynomials $\theta^{-k}(\frac{1}{c}) - c$ which must therefore be zero. As $\theta^k(c) = c$, we get $\theta^{-k}(\frac{1}{c}) - \theta^k(c) = 0$ and as the order of θ divides $2k$, we deduce from this equality that $c = \frac{1}{c}$ and in the finite field \mathbf{F}_q we obtain $c = 1$ or -1 .

Therefore g is a left and right divisor of the central polynomial $X^{2k} - c$ with $\theta(c) = c \in \{-1, 1\}$, which implies that \mathcal{C} is an ideal θ -constacyclic code.

Lastly, over \mathbf{F}_4 , $m = 2$ divides the length $2k$ of the code, which implies the result. ■

EXAMPLE. Over $\mathbf{F}_9 = \mathbf{F}_3(\alpha)$ with $\alpha^2 - \alpha - 1 = 0$ and $\theta : a \mapsto a^3$ the (Euclidean) self-dual module θ -codes of length 12 are generated by the polynomials $X^6 + g_5X^5 + \dots + g_1X + g_0$ where g_0, \dots, g_5 satisfy the equations

$$\begin{cases} g_0^3 g_5 + g_1^3 = 0 \\ g_0 g_4 + g_1 g_5 + g_2 = 0 \\ g_0^3 g_3 + g_1^3 g_4 + g_2^3 g_5 + g_3^3 = 0 \\ g_0 g_2 + g_1 g_3 + g_2 g_4 + g_3 g_5 + g_4 = 0 \\ g_0^3 g_1 + g_1^3 g_2 + g_2^3 g_3 + g_3^3 g_4 + g_4^3 g_5 + g_5^3 = 0 \\ g_0^2 + g_1^2 + g_2^2 + g_3^2 + g_4^2 + g_5^2 + 1 = 0 \end{cases}$$

Solving the corresponding polynomial system, we find 40 solutions in \mathbf{F}_9^6 ; one of these gives the polynomial $X^6 + 2X^5 + \alpha^3 X^4 + \alpha^2 X^3 + \alpha X^2 + X + 1$ which divides on the right the polynomial $X^{12} + 1$. It generates an Euclidean self-dual $[12, 6, 6]_9$ ideal θ -constacyclic code. ■

We conjecture that an Euclidean self-dual module θ -code of length $2k$ is always a module (not ideal) θ -constacyclic code $(g)/(X^{2k} - c)$ with $c^2 = 1$ when the order of θ does not divide $2k$. Here is an example over \mathbf{F}_{16} .

EXAMPLE. Over $\mathbf{F}_{16}[X, \theta]$ with $\mathbf{F}_{16} = \mathbf{F}_2(\alpha)$ where $\alpha^4 + \alpha + 1 = 0$ and with the Frobenius automorphism $\theta : a \mapsto a^2$ of order 4, the Euclidean self-dual module θ -codes of length 10 are generated by the polynomials

$$\begin{aligned} & X^5 + \alpha^{10} X^4 + \alpha^{10} X^3 + \alpha^{10} X^2 + \alpha^{10} X + 1, \\ & X^5 + \alpha^5 X^4 + \alpha^5 X^3 + \alpha^5 X^2 + \alpha^5 X + 1, \\ & X^5 + X^4 + \alpha^{10} X^3 + \alpha^{10} X^2 + X + 1, \\ & X^5 + X^4 + \alpha^5 X^3 + \alpha^5 X^2 + X + 1, \\ & X^5 + 1 \end{aligned}$$

These polynomials all divide $X^{10} - 1$ on the right, so the module θ -codes they generate are module θ -constacyclic codes. They are not ideal θ -codes because the order of θ does not divide the length of the code. ■

In the next section, we consider Hermitian duals of module θ -codes. We will see that they all are θ -constacyclic but that there exist Hermitian self-dual module θ -codes over \mathbf{F}_4 which are *not* ideal θ -constacyclic.

4.2 Dual for the Hermitian Scalar Product

Let q be an even power of a prime number and let θ be the automorphism of order 2 over \mathbf{F}_q : $a \mapsto a^{\sqrt{q}}$. The *Hermitian scalar product* is defined over \mathbf{F}_q^n by

$$\forall x, y \in \mathbf{F}_q^n, \langle x, y \rangle_H = \sum_{i=1}^n x_i \cdot \theta(y_i), \quad \text{i.e. } \langle x, y \rangle_H = \langle x, \theta(y) \rangle.$$

The Hermitian dual of a code of \mathbf{F}_q^n is the set of words which are orthogonal to the code's words relatively to the Hermitian scalar product.

Proposition 4 : Hermitian dual of a module θ -code. Let $k \leq n$ be integers, $g \in \mathbf{F}_q[X, \theta]$ of degree $n - k$ with constant term $\neq 0$ and \mathcal{C} be the module θ -code of length n generated by g . The Hermitian dual \mathcal{C}^H of \mathcal{C} is a module θ -code generated by a polynomial of degree k with constant term $\neq 0$ if and only if there exist $\hbar \in \mathbf{F}_q[X, \theta]$ and c in $\mathbf{F}_q - \{0\}$ such that $g\hbar = X^n - c$ (here g is on the left).

In this case the generator polynomial of \mathcal{C}^H is given by :

$$g^H = \sum_{i=0}^k \theta^{i+1}(\hbar_{k-i}) X^i = \phi(g^\perp)$$

where $\phi(\sum_{i=0}^n a_i X^i) = \sum_{i=0}^n \theta(a_i) X^i$. Lastly, g^H divides the polynomial $X^n - \theta^{k-n+1}(1/c)$ on the left.

PROOF. The proof follows from the theorem 2 using that $\forall i_0 \in \{0, \dots, k\}$, $\forall i_1 \in \{0, \dots, n - k\}$,

$$\langle X^{i_0} g, X^{i_1} g^H \rangle_H = \langle X^{i_0} g, X^{i_1} \phi(g^H) \rangle = \langle X^{i_0} g, X^{i_1} g^\perp \rangle$$

■

Corollary 5 : Hermitian self-dual module θ -codes. Let $k \in \mathbb{N}$ and $g \in \mathbf{F}_q[X, \theta]$ be a monic polynomial of degree k and constant term $\neq 0$. Let \mathcal{C} be the module θ -code of length $2k$ generated by g . The code \mathcal{C} is Hermitian self-dual if, and only if,

$$\forall l \in \{1, \dots, k\}, \sum_{i=0}^l \theta^{k-l-1}(g_i) g_{i+k-l} = 0 \quad (2)$$

PROOF. We use the same techniques as in the lemma for Euclidean self-dual codes. ■

EXAMPLE. Over \mathbf{F}_4 the (Hermitian) self-dual module θ -codes of length 4 are generated by the polynomials $X^2 + g_1 X + g_0$ where g_0 and g_1 satisfy the equations

$$\begin{cases} g_0 g_1 + g_1 = 0 \\ g_0 \theta(g_0) + g_1 \theta(g_1) + 1 = 0 \end{cases}$$

i.e $g_1(g_0 + 1) = 0$ and $g_0^3 + g_1^3 = 1$. We get three polynomials $X^2 + 1$, $X^2 + \alpha$ and $X^2 + \alpha^2$. The bound of the polynomials $X^2 + \alpha$ and $X^2 + \alpha^2$ is $X^4 + X^2 + 1$, so these two polynomials generate Hermitian self-dual central θ -codes which are not ideal θ -cyclic. One can notice that the remainder in the left division of X^4 by $X^2 + \alpha$ (resp. $X^2 + \alpha^2$) is equal to α^2 (resp. α). ■

Like for the Euclidean scalar product, using the corollary 5, we constructed all Hermitian self-dual module θ -codes of length ≤ 66 . We found 14 (up to equivalence with respect to the monomial action) new [50, 25, 14]₄ Hermitian self-dual codes, which improve the best known distance (12) by 2 and 20 (up to equivalence with respect to the monomial action) new [58, 29, 16]₄ codes which also improve the best distance (14) by 2 (see table 3).

Table 3. Hermitian self-dual module codes over \mathbf{F}_4 of lengths > 40 improving minimum distances

Length	A generator polynomial	Minimum Distance	Number of codes
50	$X^{25} + X^{21} + \alpha X^{20} + X^{19} + \alpha^2 X^{17} + \alpha X^{16} + \alpha X^{15} + \alpha^2 X^{14} + X^{11} + \alpha X^{10} + \alpha X^9 + X^8 + \alpha^2 X^6 + \alpha X^5 + \alpha^2 X^4 + \alpha^2$	14	14
58	$X^{29} + X^{25} + \alpha X^{24} + X^{23} + \alpha^2 X^{21} + \alpha^2 X^{19} + \alpha X^{18} + X^{17} + \alpha^2 X^{16} + X^{13} + \alpha^2 X^{12} + \alpha X^{11} + X^{10} + X^8 + \alpha^2 X^6 + \alpha X^5 + \alpha^2 X^4 + \alpha^2$	16	20

It turns out that all these new codes are ideal θ -cyclic, however there exist Hermitian self-dual module θ -codes over \mathbf{F}_4 which are not ideal θ -constacyclic. The following lemma enables to give a more accurate description of Hermitian self-dual module θ -codes.

Lemma 1. Let \mathcal{C} be a Hermitian self-dual module θ -code with generator polynomial g of degree k and constant term $\neq 0$. Then

- \mathcal{C} is a module θ -constacyclic code : there exist a non zero constant c and $h \in \mathbf{F}_q[X, \theta]$ such that $gh = X^{2k} - c = hg$;
- if k is odd then \mathcal{C} is an ideal θ -constacyclic code with $c^2 = 1$;
- if k is even then $\theta(c)c = 1$ and all odd terms of g cancel.

PROOF. Let us assume that \mathcal{C} is a Hermitian self-dual module θ -code of length $2k$ and let g be its generator polynomial. Then there exist c in $\mathbf{F}_q - \{0\}$ and \hbar in $\mathbf{F}_q[X, \theta]$ such that $g\hbar = X^{2k} - c$. So $(g\hbar)g = X^{2k}g - cg = gX^{2k} - cg$ because X^{2k} is in the center of $\mathbf{F}_q[X, \theta]$. We deduce from this that g divides cg on the left and as $g_0 \neq 0$, we get $gc = cg$. So $g(\hbar g) = gX^{2k} - gc$ and $\hbar g = X^{2k} - c$.

The polynomial g^H divides on the left $X^{2k} - \theta^{k+1}(1/c)$ and as the leading term $\theta^{k+1}(\hbar_0)$ of g^H commutes with X^{2k} , the polynomial $1/\theta^{k+1}(\hbar_0)g^H$ divides also $X^{2k} - \theta^{k+1}(1/c)$. Therefore g and $1/\theta^{k+1}(\hbar_0)g^H$ are both monic and generate the same code, they must be equal. So $X^{2k} - c = X^{2k} - \theta^{k+1}(1/c)$ and if k is odd $c = 1/c$; if k is even, $c = \theta(1/c)$.

Lastly, as $g\hbar = \hbar g = X^{2k} - c$, we have $(X^{2k} - c)g = g(X^{2k} - c)$ so $\forall i \in \{0, \dots, k\}, (c - \theta^i(c))g_i = 0$. For odd integers i , we get $(c - \theta(c))g_i = 0$. So if $\theta(c) \neq c$ then all odd terms of g cancel. ■

The last point of this lemma implies that any Hermitian self-dual module θ -code which is not an ideal θ -constacyclic code (i.e. $\theta(c) \neq c$) has a generator polynomial which is quite "sparse". The minimum distances of these codes are worse than the minimum distances of the self-dual θ -constacyclic codes previously obtained in [5]. This may be explained by the "sparsity" of this generator polynomial.

EXAMPLE. There are six Hermitian self-dual module θ -codes over \mathbf{F}_4 of length 20 which are central but not θ -cyclic. They give two non equivalent Hermitian self-dual module θ -codes over \mathbf{F}_4 . The polynomial $X^{10} + \alpha^2$ divides on the left and on the right $X^{20} - \alpha$ and its bound is $X^{20} + X^{10} + 1$. It generates a $[20, 10, 2]_4$ Hermitian self-dual module θ -code which is θ -central and not θ -cyclic. The polynomial $X^{10} + \alpha X^8 + X^6 + \alpha X^4 + \alpha X^2 + \alpha^2$ also divides on the left and on the right $X^{20} - \alpha$ and its bound is $X^{20} + X^{18} + X^{16} + X^8 + X^6 + X^2 + 1$. It generates an $[20, 10, 4]_4$ Hermitian self-dual module θ -code which is θ -central and not θ -cyclic. The best distance for ideal θ -cyclic codes of the same length is 6 ([5]). One can notice the sparsity of these generator polynomials which may explain the bad distances of the codes they generate. ■

Acknowledgments. We would like to thank Bas Edixhoven and Antoine Chambert-Loir for suggesting the use of modules for skew codes and Frédéric Chyzak and Bruno Salvy for suggesting an exhaustive search instead of Gröbner Bases in order to compute self-dual codes.

References

1. Berrick, A., Keating, M.: An introduction to rings and modules. Cambridge Studies in Advanced Mathematics, vol. 65. Cambridge University Press, Cambridge (2000)
2. Bosma, W., Cannon, J., Playoust, C.: The magma algebra system i: The user language. *Journal of Symbolic Computation* 24, 235–265 (1997)
3. Boucher, D., Geiselmann, W., Ulmer, F.: Skew Cyclic Codes, Applied Algebra in Engineering, Communication and Computing 18, 379–389 (2007)
4. Boucher, D., Solé, P., Ulmer, F.: Skew Constacyclic Codes over Galois Rings. *Advances in Mathematics of Communications* 2, 273–292 (2008)
5. Boucher, D., Ulmer, F.: Coding with skew polynomial rings. *Journal of Symbolic Computation* 44, 1644–1656 (2009)
6. Chaussade, L., Loidreau, P., Ulmer, F.: Skew codes of prescribed distance or rank. *Designs, Codes and Cryptography* 50(3), 267–284 (2009)
7. Gaborit, P., Otmani, A.: Tables of Euclidean and Hermitian self-dual codes over GF(4) (2002), http://www.unilim.fr/pages_perso/philippe.gaborit/SD/
8. Jacobson, N.: The theory of rings. Publication of the AMS (1943)
9. McDonald, B.R.: Finite Rings with Identity. Marcel Dekker Inc., New York (1974)
10. Ore, O.: Theory of non-commutative polynomials. *Ann. of Math.* 34 (1933)

On Higher Weights and Code Existence

Hans Georg Schaathun

University of Surrey
Department of Computing
GU2 7XH Guildford, Surrey
H.Schaathun@surrey.ac.uk

Abstract. Several open questions in coding theory relate to non-existence or construction of certain optimal codes. Many previous problems of this kind have been solved by studying possible weight enumerators. A couple of authors in this decade have proposed using higher weights (generalised Hamming weights) to a similar effect. In this paper we suggest one approach based on the weight hierarchy, and it allows us to conduct an extremely rapid computer search to prove that there are exactly two inequivalent [36, 8, 16] codes. The technique can also be used to gain new information about the weight hierarchy of the putative [72, 36, 16] code, but not yet enough to say if it exists or not.

1 Introduction

Higher Weights, that is, parameters describing the support weight of subcodes of dimension higher than one, was a very hot topic during the 1990-s. Helleseth, Kløve, and Mykkeltveit [6] had already in 1977 introduced the support weight distributions. From the support weight distribution for a single binary code, they could determine the weight distribution for an infinite class of non-binary codes.

Victor Wei [13] introduced the weight hierarchy, that is, the sequence of minimal support weights of any r -dimensional subcode, which he used to analyse information-theoretic security on the Wire-Tap Channel of Type II [10].

Later it has been suggested to use higher weights to limit searches for putative optimal codes. Dougherty, Gulliver, and Oura [4] showed that the second support weight distribution of the putative [72, 36, 16] code could be calculated by using the MacWilliams-Kløve-Simonis identities [7, 12]. Another work [11] found a way to calculate some of the high-order support weight distributions for this code.

More recently Luo, Mitrpant, Han Vinck, and Chen [8] introduced the concept of relative generalised Hamming weights. Their application was analysis of a two-party Wire-Tap Channel of Type II, and the work has drawn little attention in the subsequent literature. Could it help us solve some of the long-standing problems in coding theory?

In the present paper, we discuss the idea of using the weight hierarchy to constrain code searches. We show that the weight hierarchy of any [36, 8, 16] code can be uniquely determined, and that this gives us enough information to run an exhaustive search in less than 2 minutes.

2 Preliminaries

For any vector $\mathbf{c} = (c_1, \dots, c_n) \in F^n$ (where F is a field), the support of \mathbf{c} is defined as

$$\chi(\mathbf{c}) = \{i : c_i \neq 0\}.$$

The (support) weight of \mathbf{c} is $w(\mathbf{c}) = \#\chi(\mathbf{c})$. For a set $D \subset F^n$, the support is defined as

$$\chi(D) = \cup_{\mathbf{c} \in D} \chi(\mathbf{c}),$$

and the (support) weight, as before, is $w(D) = \#\chi(D)$.

Let C be an $[n, k]$ code over F . The weight hierarchy (d_1, d_2, \dots, d_k) is defined by

$$d_i = \min_{D \leq C, \dim D=i} w(D),$$

i.e. d_i is the minimal weight of an i -dimensional subcode. Clearly, $d_1 = d$ is the regular minimal distance, and $d_0 = 0$ for completeness.

The support weight distribution is the set of parameters $A_i^{(r)}$ for $r = 0, 1, \dots, k$ and $i = 0, 1, \dots, n$, where $A_i^{(r)}$ is the number of r -dimensional subcodes $D \leq C$ of weight $w(D) = i$. Like the traditional weigh enumerator, we can form support weight enumerators

$$W_r(Z) = \sum_{i=0}^n A_i^{(r)} Z^i.$$

Obviously the weight enumerator $W(Z)$ is $W(Z) = W_1(Z) + 1$.

Two binary codes are said to be equivalent if one can be obtained from the other by a combination of permutations of the columns (coordinate positions).

3 The [36, 8, 16] Code

The first binary [36, 8, 16] code was discovered by Helleseth and Ytrehus [2], using a computer search detailed in [15]. It had been a long-standing open question whether the optimal minimal distance for a [36, 8] code would be 15 or 16. An exhaustive search was not feasible, and until now, it has not been known whether the code they found is unique. Our technique gives us the following proposition in few minutes on any personal computer.

Proposition 1. *There are exactly two distinct [36, 8, 16] codes up to equivalence.*

3.1 The Weight Hierarchy

Lemma 1 (Ytrehus). *Every binary [36, 8, 16] code has weight enumerator*

$$A(Z) = 1 + 153Z^{16} + 72Z^{20} + 30Z^{24}.$$

The weight enumerator was found in [15], showing clearly that the code is doubly-even and consequently self-orthogonal.

Using the MacWilliams-Kløve-Simonis identities [7,12], and using the generalised Griesmer bound to fix some zero coefficients for small weights, we can get most of the coefficients in the support weight distribution as well, as seen in Tables 1 and 2.

Table 1. The second through the fourth support weight distribution of a [36, 8, 16] binary code

$$\begin{aligned}
A_{24}^{(2)} &= 2420 + A_{36}^{(2)} \\
A_{26}^{(2)} &= 3228 - 6A_{36}^{(2)} \\
A_{28}^{(2)} &= 2640 + 15A_{36}^{(2)} \\
A_{30}^{(2)} &= -20A_{36}^{(2)} + 1832 \\
A_{32}^{(2)} &= 615 + 15A_{36}^{(2)} \\
A_{34}^{(2)} &= 60 - 6A_{36}^{(2)} \\
A_{36}^{(2)} &= A_{36}^{(2)} \\
\\
A_{28}^{(3)} &= A_{28}^{(3)} \\
A_{29}^{(3)} &= 48A_{36}^{(2)} + 36240 - 8A_{28}^{(3)} \\
A_{30}^{(3)} &= 28A_{28}^{(3)} - 288A_{36}^{(2)} - 77040 \\
A_{31}^{(3)} &= 720A_{36}^{(2)} + 203184 - 56A_{28}^{(3)} \\
A_{32}^{(3)} &= 70A_{28}^{(3)} - 960A_{36}^{(2)} - 213645 \\
A_{33}^{(3)} &= 720A_{36}^{(2)} + 201840 - 56A_{28}^{(3)} \\
A_{34}^{(3)} &= 28A_{28}^{(3)} - 288A_{36}^{(2)} - 82080 \\
A_{35}^{(3)} &= -8A_{28}^{(3)} + 48A_{36}^{(2)} + 31056 \\
A_{36}^{(3)} &= A_{28}^{(3)} - 2400 \\
\\
A_{30}^{(4)} &= 320 + 4A_{36}^{(2)} \\
A_{31}^{(4)} &= 7008 - 24A_{36}^{(2)} \\
A_{32}^{(4)} &= 25815 + 60A_{36}^{(2)} \\
A_{33}^{(4)} &= 41600 - 80A_{36}^{(2)} \\
A_{34}^{(4)} &= 55560 + 60A_{36}^{(2)} \\
A_{35}^{(4)} &= 49056 - 24A_{36}^{(2)} \\
A_{36}^{(4)} &= 21428 + 4A_{36}^{(2)}
\end{aligned}$$

Table 2. The fifth through the eighth support weight distribution for a [36, 8, 16] binary code are uniquely determined by the MacWilliams-Kløve identities

	5	6	7	8
32	225	—	—	—
33	6240	—	—	—
34	19620	630	—	—
35	37152	3312	36	—
36	33918	6853	219	1

Lemma 2. *Every binary [36, 8, 16] code has weight hierarchy*

$$(16, 24, 28, 30, 32, 34, 35, 36).$$

Proof. Note in particular that $A_{24}^{(2)}$ and $A_{30}^{(4)}$ are positive, so $d_2 = 24$ and $d_4 = 30$. It follows from the generalised Griesmer bound that $d_3 = 28$.

The values for d_5 , d_6 , d_7 , and d_8 may be read directly from Table 2.

The chain condition, introduced by Wei and Yang [14], states that there exists a sequence of subcodes

$$\{0\} < D_1 < D_2 < \dots < D_{k-1} < C$$

such that $w(D_r) = d_r$ for each r , where $<$ denotes a proper subgroup (subspace). Note that it follows that $\dim D_r = r$.

Lemma 3. *Every binary [36, 8, 16] code satisfies the chain condition.*

Proof. Consider a subcode D_{30}^4 of dimension 4 and weight 30. Any [30, 4, 16] code is equivalent to the two-fold replication of the [15, 4, 8] simplex code. All the codewords in such a subcode has weight zero or 16, and in particular, every three-dimensional subcode $D^3 < D_{30}^4$ contains seven words of weight 16, and thus $w(D^3) = 28$. Hence there are $15A_{30}^4 \geq 4800$ pairs $(D_{28}^3 < D_{30}^4)$.

Solving the system of inequalities $A_i^{(r)} \geq 0$ for $r = 2, 3$ and all i , we get that $A_{28}^3 \leq 3732$, and consequently there must be two four-dimensional subcodes E_1 and E_2 of weight 30 that intersect in a three-dimensional subcode D_{28}^3 of weight 28. The span $D_{32}^5 = \langle E_1, E_2 \rangle$ must be a five-dimensional subcode of weight 32. Hence we have a chain of subcodes

$$\{0\} < D_{16}^1 < D_{24}^2 < D_{28}^3 < D_{30}^4 < D_{32}^5 < D_{36}^8 = C.$$

By puncturing C on an arbitrary coordinate not in $\chi(D_{32}^5)$ we obtain a seven-dimensional subcode D_{35}^7 of weight 35, and by puncturing on a second coordinate, also a six-dimensional subcode D_{34}^6 of weight 34, completing the chain

$$\{0\} < D_{16}^1 < D_{24}^2 < D_{28}^3 < D_{30}^4 < D_{32}^5 < D_{34}^6 < D_{35}^7 < D_{36}^8 = C. \quad (1)$$

Ergo, any [36, 8, 16] binary code satisfies the chain condition.

Table 3. Two inequivalent [36, 8, 16] codes, where G_Y is equivalent to Ytrehus' code, and G_{new} is new

$$G_Y = \begin{bmatrix} 111111111111111100000000000000000000000 \\ 1111111100000000111111100000000000000 \\ 111100001111000011110000111100000000 \\ 110011001100110011001100110011000000 \\ 1011101110111011101110111011101100000 \\ 0000000000111010001110111001111100 \\ 00010001000001101110111100110010110 \\ 00000000110100101101011111000011 \end{bmatrix}$$

$$G_{\text{new}} = \begin{bmatrix} 11111111111111110000000000000000000000 \\ 1111111100000000111111100000000000000 \\ 111100001111000011110000111100000000 \\ 110011001100110011001100110011000000 \\ 1011101110111011101110111011101100000 \\ 0000000000111010001110111001111100 \\ 0001000100000101111010110101010110 \\ 00000000100011101110100001100111111 \end{bmatrix}$$

3.2 The Code Search

The [30, 4, 16] subcode is clearly unique, where all non-zero codewords have weight 16. It may be possible to construct the possible [32, 5, 16] codes analytically as well, but there is little point as a computer search can be made in less than a minute.

Suppose we have constructed a $[N, K]$ subcode D , and need a $[N + t, K + 1]$ subcode. We construct candidates for Row $K + 1$, as the set

$$S := \{\mathbf{x} \in D^\perp : \forall \mathbf{c} \in c, w := w(\mathbf{x} + \mathbf{c}) + t, d \leq w \leq m \wedge w \mod 4 \cong 0\},$$

where d and m are the minimal and maximal weights (16 and 24 for the [36, 8, 16] code). Obviously, the conditions can be modified to allow for singly-even codes, or even codes which are not self-orthogonal.

All possible codes are constructed by appending t zero columns, and all possible rows $\mathbf{x}||(1 \dots 1)$ for $\mathbf{x} \in S$ to the generator matrix of D .

In order to rule out equivalent codes, we use the nauty library of Brendan McKay [9]. Nauty works on coloured graphs, so we use a standard technique to represent codes as graphs. We need a set S of codewords which is invariant under all automorphisms, and which spans the code. Usually, the set of minimal weight codewords will do, but if this does not span the code, we add codewords of the next higher weight, until we span the code.

Each codeword in S corresponds to a black vertex in the graph. There is a white vertex for each coordinate position, and there is an edge between a black vertex B and a white vertex W , if the codeword corresponding to B is one in the position corresponding to W .

The graphs are represented by incidence matrices, and every graph corresponding to a linear code C has an incidence matrix of the form

$$I = \begin{bmatrix} 0 & M^T \\ M & 0 \end{bmatrix},$$

where the rows of M are the necessary low weight codewords of C .

From an incidence matrix I , nauty can produce a canonical incidence matrix which is common for all isomorphic graphs (equivalent codes). We can form a canonical generator matrix for the corresponding code by Gaussian elimination on the matrix M . Since this algorithm is deterministic, the same canonical graph will always give the same canonical generator matrix.

When we want to reject equivalent codes, we keep a hash table using the canonical generator matrix as a key. For each code we generate, we try to insert it in the hash table. If it is already there, we have already searched this code and proceed immediately to the next one. If it is successfully inserted, we continue by searching this code.

Using this search algorithm, we find two inequivalent [32, 5, 16] codes, in half a minute on a mid-range laptop. Growing the code from [32, 5] to [36, 8] it is advantageous to use the same candidate set S for all the three rows required. Because we know that the $d_7 = 35$, we know that the [36, 8, 16] code is equivalent to one with generator matrix on the form

$$G = \begin{bmatrix} G' & 0 \\ \mathbf{s}_1 & 1100 \\ \mathbf{s}_1 & 1010 \\ \mathbf{s}_1 & 1001 \end{bmatrix}, \quad (2)$$

where G' is a generator matrix of a [32, 5, 16] code. This search takes 70-80 seconds, yielding two inequivalent codes as shown in Table 3.

4 Partial Results on the [72, 36, 16] Code

Inspired by our success with the relatively small [36, 8] code, we give some preliminary results for the [72, 36, 16] code. In this section, we let C denote an arbitrary [72, 36, 16] Type II code. We know from [3] that $d_1 = 16$ and $d_2 = 24$.

4.1 Further Preliminaries

We need the well-known Johnson bounds for some of the proofs. Let $A(n, d, w)$ denote the maximum size of a (non-linear) code with constant weight w and minimum distance d .

Lemma 4 (Johnson bounds). *We have*

$$\begin{aligned} A(n, 2w, w) &= \lfloor \frac{n}{w} \rfloor, \\ A(n, d, w) &\leq \lfloor \frac{n}{w} A(n - 1, d, w - 1) \rfloor, \\ A(n, d, w) &\leq \lfloor \frac{n}{n - w} A(n - 1, d, w) \rfloor. \end{aligned}$$

Forney [5] discussed a series of duality results for higher weights, some of which could be traced back to Wei [13]. We summarise a few key points which we will use. Let $I = \{1, \dots, n\}$ be the co-ordinate index set. For any $J \subset I$, let C_J denote code C shortened on $I \setminus J$, i.e.

$$C_J = \{\mathbf{c} \in C : \forall i \notin J, c_i = 0\}.$$

It is known that if $\dim C_J = r$, then $\dim(C^\perp)_{I \setminus J} = r + n - k - \#J$. Clearly, for each r , there is $J \subset I$ such that $w(C_J) = d_r$. Then, $w((C^\perp)_{I \setminus J}) = d_{r+n-k-d_r}^\perp$.

Let $P_J(C)$ be the code punctured on $I \setminus J$, i.e. the code

$$P_J(C) = \{(c_1, \dots, c_n) : \exists (c'_1, \dots, c'_n) \in C, \forall i \in I \setminus J, c_i = c'_i; \forall i \in J, c_i = 0\}.$$

Clearly $\dim P_J(C) + \dim C_J = \dim C$.

We define the past subcode $P_i = C_{1, \dots, i}$ and the future subcode $F_i = C_{i+1, \dots, n}$.

4.2 The Third, Fourth, and Fifth Weight

Lemma 5. *Any [72, 36, 16] code has $d_3 = 28$ or $d_3 = 29$.*

Proof. We get $d_3 \geq 28$ from the Griesmer bound. Consider a shortened code C_J of weight $w(C_J) = 24$ and dimension 2. Then $P_{I \setminus J}(C)$ would be a [48, 34] code, which has minimum distance 6 or less by Brouwer's tables [1]. Hence $d_3 \leq 30$.

Suppose for a contradiction that $d_3 = 30$.

Assume a coordinate ordering such that P_{16} has dimension one and P_{24} dimension two. Now P_{16} is a doubly-even [56, 21, 16] code containing the all-one word, and thus F_{16}^\perp is a [56, 35, 8] even code. Solving the MacWilliams identities, we find that F_{16}^\perp has 1155 words of weight 8.

Since C has $d_3 = 30$, F_{16}^\perp has $d_2 \geq 14$, and thus two words of weight 8 must have distance at least 12. This results in a (56, 1155, 12) constant weight code with $w = 8$. However, this is impossible because

$$\begin{aligned} A(56, 12, 8) &\leq \lfloor \frac{56}{8} A(55, 12, 7) \rfloor \leq \lfloor \frac{56}{8} \lfloor \frac{55}{7} A(54, 12, 6) \rfloor \rfloor \\ &\leq \lfloor \frac{56}{8} \lfloor \frac{55}{7} \lfloor \frac{54}{6} \rfloor \rfloor \rfloor = 490, \end{aligned}$$

by Lemma 4.

Lemma 6. *If C has $d_3 = 28$, then $30 \leq d_4 \leq 32$.*

Proof. We have $d_4 \geq 30$ by the Griesmer bound, and $d_4 \leq 33$ because $d(44, 33) \leq 5$. Suppose $d_4 = 33$ for a contradiction. Assume a coordinate ordering such that P_{28} has dimension 3. Now F_{28}^\perp is a [44, 33, 5] code containing the all-one word, and F_{28} is doubly-even without the all-one word. The MacWilliams identities for this code pair has no integer solutions, so the codes cannot exist.

Lemma 7. *If C has $d_3 = 29$, then $32 \leq d_4 \leq 33$.*

Proof. The lower bound follows from Griesmer and the upper bound follows from the fact that $d(43, 33) = 4$.

Lemma 8. *We have $d_5 < 37$.*

Proof. We know that $d_4 \leq 33$, so $d_5 \leq 37$ follows from Brouwer's tables. Suppose for a contradiction that $d_5 = 37$. Then $d_6 \geq 39$ by Griesmer, and $d_6 \leq 39$ by Brouwer's tables. Thus we get top-down greedy weights $\tilde{e}_4 = 35$ and $\tilde{e}_3 = 33$. We know that \tilde{e}_2 is even, so it must be 30 or 32, but then there is no possible choice for \tilde{e}_1 and we therefore conclude that $d_5 < 37$.

5 Conclusion

We have presented a novel approach to constraining code searches. This approach proved very effective in the case of [36, 8, 16] where an exhaustive search can be done in less than two CPU-minutes, and show exactly two distinct codes up to equivalence.

Hopefully, this can inspire renewed interest in some of the legendary problems of coding theory, and combining the present techniques with others, one might just see some solutions in the foreseeable future.

References

1. Ore, O.: Theory of non-commutative polynomials. *Ann. of Math.* 34 (1933)
2. Dodunekov, S.M., Helleseth, T., Manev, N., Ytrehus, Ø.: New bounds on binary linear codes of dimension eight. *IEEE Trans. Inform. Theory* 33(6), 917–919 (1987)
3. Dougherty, S., Gulliver, A.: Higher weights of self-dual codes. In: Augot, D. (ed.) *Workshop on Coding and Cryptography*, pp. 177–188 (January 2001)
4. Dougherty, S., Gulliver, A., Oura, M.: Higher weights and graded rings for binary self-dual codes. *Discrete Applied Mathematics* 128, 251–261 (2003); Special issue for WCC 2001
5. David Forney Jr., G.: Dimension/length profiles and trellis complexity of linear block codes. *IEEE Trans. Inform. Theory* 40(6), 1741–1752 (1994)
6. Helleseth, T., Kløve, T., Mykkeltveit, J.: The weight distribution of irreducible cyclic codes with block lengths $n_1((q^l - 1)/n)$. *Discrete Math.* 18, 179–211 (1977)
7. Kløve, T.: Support weight distribution of linear codes. *Discrete Math.* 106/107, 311–316 (1992)
8. Luo, Y., Mitrpant, C., Vinck, A.J.H., Chen, K.: Some new characters on the wire-tap channel of type II. *IEEE Transactions on Information Theory* 51(3), 1222–1229 (2005)
9. McKay, B.D.: The nauty page (2002), <http://cs.anu.edu.au/people/bdm/nauty/>
10. Ozarow, L.H., Wyner, A.D.: Wire-tap channel II. *AT&T Bell Laboratories Technical Journal* 63(10), 2135–2157 (1984)
11. Schaathun, H.G.: Duality and support weight distributions. *IEEE Transactions on Information Theory* 50(5), 862–867 (2004)

12. Simonis, J.: The effective length of subcodes. *Appl. Algebra Engrg. Comm. Comput.* 5(6), 371–377 (1994)
13. Wei, V.K.: Generalized Hamming weights for linear codes. *IEEE Trans. Inform. Theory* 37(5), 1412–1418 (1991)
14. Wei, V.K., Yang, K.: On the generalized Hamming weights of product codes. *IEEE Trans. Inform. Theory* 39(5), 1709–1713 (1993)
15. Ytrehus, Ø.: Code-buster: A software tool for characterizing abstract codes. Technical report, Dept. of Informatics, University of Bergen (March 1987)

Mass Formula for Even Codes over \mathbb{Z}_8

Koichi Betsumiya¹, Rowena Alma L. Betty^{2,*}, and Akihiro Munemasa²

¹ Graduate School of Science and Technology,
Hirosaki University, Hirosaki 036-8561, Japan

² Graduate School of Information Sciences, Tohoku University,
Sendai 980-8579, Japan

Abstract. In this paper, we establish a mass formula for even codes over \mathbb{Z}_8 . In particular, a formula giving the total number of distinct Type II self-dual codes over \mathbb{Z}_8 of length n is established for each positive integer n divisible by 8.

1 Introduction

In this paper, we establish a mass formula for even codes over \mathbb{Z}_8 . An even code is a self-orthogonal code having Euclidean weights divisible by 16, or equivalently, a code such that the lattice constructed by Construction A is even. Self-dual codes which are even are also known as Type II codes, and we will establish a formula for the number of distinct Type II codes of a given length. This means finding a formula for

$$\sum_{\mathcal{C}} \frac{2^n \cdot n!}{|\text{Aut } \mathcal{C}|}$$

where \mathcal{C} runs through the equivalence classes of Type II codes of length n over \mathbb{Z}_8 and $\text{Aut } \mathcal{C}$ is the automorphism group of \mathcal{C} . The formula is a special case of a mass formula for even codes of given type, and the proof requires an analogous formula for codes over \mathbb{Z}_4 , extending our previous work [2].

Like the mass formula given in [2], our mass formula for even codes over \mathbb{Z}_8 is given as the sum of mass formulas over some finer classes of codes. This is because codes over \mathbb{Z}_8 have an invariant called a type, denoted by $\{k_0, k_1, k_2\}$. The type of a code over \mathbb{Z}_8 is determined by the type of its 4-residue and dimension of the torsion. We shall determine the number of even codes over \mathbb{Z}_8 with given 4-residue and torsion, and this number depends only on the type of the 4-residue and dimension of the torsion.

To conclude the paper, we demonstrate that our theoretical count of Type II codes of length 8 over \mathbb{Z}_8 agrees with a recent computer enumeration of these codes given by [6].

* On study leave from the Institute of Mathematics, University of the Philippines-Diliman, Quezon City 1101 Philippines.

2 Preliminaries

For a positive integer m , we denote by \mathbb{Z}_m the ring of integers modulo m . A code \mathcal{C} of length n over \mathbb{Z}_m is a submodule of \mathbb{Z}_m^n . For a matrix $G \in M_{k \times n}(\mathbb{Z})$, we denote by $\mathbb{Z}_m^k G$ the code $\{aG \bmod m \mid a \in \mathbb{Z}^k\}$ of length n over \mathbb{Z}_m . A generator matrix of a code \mathcal{C} of length n over \mathbb{Z}_m is a matrix $G \in M_{k \times n}(\mathbb{Z})$ such that $\mathcal{C} = \mathbb{Z}_m^k G$. Usually, entries of a generator matrix of a code over \mathbb{Z}_m are taken to be in \mathbb{Z}_m . However, since we deal with codes over \mathbb{Z}_p and \mathbb{Z}_{p^2} at the same time, we adopt this non-standard convention to avoid cumbersome notation.

We denote by $x \cdot y$ the standard inner product of vectors x, y in \mathbb{Z}_m^n , and by \mathcal{C}^\perp the dual code of a code \mathcal{C} over \mathbb{Z}_m with respect to this inner product. A code \mathcal{C} is said to be self-orthogonal (respectively self-dual) if $\mathcal{C} \subset \mathcal{C}^\perp$ (respectively $\mathcal{C} = \mathcal{C}^\perp$) holds.

Let \mathcal{C} be a code of length n over \mathbb{Z}_8 . For $i = 0, 1, 2$, we define the torsion codes of \mathcal{C} (see [4]) as follows:

$$\text{tor}_i(\mathcal{C}) = \{v \bmod 2 \mid v \in \mathbb{Z}_8^n, 2^i v \in \mathcal{C}\} .$$

The code $\text{tor}_0(\mathcal{C})$ is also called the residue code of \mathcal{C} , which we will denote by $\text{res}(\mathcal{C})$. Observe

$$\text{tor}_0(\mathcal{C}) \subset \text{tor}_1(\mathcal{C}) \subset \text{tor}_2(\mathcal{C}) . \quad (1)$$

Every code of length n over \mathbb{Z}_8 is equivalent to a code \mathcal{C} with generator matrix

$$\begin{bmatrix} I_{k_0} & A_{0,1} & A_{0,2} & A_{0,3} \\ 0 & 2I_{k_1} & 2A_{1,2} & 2A_{1,3} \\ 0 & 0 & 4I_{k_2} & 4A_{2,3} \end{bmatrix}$$

where $A_{i,j}$ are matrices of appropriate sizes with integer entries. Observe that $\text{tor}_i(\mathcal{C})$ is an $[n, \sum_{j=0}^i k_j]$ binary code. In fact, generator matrices of $\text{tor}_0(\mathcal{C})$, $\text{tor}_1(\mathcal{C})$, $\text{tor}_2(\mathcal{C})$ are given by

$$\begin{bmatrix} I_{k_0} & A_{0,1} & A_{0,2} & A_{0,3} \end{bmatrix}, \begin{bmatrix} I_{k_0} & A_{0,1} & A_{0,2} & A_{0,3} \\ 0 & I_{k_1} & A_{1,2} & A_{1,3} \end{bmatrix}, \begin{bmatrix} I_{k_0} & A_{0,1} & A_{0,2} & A_{0,3} \\ 0 & I_{k_1} & A_{1,2} & A_{1,3} \\ 0 & 0 & I_{k_2} & A_{2,3} \end{bmatrix} ,$$

respectively. We say that the code \mathcal{C} has type $\{k_0, k_1, k_2\}$ and then \mathcal{C} contains $8^{k_0} 4^{k_1} 2^{k_2}$ codewords.

If \mathcal{C} is self-orthogonal, then it is easy to prove that

$$\text{tor}_0(\mathcal{C}) \subset \text{tor}_2(\mathcal{C})^\perp , \quad (2)$$

and

$$\text{tor}_1(\mathcal{C}) \subset \text{tor}_1(\mathcal{C})^\perp . \quad (3)$$

The Euclidean weight $\text{wt}_e(x)$ of an element $x \in \mathbb{Z}_8$ is defined by $\text{wt}_e(0) = 0$, $\text{wt}_e(1) = \text{wt}_e(7) = 1$, $\text{wt}_e(2) = \text{wt}_e(6) = 4$, $\text{wt}_e(3) = \text{wt}_e(5) = 9$, and $\text{wt}_e(4) =$

16. The Euclidean weight of a vector in \mathbb{Z}_8^n is the sum of the Euclidean weights of its components. A code over \mathbb{Z}_8 whose codewords have Euclidean weights divisible by 16 is said to be an even code. Every even code is self-orthogonal. A self-orthogonal code over \mathbb{Z}_8 with generator matrix all of whose row vectors have Euclidean weight divisible by 16 is even by [1, Lemma 2.2]. An even self-dual code is also called a Type II code. It is known that a Type II code of length n over \mathbb{Z}_8 exists if and only if n is a multiple of 8 [1, Proposition 3.4].

Lemma 1. *Let \mathcal{C} be an even code over \mathbb{Z}_8 . Then $\text{tor}_1(\mathcal{C})$ is doubly even.*

Proof. If $2v \in \mathcal{C}$, then $2v \cdot 2v \equiv 0 \pmod{16}$. Thus $\text{wt}(v \bmod 2) \equiv 0 \pmod{4}$. \square

Analogously, we can define torsion codes $\text{tor}_0(\mathcal{C})$ and $\text{tor}_1(\mathcal{C})$ for a code \mathcal{C} over \mathbb{Z}_4 . It is known that if \mathcal{C} is a self-orthogonal code over \mathbb{Z}_4 , then $\text{tor}_0(\mathcal{C})$ is doubly even and $\text{tor}_0(\mathcal{C}) \subset \text{tor}_1(\mathcal{C}) \subset \text{tor}_0(\mathcal{C})^\perp$ (see [3]). The code over \mathbb{Z}_4 obtained from a code \mathcal{C} over \mathbb{Z}_8 by modulo 4 reduction is called the 4-residue of \mathcal{C} which we will denote by $\text{Res}(\mathcal{C})$. Then we have

$$\text{tor}_i(\text{Res}(\mathcal{C})) = \text{tor}_i(\mathcal{C}), \text{ for } i = 0, 1. \quad (4)$$

The Euclidean weights of $0, 1, 2, 3 \in \mathbb{Z}_4$ are $0, 1, 4, 1$, respectively, and the Euclidean weight of a vector in \mathbb{Z}_4^n is the sum of the Euclidean weights of its components. A code over \mathbb{Z}_4 is even if the Euclidean weight of every codeword is divisible by 8 (see [2]). The proof of the following lemma is analogous to that of Lemma 1.

Lemma 2. *If \mathcal{C} is an even code over \mathbb{Z}_4 , then every codeword in $\text{tor}_1(\mathcal{C})$ has an even weight.*

3 Results on Binary Codes

In this section, we collect results concerning binary codes which will be needed in later sections. Let $\sigma_1(n, k)$ be the number of distinct doubly even binary codes of length n and dimension k containing $\mathbf{1}$, and let $\sigma'_1(n, k)$ be the number of distinct doubly even binary codes of length n and dimension k not containing $\mathbf{1}$. The values of $\sigma_1(n, k)$ and $\sigma'_1(n, k)$ are given in [5].

Lemma 3. *Let \mathcal{C} be an $[n, m]$ doubly even binary code.*

(i) *If $\mathbf{1} \in \mathcal{C}$, then \mathcal{C} is contained in $\tau_1(n, m, k)$ doubly even $[n, m+k]$ binary codes, where*

$$\tau_1(n, m, k) = \begin{cases} \prod_{i=0}^{k-1} \frac{2^{n-2i-2m-1} + 2^{\frac{n}{2}-i-m-1} - 1}{2^{i+1} - 1}, & \text{if } n \equiv 0 \pmod{8}, \\ \prod_{i=0}^{k-1} \frac{2^{n-2i-2m-1} - 2^{\frac{n}{2}-i-m-1} - 1}{2^{i+1} - 1}, & \text{if } n \equiv 4 \pmod{8}. \end{cases}$$

- (ii) If $\mathbf{1} \notin \mathcal{C}$, then \mathcal{C} is contained in $\tau'_1(n, m, k)$ doubly even $[n, m+k]$ binary codes, where

$$\tau'_1(n, m, k) = \begin{cases} \prod_{i=0}^{k-1} \frac{2^{n-2i-2m-2} - 1}{2^{i+1} - 1}, & \text{if } n \equiv 2 \pmod{4}, \\ \tau_1(n, m+1, k-1) \\ + 2^k \tau_1(n, m+1, k), & \text{if } n \equiv 0 \pmod{4}. \end{cases}$$

Proof. Computations of $\tau_1(n, m, k)$ for $n \equiv 0 \pmod{4}$ and $\tau'_1(n, m, k)$ for $n \equiv 2 \pmod{4}$ follow the line of the proof of [7, Theorem 9.5.5(ii)]. Suppose $\mathbf{1} \notin \mathcal{C}$ and $n \equiv 0 \pmod{4}$. Then there are $\tau_1(n, m+1, k-1)$ doubly even $[n, m+k]$ codes containing $\langle \mathcal{C}, \mathbf{1} \rangle$. In order to count the number of doubly even $[n, m+k]$ codes \mathcal{D} with $\mathcal{C} \subset \mathcal{D}$ and $\mathbf{1} \notin \mathcal{D}$, observe that there are $\tau_1(n, m+1, k)$ doubly even $[n, m+k+1]$ codes $\tilde{\mathcal{D}}$ containing $\langle \mathcal{C}, \mathbf{1} \rangle$. Each such $\tilde{\mathcal{D}}$ contains 2^k doubly even $[n, m+k]$ codes \mathcal{D} with $\mathcal{C} \subset \mathcal{D}$ and $\mathbf{1} \notin \mathcal{D}$. The result follows by adding the two cases. \square

Lemma 4. Let \mathcal{C} be a binary self-orthogonal code of length n and dimension k with generator matrix $[I \ A]$. Then $\mathbf{1}_n$ belongs to \mathcal{C} if and only if $\mathbf{1}_{n-k}$ belongs to the row space of A .

Proof. Clearly, if $\mathbf{1}_n$ belongs to \mathcal{C} , then $\mathbf{1}_{n-k}$ belongs to the row space of A . Conversely, suppose $\mathbf{1}_{n-k} = \mathbf{u}A$ for some $\mathbf{u} \in \mathbb{Z}_2^k$. Since \mathcal{C} is self-orthogonal, we have $A\mathbf{1}_{n-k}^t = \mathbf{1}_k^t$. Thus

$$\begin{aligned} \mathbf{u} &= [\mathbf{u} \ \mathbf{1}_{n-k}] [I \ A]^t + (A\mathbf{1}_{n-k}^t)^t \\ &= \mathbf{u} [I \ A] [I \ A]^t + \mathbf{1}_k \\ &= \mathbf{1}_k. \end{aligned}$$

This implies $\mathbf{1}_n = \mathbf{1}_k [I \ A] \in \mathcal{C}$. \square

To conclude this section, we give a lemma which will be useful when counting the number of generator matrices of codes. Since this lemma holds in greater generality, we formulate it for an arbitrary field of characteristic 2.

Let K be a field, m a positive integer. We denote by $\text{Sym}_m(K)$ the set of $m \times m$ symmetric matrices over K . For a square matrix A of order n and a vector v of length n , we denote by $\text{diag}(A)$ the n -dimensional vector composed by the diagonal entries of A , $\text{Diag}(A)$ the diagonal matrix whose diagonal entries are those of A , and by $\text{Diag}(v)$ the diagonal matrix whose diagonal entries are those of v .

Lemma 5. Let K be a field of characteristic 2, $A \in M_{m \times n}(K)$, and $\text{rank } A = m$. Let $\alpha : M_{m \times n}(K) \rightarrow K^m$ be the mapping defined by $\alpha N = \mathbf{1}N^t$. Define a mapping

$$\begin{aligned} \Phi_A : M_{m \times n}(K) &\rightarrow \text{Sym}_m(K) \\ N &\mapsto AN^t + NA^t + \text{Diag}(AN^t). \end{aligned} \tag{5}$$

If the vector $\mathbf{1}$ does not belong to the row space of A , then the mappings

$$\begin{aligned}\Phi_A \oplus \alpha : M_{m \times n}(K) &\rightarrow \text{Sym}_m(K) \oplus K^m \\ N &\mapsto (AN^t + NA^t + \text{Diag}(AN^t), \mathbf{1}N^t)\end{aligned}$$

and

$$\begin{aligned}\Theta_A : M_{m \times n}(K) &\rightarrow \text{Sym}_m(K) \\ N &\mapsto AN^t + NA^t + \text{Diag}((A + J)N^t)\end{aligned}\tag{6}$$

are surjective.

Proof. The surjectivity of $\Phi_A \oplus \alpha$ has been established in [2, Lemma 3.2]. Since the mapping $\gamma : \text{Sym}_m(K) \oplus K^m \rightarrow \text{Sym}_m(K)$ defined by $\gamma(S, v) = S + \text{Diag}(v)$ is surjective, the surjectivity of Θ_A follows from that of $\Phi_A \oplus \alpha$ since $\Theta_A = \gamma \circ (\Phi_A \oplus \alpha)$. \square

4 Quaternary Even Codes

The number of even quaternary codes whose residue contains $\mathbf{1}$ can be found in [2]. We will be concerned with the enumeration of even quaternary codes whose residue does not contain $\mathbf{1}$. Let $\mathcal{C}_1, \mathcal{C}_2$ be binary codes of length n such that $\mathbf{1} \notin \mathcal{C}_1$, \mathcal{C}_1 is doubly even, and has generator matrix

$$\begin{bmatrix} I_{k_0} & A \end{bmatrix}, \tag{7}$$

\mathcal{C}_2 has generator matrix

$$\begin{bmatrix} I_{k_0} & A \\ 0 & B \end{bmatrix}, \tag{8}$$

$A \in M_{k_0 \times (n-k_0)}(\mathbb{Z})$, $B \in M_{k_1 \times (n-k_0)}(\mathbb{Z})$ and $\dim \mathcal{C}_1 = k_0$, $\dim \mathcal{C}_2 = k_0 + k_1$. Moreover, we assume that $\mathcal{C}_1 \subset \mathcal{C}_2 \subset \mathcal{C}_1^\perp$, and the rows of the matrix B mod 2 have even weights. Then the matrices A and B satisfy

$$I_{k_0} + AA^t \equiv 0 \pmod{2}, \tag{9}$$

$$AB^t \equiv 0 \pmod{2}, \tag{10}$$

$$\mathbf{1}B^t \equiv 0 \pmod{2}, \tag{11}$$

$$\text{diag}(I_{k_0} + AA^t) \equiv 0 \pmod{4}. \tag{12}$$

Lemma 6. For $N \in M_{k_0 \times (n-k_0)}(\mathbb{Z})$, the code $\mathcal{C} = \mathbb{Z}_4^{k_0} [I_{k_0} A + 2N]$ is an even code if and only if

$$\Theta_A(N \bmod 2) = \frac{1}{2}(I_{k_0} + AA^t) + \frac{1}{4} \text{Diag}(I_{k_0} + AA^t) \bmod 2, \tag{13}$$

where $\Theta_A : M_{m \times n}(\mathbb{Z}_2) \rightarrow \text{Sym}_m(\mathbb{Z}_2)$ is the mapping defined in (6).

Proof. Note that the right-hand side of (13) makes sense by (9). Moreover, the diagonal entries of $\frac{1}{2}(I_{k_0} + AA^t) \bmod 2$ are zero. The code \mathcal{C} is self-orthogonal if and only if

$$I_{k_0} + AA^t + 2(AN^t + NA^t) \equiv 0 \pmod{4},$$

which is equivalent to

$$AN^t + NA^t \equiv \frac{1}{2}(I_{k_0} + AA^t) \pmod{2}. \quad (14)$$

Moreover, when \mathcal{C} is self-orthogonal, \mathcal{C} is even if and only if

$$\text{Diag}(I_{k_0} + AA^t + 2(AN^t + NA^t) + 4NN^t) \equiv 0 \pmod{8}. \quad (15)$$

Since $\text{Diag}(AN^t) = \text{Diag}(NA^t)$ and $\text{diag } NN^t \equiv \mathbf{1}N^t \pmod{2}$, this is equivalent to

$$\text{Diag}((A + J)N^t) \equiv \frac{1}{4} \text{Diag}(I_{k_0} + AA^t) \pmod{2}. \quad (16)$$

Since (16) and (14) equate the diagonal entries, the off-diagonal entries, respectively, of (13), we obtain the desired result. \square

Let us consider the sets

$$\begin{aligned} X &= \{\mathcal{C} \mid \mathcal{C} \text{ is quaternary even, } \text{res}(\mathcal{C}) = \text{tor}_1(\mathcal{C}) = \mathcal{C}_1\}, \\ X' &= \{\mathcal{C}' \mid \mathcal{C}' \text{ is quaternary even, } \text{res}(\mathcal{C}') = \mathcal{C}_1, \text{tor}_1(\mathcal{C}') = \mathcal{C}_2\}. \end{aligned}$$

Lemma 7. *We have $|X| = 2^{k_0(2n-3k_0-1)/2}$.*

Proof. Since $\mathbf{1} \notin \mathcal{C}_1$, Lemma 4 implies that $\mathbf{1}$ does not belong to the row space of A over \mathbb{Z}_2 . Thus, the mapping Θ_A is surjective by Lemma 5. By Lemma 6, we have

$$\begin{aligned} |X| &= \left| \Theta_A^{-1} \left(\frac{1}{2}(I + AA^t) + \frac{1}{4} \text{Diag}(I + AA^t) \bmod 2 \right) \right| \\ &= |\text{Ker } \Theta_A| \\ &= |M_{k_0 \times (n-k_0)}(\mathbb{Z}_2)| / |\text{Sym}_{k_0}(\mathbb{Z}_2)|, \end{aligned}$$

and this gives the desired result. \square

Lemma 8. *If $\mathcal{C} \in X$, then there exists a unique $\mathcal{C}' \in X'$ containing \mathcal{C} .*

Proof. Let $[I \ A + 2N]$ be a generator matrix of \mathcal{C} , where $N \in M_{k_0 \times (n-k_0)}(\mathbb{Z})$. Then the code \mathcal{C}' with generator matrix

$$\begin{bmatrix} I_{k_0} & A + 2N \\ 0 & 2B \end{bmatrix} \quad (17)$$

is even, since the rows of the matrix $B \bmod 2$ has even weights. \square

Lemma 9. Let $\mathcal{C}' \in X'$. Then $|\{\mathcal{C} \in X \mid \mathcal{C} \subset \mathcal{C}'\}| = 2^{k_0 k_1}$.

Proof. Let (17) be a generator matrix of \mathcal{C}' . Consider a mapping

$$\varphi : M_{k_0 \times k_1}(\mathbb{Z}_2) \rightarrow \{\mathcal{C} \in X \mid \mathcal{C} \subset \mathcal{C}'\}$$

$$M \bmod 2 \mapsto \mathbb{Z}_4^{k_0} [I_{k_0} A + 2(N + MB)] .$$

Suppose $M_1, M_2 \in M_{k_0 \times k_1}(\mathbb{Z}_2)$. Since $\text{rank}(B \bmod 2) = k_1$, $M_1 \equiv M_2 \pmod{2}$ if and only if $M_1 B \equiv M_2 B \pmod{2}$. The latter is easily seen to be equivalent to $\varphi(M_1 \bmod 2) = \varphi(M_2 \bmod 2)$. This shows that φ is well-defined and injective. Next we show that φ is surjective. Suppose $\mathcal{C} \in X$ and $\mathcal{C} \subset \mathcal{C}'$. Then $\mathcal{C} = \mathbb{Z}_4^{k_0} [I_{k_0} A + 2F]$ for some matrix F . Since $\mathcal{C} \subset \mathcal{C}'$, $A + 2F \equiv A + 2N + 2MB \pmod{4}$ for some matrix M . Then we have $F \equiv N + MB \pmod{2}$, so we conclude that φ is surjective. Therefore, the mapping φ is bijective, and the result follows. \square

Theorem 1. Let \mathcal{C}_1 and \mathcal{C}_2 be binary codes of length n . Suppose

$$\mathcal{C}_1 \text{ is doubly even, } \dim \mathcal{C}_1 = k_0, \mathbf{1} \notin \mathcal{C}_1 \text{ and} \quad (18)$$

$$\mathcal{C}_1 \subset \mathcal{C}_2 \subset \mathcal{C}_1^\perp, \dim \mathcal{C}_2 = k_0 + k_1, \mathcal{C}_2 \subset \mathbf{1}^\perp. \quad (19)$$

Then the number of quaternary even codes \mathcal{C}' such that $\text{res}(\mathcal{C}') = \mathcal{C}_1$ and $\text{tor}_1(\mathcal{C}') = \mathcal{C}_2$ is $2^{k_0(2n-3k_0-2k_1-1)/2}$.

Proof. We may assume without loss of generality that \mathcal{C}_1 and \mathcal{C}_2 are codes with generator matrices given by (7) and (8), respectively. By Lemma 8 and Lemma 9, we have

$$\begin{aligned} 2^{k_0 k_1} |X'| &= \sum_{\mathcal{C}' \in X'} |\{\mathcal{C} \in X \mid \mathcal{C} \subset \mathcal{C}'\}| \\ &= \sum_{\mathcal{C} \in X} |\{\mathcal{C}' \in X' \mid \mathcal{C} \subset \mathcal{C}'\}| \\ &= |X|. \end{aligned}$$

The result then follows from Lemma 7. \square

Corollary 1. Let $\check{D}_4(n; k_0, k_1)$ (resp. $\check{D}'_4(n; k_0, k_1)$) denote the number of distinct even codes \mathcal{C} of length n over \mathbb{Z}_4 of type $\{k_0, k_1\}$ such that $\text{tor}_1(\mathcal{C})$ is doubly even and $\mathbf{1} \in \text{res}(\mathcal{C})$ (resp. $\mathbf{1} \notin \text{res}(\mathcal{C})$). Then

$$\check{D}_4(n; k_0, k_1) = \sigma_{\mathbf{1}}(n, k_0) \tau_{\mathbf{1}}(n, k_0, k_1) \cdot 2^{(n-k_0-k_1)+(k_0-1)(2n-3k_0-2k_1-2)/2}, \quad (20)$$

$$\check{D}'_4(n; k_0, k_1) = \sigma'_{\mathbf{1}}(n, k_0) \tau'_{\mathbf{1}}(n, k_0, k_1) \cdot 2^{k_0(2n-3k_0-2k_1-1)/2}. \quad (21)$$

Proof. If \mathcal{C} is an even code of length n over \mathbb{Z}_4 of type $\{k_0, k_1\}$, with residue code not containing $\mathbf{1}$, then by setting $\mathcal{C}_1 = \text{res}(\mathcal{C})$ and $\mathcal{C}_2 = \text{tor}_1(\mathcal{C})$, we see that \mathcal{C}_1 and \mathcal{C}_2 satisfy (18)–(19) by [3] and Lemma 2. Thus Theorem 1 implies

$$\begin{aligned} & \check{D}'_4(n; k_0, k_1) 2^{-k_0(2n-3k_0-1-2k_1)/2} \\ &= \sum_{\substack{\mathcal{C}_1 \text{ satisfies (18)} \\ \mathcal{C}_2 \text{ satisfies (19)}}} |\{\mathcal{C}_2 \mid \mathcal{C}_2 \text{ satisfies (19) and } \mathcal{C}_2 \text{ is doubly even}\}| \\ &= \tau'_1(n, k_0, k_1) \sigma'_1(n, k_0). \end{aligned}$$

This proves (21). Similarly, using [2, Theorem 5.7], we obtain (20). \square

5 Even Codes over \mathbb{Z}_8 with Prescribed 4-Residue and Torsion

Lemma 10. *Let \mathcal{C} be an even code over \mathbb{Z}_8 of type $\{k_0, k_1, k_2\}$, and let $\mathcal{C}_1 = \text{Res}(\mathcal{C})$ and $\mathcal{C}_2 = \text{tor}_2(\mathcal{C})$. Then*

$$\mathcal{C}_1 \text{ is even, of type } \{k_0, k_1\}, \text{ tor}_1(\mathcal{C}_1) \text{ is doubly even, and} \quad (22)$$

$$\text{tor}_1(\mathcal{C}_1) \subset \mathcal{C}_2 \subset \text{res}(\mathcal{C}_1)^\perp, \dim \mathcal{C}_2 = k_0 + k_1 + k_2. \quad (23)$$

Proof. We see that (22) holds by [4, Lemma 4.5] and Lemma 1. Also, (23) holds by (1), (2) and (4). \square

For the remainder of this section, we let \mathcal{C}_1 be an even code of length n over \mathbb{Z}_4 of type $\{k_0, k_1\}$ and \mathcal{C}_2 be a binary code of length n with $\dim \mathcal{C}_2 = k_0 + k_1 + k_2$. Then we may assume without loss of generality that \mathcal{C}_1 has generator matrix

$$\begin{bmatrix} I_{k_0} & A_{0,1} & A_{0,2} \\ 0 & 2I_{k_1} & 2A_{1,2} \end{bmatrix}, \quad (24)$$

and \mathcal{C}_2 has generator matrix

$$\begin{bmatrix} I_{k_0} & A_{0,1} & A_{0,2} \\ 0 & I_{k_1} & A_{1,2} \\ 0 & 0 & A_{2,2} \end{bmatrix}. \quad (25)$$

We assume that the matrix $A_{2,2}$ has k_2 rows and $\text{rank}(A_{2,2} \bmod 2) = k_2$. Then the matrices $A_{0,1}$, $A_{0,2}$, $A_{1,2}$ and $A_{2,2}$ satisfy

$$I_{k_0} + A_{0,1}A_{0,1}^t + A_{0,2}A_{0,2}^t \equiv 0 \pmod{4}, \quad (26)$$

$$A_{0,1} + A_{0,2}A_{1,2}^t \equiv 0 \pmod{2}, \quad (27)$$

$$A_{0,2}A_{2,2}^t \equiv 0 \pmod{2}, \quad (28)$$

$$I_{k_1} + A_{1,2}A_{1,2}^t \equiv 0 \pmod{2}, \quad (29)$$

$$\text{Diag}(I_{k_0} + A_{0,1}A_{0,1}^t + A_{0,2}A_{0,2}^t) \equiv 0 \pmod{8}, \quad (30)$$

$$\text{Diag}(I_{k_1} + A_{1,2}A_{1,2}^t) \equiv 0 \pmod{4}. \quad (31)$$

Lemma 11. *If \mathcal{C} is of type $\{k_0, k_1, 0\}$ and $\text{Res}(\mathcal{C}) = \mathcal{C}_1$, then there exist matrices N_0, N_1 such that*

$$\begin{bmatrix} I_{k_0} & A_{0,1} & A_{0,2} + 4N_0 \\ 0 & 2I_{k_1} & 2A_{1,2} + 4N_1 \end{bmatrix} \quad (32)$$

is a generator matrix of \mathcal{C} . Such matrices N_0 and N_1 are unique modulo 2.

Proof. By the assumption,

$$\mathcal{C} = \mathbb{Z}_8^{k_0+k_1} \begin{bmatrix} I_{k_0} + 4M_1 & A_{0,1} + 4M_2 & A_{0,2} + 4M_3 \\ 0 & 2I_{k_1} + 4M_4 & 2A_{1,2} + 4M_5 \end{bmatrix}$$

for some matrices M_1, \dots, M_5 . Taking $N_0 = M_3 - M_1 A_{0,2} - M_2 A_{1,2} + M_1 A_{0,1} A_{1,2}$ and $N_1 = M_5 - M_4 A_{1,2}$, we see that \mathcal{C} has a generator matrix (32). Since the matrices $4N_0, 4N_1$ are unique modulo 8, N_0, N_1 are unique modulo 2. \square

Lemma 12. *Let \mathcal{C} be a code with generator matrix (32). Define mappings*

$$\begin{aligned} \beta : M_{k_1 \times (n-k_0-k_1)}(\mathbb{Z}_2) &\rightarrow M_{k_0 \times k_1}(\mathbb{Z}_2) \\ N_1 \bmod 2 &\mapsto A_{0,2} N_1^t \bmod 2 \end{aligned}$$

and $\Phi_{A_{0,2}}$ as in (5). Then \mathcal{C} is an even code if and only if

$$\begin{aligned} \Phi_{A_{0,2}}(N_0 \bmod 2) &= \left(\frac{1}{4}(I_{k_0} + A_{0,1} A_{0,1}^t + A_{0,2} A_{0,2}^t) \right. \\ &\quad \left. + \frac{1}{8} \text{Diag}(I_{k_0} + A_{0,1} A_{0,1}^t + A_{0,2} A_{0,2}^t) \right) \bmod 2 , \end{aligned} \quad (33)$$

and

$$\beta(N_1 \bmod 2) = \frac{1}{2}(A_{0,1} + A_{0,2} A_{1,2}^t) \bmod 2 . \quad (34)$$

Proof. Note that the right-hand side of (33) makes sense by (26). Moreover the diagonal entries of $\frac{1}{4}(I_{k_0} + A_{0,1} A_{0,1}^t + A_{0,2} A_{0,2}^t) \bmod 2$ are zero. In view of (29), the code \mathcal{C} is self-orthogonal if and only if

$$I_{k_0} + A_{0,1} A_{0,1}^t + A_{0,2} A_{0,2}^t + 4(A_{0,2} N_0^t + N_0 A_{0,2}^t) \equiv 0 \pmod{8} , \quad (35)$$

$$A_{0,1} + A_{0,2} A_{1,2}^t + 2A_{0,2} N_1^t \equiv 0 \pmod{4} . \quad (36)$$

Observe that (36) is equivalent to (34), while (35) is equivalent to

$$A_{0,2} N_0^t + N_0 A_{0,2}^t = \frac{1}{4}(I_{k_0} + A_{0,1} A_{0,1}^t + A_{0,2} A_{0,2}^t) \bmod 2 . \quad (37)$$

Moreover, when \mathcal{C} is self-orthogonal, \mathcal{C} is even if and only if

$$\text{Diag}(I_{k_0} + A_{0,1} A_{0,1}^t + A_{0,2} A_{0,2}^t + 4(A_{0,2} N_0^t + N_0 A_{0,2}^t)) \equiv 0 \pmod{16} , \quad (38)$$

$$\text{Diag}(I_{k_1} + A_{1,2} A_{1,2}^t + 2(A_{1,2} N_1^t + N_1 A_{1,2}^t)) \equiv 0 \pmod{4} . \quad (39)$$

Since $\text{Diag} A_{1,2} N_1^t = \text{Diag} N_1 A_{1,2}^t$, (39) is equivalent to (31). In view of (30), (38) can be written as

$$\text{Diag}(A_{0,2} N_0^t) \equiv \frac{1}{8} \text{Diag}(I_{k_0} + A_{0,1} A_{0,1}^t + A_{0,2} A_{0,2}^t) \pmod{2} . \quad (40)$$

Since (40), (37) equate the diagonal entries, off diagonal entries, respectively of (33), we obtain the desired result. \square

Let us consider the sets

$$\begin{aligned} Y &= \{\mathcal{C} \mid \mathcal{C} \text{ is even, of type } \{k_0, k_1, 0\}, \text{Res}(\mathcal{C}) = \mathcal{C}_1\} , \\ Y' &= \{\mathcal{C}' \mid \mathcal{C}' \text{ is even, of type } \{k_0, k_1, k_2\}, \text{Res}(\mathcal{C}') = \mathcal{C}_1, \text{tor}_2(\mathcal{C}') = \mathcal{C}_2\} . \end{aligned}$$

Lemma 13. *We have $|Y| = 2^{(k_0+k_1)(n-k_0-k_1)-k_0(k_0+1)/2-k_0k_1}$.*

Proof. Note that β is surjective since $A_{0,2}$ is of full row rank. By Lemma 12, we have

$$\begin{aligned} |Y| &= |\text{Ker } \Phi_{A_{0,2}}| \cdot |\text{Ker } \beta| \\ &= \frac{|M_{k_0 \times (n-k_0-k_1)}(\mathbb{Z}_2)|}{|\text{Sym}_{k_0}(\mathbb{Z}_2)|} \cdot \frac{|M_{k_1 \times (n-k_0-k_1)}(\mathbb{Z}_2)|}{|M_{k_0 \times k_1}(\mathbb{Z}_2)|} \\ &= 2^{k_0(n-k_0-k_1)-k_0(k_0+1)/2} \cdot 2^{k_1(n-k_0-k_1)-k_0k_1}. \end{aligned}$$

\square

Lemma 14. *If $\mathcal{C} \in Y$, then there exists a unique $\mathcal{C}' \in Y'$ containing \mathcal{C} .*

Proof. By Lemma 11, \mathcal{C} has a generator matrix (32) for some matrices N_0, N_1 . Then the code \mathcal{C}' with generator matrix

$$\begin{bmatrix} I & A_{0,1} & A_{0,2} + 4N_0 \\ 0 & 2I & 2A_{1,2} + 4N_1 \\ 0 & 0 & 4A_{2,2} \end{bmatrix} \quad (41)$$

satisfies $\text{Res}(\mathcal{C}') = \mathcal{C}_1$ and $\text{tor}_2(\mathcal{C}') = \mathcal{C}_2$. Since $\mathcal{C} \in Y$, (28) implies that \mathcal{C}' is self-orthogonal. Observe that $[0 \ 0 \ 4A_{2,2}]$ has rows with Euclidean weights divisible by 16, so \mathcal{C}' is even, hence $\mathcal{C}' \in Y'$. If $\mathcal{C}'' \in Y'$ and $\mathcal{C} \subset \mathcal{C}''$, then $\mathbb{Z}_8^{k_2} [0 \ 0 \ 4A_{2,2}] \subset \mathcal{C}''$. This forces $\mathcal{C}' = \mathcal{C}''$. \square

Lemma 15. *Let $\mathcal{C}' \in Y'$. Then $|\{\mathcal{C} \in Y \mid \mathcal{C} \subset \mathcal{C}'\}| = 2^{(k_0+k_1)k_2}$.*

Proof. Suppose that $\mathcal{C}' \in Y'$ has generator matrix (41). Consider the mapping $\varphi : M_{k_0 \times k_2}(\mathbb{Z}_2) \times M_{k_1 \times k_2}(\mathbb{Z}_2) \rightarrow \{\mathcal{C} \in Y \mid \mathcal{C} \subset \mathcal{C}'\}$ defined by

$$\varphi(M \bmod 2, M' \bmod 2) = \mathbb{Z}_8^{k_0+k_1} \begin{bmatrix} I & A_{0,1} & A_{0,2} + 4(N_0 + MA_{2,2}) \\ 0 & 2I & 2A_{1,2} + 4(N_1 + M'A_{2,2}) \end{bmatrix} .$$

We claim that φ is bijective. Indeed, φ is injective since $A_{2,2} \bmod 2$ has full row rank. Next we show that φ is surjective. Suppose $\mathcal{C} \in Y$ and $\mathcal{C} \subset \mathcal{C}'$. Then by

Lemma 11, $\mathcal{C} = \mathbb{Z}_8^{k_0+k_1} \begin{bmatrix} I & A_{0,1} & A_{0,2} + 4F_1 \\ 0 & 2I & 2A_{1,2} + 4F_2 \end{bmatrix}$ for some matrices F_1, F_2 . Since $\mathcal{C} \subset \mathcal{C}'$,

$$\begin{bmatrix} I & A_{0,1} & A_{0,2} + 4F_1 \\ 0 & 2I & 2A_{1,2} + 4F_2 \end{bmatrix} \equiv \begin{bmatrix} I & 0 & M \\ 0 & I & M' \end{bmatrix} \begin{bmatrix} I & A_{0,1} & A_{0,2} + 4N_0 \\ 0 & 2I & 2A_{1,2} + 4N_1 \\ 0 & 0 & 4A_{2,2} \end{bmatrix} \pmod{8}$$

for some matrices M and M' . Then we have $F_1 \equiv N_0 + MA_{2,2} \pmod{2}$, and $F_2 \equiv N_1 + M'A_{2,2} \pmod{2}$, so we conclude φ is surjective. Therefore, φ is bijective. It follows that the number of codes $\mathcal{C} \in Y$ contained in \mathcal{C}' is $2^{(k_0+k_1)k_2}$. \square

Theorem 2. Let \mathcal{C}_1 be an even code of length n over \mathbb{Z}_4 of type $\{k_0, k_1\}$, and let \mathcal{C}_2 be a binary code of length n . Suppose \mathcal{C}_1 and \mathcal{C}_2 satisfy (22)–(23). Then the number of even codes \mathcal{C}' of length n over \mathbb{Z}_8 such that $\text{Res}(\mathcal{C}') = \mathcal{C}_1$ and $\text{tor}_2(\mathcal{C}') = \mathcal{C}_2$ is $2^{k_0(2n-3k_0-6k_1-2k_2-1)/2+k_1(n-k_1-k_2)}$.

Proof. We may assume without loss of generality that \mathcal{C}_1 and \mathcal{C}_2 are codes with generator matrices given by (24) and (25), respectively. Then $|Y'|$ is the number we need to compute. By Lemmas 14–15, we have

$$\begin{aligned} 2^{(k_0+k_1)k_2} |Y'| &= \sum_{\mathcal{C}' \in Y'} |\{\mathcal{C} \in Y \mid \mathcal{C} \subset \mathcal{C}'\}| \\ &= \sum_{\mathcal{C} \in Y} |\{\mathcal{C}' \in Y' \mid \mathcal{C} \subset \mathcal{C}'\}| \\ &= |Y|. \end{aligned}$$

The result then follows from Lemma 13. \square

For $k \leq n$, we define the Gaussian 2-binomial coefficient $\begin{bmatrix} n \\ k \end{bmatrix}_2$ as

$$\begin{bmatrix} n \\ k \end{bmatrix}_2 = \frac{(2^n - 1)(2^n - 2) \cdots (2^n - 2^{k-1})}{(2^k - 1)(2^k - 2) \cdots (2^k - 2^{k-1})}.$$

This is the number of k -dimensional subspaces in an n -dimensional vector space over \mathbb{Z}_2 .

Corollary 2. Let $D_8(n; k_0, k_1, k_2)$ (resp. $D'_8(n; k_0, k_1, k_2)$) denote the number of distinct even codes \mathcal{C} over \mathbb{Z}_8 of length n of type $\{k_0, k_1, k_2\}$ such that $\mathbf{1} \in \text{res}(\mathcal{C})$ (resp. $\mathbf{1} \notin \text{res}(\mathcal{C})$). Then

$$\begin{aligned} D_8(n; k_0, k_1, k_2) &= \sigma_1(n, k_0) \tau_1(n, k_0, k_1) \begin{bmatrix} n - 2k_0 - k_1 \\ k_2 \end{bmatrix}_2 \\ &\quad \times 2^{1+k_0(2n-3k_0-4k_1-k_2-1)+k_1(n-k_1-k_2)}, \end{aligned} \tag{42}$$

$$\begin{aligned} D'_8(n; k_0, k_1, k_2) &= \sigma'_1(n, k_0) \tau'_1(n, k_0, k_1) \begin{bmatrix} n - 2k_0 - k_1 \\ k_2 \end{bmatrix}_2 \\ &\quad \times 2^{k_0(2n-3k_0-4k_1-k_2-1)+k_1(n-k_1-k_2)}. \end{aligned} \tag{43}$$

Proof. If \mathcal{C} is an even code of length n over \mathbb{Z}_8 of type $\{k_0, k_1, k_2\}$, then by setting $\mathcal{C}_1 = \text{Res}(\mathcal{C})$ and $\mathcal{C}_2 = \text{tor}_2(\mathcal{C})$, we see that \mathcal{C}_1 and \mathcal{C}_2 satisfy (22)–(23) by Lemma 10. Thus Theorem 2 implies

$$\begin{aligned} & D_8(n; k_0, k_1, k_2) 2^{-k_0(2n-3k_0-6k_1-2k_2-1)/2-k_1(n-k_1-k_2)} \\ &= \sum_{\substack{\mathcal{C}_1 \text{ satisfies (22)} \\ \mathbf{1} \in \text{res}(\mathcal{C}_1)}} |\{\mathcal{C}_2 \mid \mathcal{C}_2 \text{ satisfies (23)}\}| \\ &= \begin{bmatrix} n - 2k_0 - k_1 \\ k_2 \end{bmatrix}_2 |\{\mathcal{C}_1 \mid \mathcal{C}_1 \text{ satisfies (22), } \mathbf{1} \in \text{res}(\mathcal{C}_1)\}| \\ &= \begin{bmatrix} n - 2k_0 - k_1 \\ k_2 \end{bmatrix}_2 \sigma_{\mathbf{1}}(n, k_0) \tau_{\mathbf{1}}(n, k_0, k_1) \cdot 2^{(n-k_0-k_1)+(k_0-1)(2n-3k_0-2k_1-2)/2}, \end{aligned}$$

by (20). This proves (42). Similarly, using Theorem 2 and (21), we obtain (43). \square

Corollary 3. *Let n be a positive integer divisible by 8. Then the number of distinct Type II codes \mathcal{C} of length n over \mathbb{Z}_8 such that $\dim \text{res}(\mathcal{C}) = k_0$ and $\mathbf{1} \in \text{res}(\mathcal{C})$ (resp. $\mathbf{1} \notin \text{res}(\mathcal{C})$) is $\sigma_{\mathbf{1}}(n, k_0) \tau_{\mathbf{1}}(n, k_0, \frac{n}{2} - k_0) \cdot 2^{3k_0+1}$ (resp. $\sigma'_{\mathbf{1}}(n, k_0) \tau'_{\mathbf{1}}(n, k_0, \frac{n}{2} - k_0) \cdot 2^{3k_0}$).*

Proof. Let \mathcal{C} be a Type II code of length n over \mathbb{Z}_8 of type $\{k_0, k_1, k_2\}$. Then \mathcal{C} is self-dual, so $3n = 2(3k_0 + 2k_1 + k_2)$. By (2)–(3), we have $n \geq 2k_0 + k_1 + k_2$ and $n \geq 2(k_0 + k_1)$. Thus we have $n = 2(k_0 + k_1)$ and $k_1 = k_2$, and the result follows from Corollary 2. \square

As an example, we consider Type II codes over \mathbb{Z}_8 of length $n = 8$. By [5], we have

$$\begin{aligned} \sigma_{\mathbf{1}}(8, k_0) &= \prod_{i=0}^{k_0-2} \frac{2^{5-2i} + 2^{2-i} - 1}{2^{i+1} - 1}, \\ \sigma'_{\mathbf{1}}(8, k_0) &= \prod_{i=0}^{k_0-1} \frac{2^{6-2i} + 2^{3-i} - 2}{2^{i+1} - 1}. \end{aligned}$$

The numbers given in Corollary 3 are tabulated in Table 1.

Therefore, there are 668190 Type II codes over \mathbb{Z}_8 of length 8, which agrees with the result in [6].

Table 1. Type II Codes of Length 8

$\dim \text{res} \mathcal{C}$	The number of \mathcal{C} with	
	$\mathbf{1} \in \text{res} \mathcal{C}$	$\mathbf{1} \notin \text{res} \mathcal{C}$
0	0	30
1	480	3360
2	26880	53760
3	215040	122880
4	245760	0

References

1. Bannai, E., Dougherty, S.T., Harada, M., Oura, M.: Type II Codes, Even Unimodular Lattices and Invariant Rings. *IEEE Trans. Inform. Theory* 45, 1194–1205 (1999)
2. Betty, R.A.L., Munemasa, A.: Mass Formula for Self-Orthogonal Codes over \mathbb{Z}_{p^2} . *Journal of Combinatorics, Information and System Sciences* (to appear)
3. Conway, J.H., Sloane, N.J.A.: Self-Dual Codes over the Integers Modulo 4. *J. Combin. Theory Ser. A* 62, 30–45 (1993)
4. Dougherty, S.T., Gulliver, T.A., Wong, J.: Self-Dual Codes over \mathbb{Z}_8 and \mathbb{Z}_9 . *Designs, Codes and Cryptogr.* 41, 235–249 (2006)
5. Gaborit, P.: Mass Formulas for Self-Dual Codes over \mathbb{Z}_4 and $\mathbb{F}_q + u\mathbb{F}_q$ Rings. *IEEE Trans. Inform. Theory* 42, 1222–1228 (1996)
6. Harada, M., Munemasa, M.: On the Classification of Self-Dual \mathbb{Z}_k -Codes. In: Parker, M.G. (ed.) IMACC 2009. LNCS, vol. 5921, pp. 81–93. Springer, Heidelberg (2009)
7. Huffman, W.C., Pless, V.: Fundamentals of Error-Correcting Codes. Cambridge University Press, United Kingdom (2003)

On the Classification of Self-dual \mathbb{Z}_k -Codes

Masaaki Harada¹ and Akihiro Munemasa²

¹ Department of Mathematical Sciences, Yamagata University, Yamagata 990–8560, Japan and PRESTO, Japan Science and Technology Agency, Kawaguchi, Saitama 332–0012, Japan

² Graduate School of Information Sciences, Tohoku University, Sendai 980–8579, Japan

Abstract. A classification of self-dual \mathbb{Z}_k -codes of modest lengths is known for small k . For $k = 4, 6, 8, 9$ and 10 , the classification of self-dual \mathbb{Z}_k -codes is extended to lengths $19, 12, 12, 12$ and 10 , respectively, by considering k -frames of unimodular lattices.

1 Introduction

Let \mathbb{Z}_k be the ring of integers modulo k , where k is a positive integer greater than 1. A \mathbb{Z}_k -code C of length n (or a code C of length n over \mathbb{Z}_k) is a \mathbb{Z}_k -submodule of \mathbb{Z}_k^n . A code C is *self-dual* if $C = C^\perp$ where the dual code C^\perp of C is defined as $C^\perp = \{x \in \mathbb{Z}_k^n \mid x \cdot y = 0 \text{ for all } y \in C\}$ under the standard inner product $x \cdot y$. Two \mathbb{Z}_k -codes C and C' are *equivalent* if there exists a monomial $(\pm 1, 0)$ -matrix P with $C' = C \cdot P = \{xP \mid x \in C\}$. A *Type II* \mathbb{Z}_{2k} -code was defined in [2] as a self-dual code with the property that all Euclidean weights are divisible by $4k$ (see [2] for the definition of Euclidean weights). It is known that a Type II \mathbb{Z}_{2k} -code of length n exists if and only if n is divisible by eight [2]. A self-dual code which is not Type II is called *Type I*.

As described in [18], self-dual codes are an important class of linear codes for both theoretical and practical reasons. It is a fundamental problem to classify self-dual codes of modest lengths and much work has been done towards classifying self-dual \mathbb{Z}_k -codes for $k = 2$ and 3 (see [18]). Self-dual \mathbb{Z}_4 -codes have also been widely studied because such codes have applications to unimodular lattices and nonlinear binary codes. The classification of self-dual \mathbb{Z}_4 -codes was done for lengths $n \leq 9$ in [5] and for lengths $n = 10, \dots, 15$ in [9]. At length 16, the classification of Type II codes was done in [16]. For $k = 4, 6, 8, 9$ and 10 , the lengths for which a classification of self-dual \mathbb{Z}_k -codes is complete are listed in the second column of Table 1.

Recently, a classification method of self-dual \mathbb{Z}_k -codes based on a classification of k -frames of unimodular lattices have been given by the authors and Venkov [12]. In this paper, using this method, we give a classification of Type I \mathbb{Z}_4 -codes of lengths 16, 17, 18 and 19 (Section 3). For self-dual \mathbb{Z}_8 -codes, the classification was done in [7] for lengths $n \leq 8$. As pointed out in [14], the classification of self-dual codes of length 6 given in [7] misses some codes. In Section 4, we give a revised list of self-dual \mathbb{Z}_8 -codes of lengths 6 and 8 which are marked

Table 1. Classification of self-dual \mathbb{Z}_k -codes

\mathbb{Z}_k	Known classification	New classification
\mathbb{Z}_4	(Type II) 8, 16 [5], [16]	-
	(Type I) 1, 2, ..., 15 [5], [9]	16, 17, 18, 19
\mathbb{Z}_6	(Type II) 8 [13]	-
	(Type I) 4 [8], [15]	8, 12
\mathbb{Z}_8	(Type II) 8 [7]	8*
	(Type I) 2, 4, 6, 8 [7]	6*, 8*, 10, 12
\mathbb{Z}_9	1, 2, ..., 8 [1]	9, 10, 11, 12
\mathbb{Z}_{10}	(Type II)	8
	(Type I)	2, 4, 6, 8, 10

by * in Table 1. We also extend the classification to length 12. For other k , we give a classification of self-dual \mathbb{Z}_k -codes of length n for $(k, n) = (6, 8), (6, 12), (9, 9), \dots, (9, 12), (10, 2), (10, 4), \dots, (10, 10)$ (see the last column of Table 1). In Table 13, as a summary, we give the number of inequivalent self-dual \mathbb{Z}_k -codes whose lengths are listed in the last column of Table 1.

In order to save space, we list generator matrices of the inequivalent self-dual codes for small lengths only. Generator matrices of those codes which are not listed in this paper can be obtained electronically from [11]. All computer calculations in this paper were done by MAGMA [4].

2 Preliminaries

2.1 Unimodular Lattices

A (Euclidean) lattice L in dimension n is *integral* if $L \subseteq L^*$, where the dual lattice L^* is defined as $L^* = \{x \in \mathbb{R}^n | (x, y) \in \mathbb{Z} \text{ for all } y \in L\}$ under the standard inner product (x, y) . An integral lattice with $L = L^*$ is called *unimodular*. The *norm* of a vector x is (x, x) . The *minimum norm* of L is the smallest norm among all nonzero vectors of L . A unimodular lattice L is *even* if all vectors of L have even norms, and *odd* if some vector has an odd norm. Two lattices L and L' are *isomorphic*, denoted $L \simeq L'$, if there exists an orthogonal matrix A with $L' = L \cdot A$. An *automorphism* of L is an orthogonal matrix A with $L = L \cdot A$, and the *automorphism group* $\text{Aut}(L)$ of L is the group consisting of all automorphisms.

Our classification method of self-dual codes of length n (see Subsection 2.2) requires a classification of n -dimensional unimodular lattices. A set $\{L_{n,i} | i = 1, 2, \dots, l(n)\}$ of representatives of isomorphism classes of n -dimensional unimodular lattices used in our classification is listed in Table 2, where $l(n)$ denotes the number of non-isomorphic n -dimensional unimodular lattices. All unimodular lattices with minimum norm ≥ 2 for dimensions up to 23 can be found in Table 16.7 of [6]. We denote the unimodular lattices with minimum norm ≥ 2 by their components listed in that table.

Table 2. List of unimodular lattices

n	$l(n)$	Unimodular lattices $L_{n,1}, \dots, L_{n,l(n)}$
2, 4, 6	1	\mathbb{Z}^n
8	2	\mathbb{Z}^8, E_8
9	2	$\mathbb{Z}^9, E_8 \oplus \mathbb{Z}$
10	2	$\mathbb{Z}^{10}, E_8 \oplus \mathbb{Z}^2$
11	2	$\mathbb{Z}^{11}, E_8 \oplus \mathbb{Z}^3$
12	3	$D_{12}, \mathbb{Z}^{12}, E_8 \oplus \mathbb{Z}^4$
16 (odd)	6	$D_8^2, \mathbb{Z}^{16}, E_8 \oplus \mathbb{Z}^8, D_{12} \oplus \mathbb{Z}^4, E_7^2 \oplus \mathbb{Z}^2, A_{15} \oplus \mathbb{Z}$
17	9	$A_{11}E_6, E_8^2 \oplus \mathbb{Z}, D_{16} \oplus \mathbb{Z}, L_{16,1} \oplus \mathbb{Z}, \dots, L_{16,6} \oplus \mathbb{Z}$
18	13	$A_{17}A_1, D_{10}E_7A_1, D_6^3, A_9^2, L_{17,1} \oplus \mathbb{Z}, \dots, L_{17,9} \oplus \mathbb{Z}$

2.2 Classification Method

A set $\{f_1, \dots, f_n\}$ of n vectors f_1, \dots, f_n in an n -dimensional lattice L with $(f_i, f_j) = k\delta_{i,j}$ is called a k -frame of L , where $\delta_{i,j}$ is the Kronecker delta. The following construction of lattices from codes is called *Construction A*. If C is a \mathbb{Z}_k -code of length n then

$$A_k(C) = \frac{1}{\sqrt{k}} \{(x_1, \dots, x_n) \in \mathbb{Z}^n \mid (x_1 \bmod k, \dots, x_n \bmod k) \in C\}$$

is an n -dimensional lattice. In particular, C is Type I (resp. Type II) if and only if $A_k(C)$ is odd (resp. even) unimodular. Clearly, $A_k(C)$ has the k -frame $\{\sqrt{k}e_1, \sqrt{k}e_2, \dots, \sqrt{k}e_n\}$ where e_i denotes the i -th unit vector $(\delta_{i,1}, \delta_{i,2}, \dots, \delta_{i,n})$ ($i = 1, 2, \dots, n$). Let $\mathcal{F} = \{f_1, \dots, f_n\}$ be a k -frame of L . Consider the mapping

$$\begin{aligned} \pi_{\mathcal{F}} : \frac{1}{k} \bigoplus_{i=1}^n \mathbb{Z} f_i &\rightarrow \mathbb{Z}_k^n \\ \pi_{\mathcal{F}}(x) &= ((x, f_i) \bmod k)_{1 \leq i \leq n}. \end{aligned}$$

Then $\text{Ker } \pi_{\mathcal{F}} = \bigoplus_{i=1}^n \mathbb{Z} f_i \subset L$, so the code $C = \pi_{\mathcal{F}}(L)$ satisfies $\pi_{\mathcal{F}}^{-1}(C) = L$. This implies $A_k(C) \simeq L$, and every code C with $A_k(C) \simeq L$ is obtained as $\pi_{\mathcal{F}}(L)$ for some k -frame \mathcal{F} of L . Moreover, every Type I (resp. Type II) \mathbb{Z}_k -code of length n can be obtained from a certain k -frame in some n -dimensional odd (resp. even) unimodular lattice.

Lemma 1 ([12]). *Let L be an n -dimensional integral lattice, and let $\mathcal{F} = \{f_1, \dots, f_n\}$, $\mathcal{F}' = \{f'_1, \dots, f'_n\}$ be k -frames of L . Then the codes $\pi_{\mathcal{F}}(L)$ and $\pi_{\mathcal{F}'}(L)$ are equivalent if and only if there exists an automorphism P of L such that $\{\pm f_1, \dots, \pm f_n\} \cdot P = \{\pm f'_1, \dots, \pm f'_n\}$.*

We describe how to classify self-dual \mathbb{Z}_k -codes of length n explicitly by Lemma 1. Let L be an n -dimensional unimodular lattice and let V be the set of pairs $\{v, -v\}$ with $(v, v) = k$, $v \in L$. We define the undirected simple graph Γ , whose set of vertices is the set V and two vertices $\{v, -v\}, \{w, -w\} \in V$ are adjacent if $(v, w) = 0$.

It follows that the k -frames are precisely the n -cliques in the graph Γ . It is clear that the group $\text{Aut}(L)$ acts on the graph Γ as an automorphism group, and Lemma 1 implies that the $\text{Aut}(L)$ -orbits on the set of n -cliques of Γ are in one-to-one correspondence with the equivalence classes of codes C satisfying $A_k(C) \simeq L$. Therefore, the classification of such codes reduces to finding a set of representatives of n -cliques of Γ up to the action of $\text{Aut}(L)$. This computation was performed in MAGMA [4], the results were then converted to k -frames, and then to self-dual codes of length n . In principle, such a computation can be done by enumerating cliques by the MAGMA function `AllCliques`, then by classifying them up to $\text{Aut}(L)$ -action by `IsConjugate`. In some cases, we also tried to minimize memory usage when $\text{Aut}(L)$ is large. In this way, by considering k -frames of all non-isomorphic n -dimensional unimodular lattices, we have all inequivalent self-dual \mathbb{Z}_k -codes of length n for the cases (k, n) listed in the last column of Table 1.

2.3 Automorphism Groups of Codes

An *automorphism* of C is a monomial $(\pm 1, 0)$ -matrix P with $C = C \cdot P$, and the *automorphism group* $\text{Aut}(C)$ of C is the group consisting of all automorphisms. In this subsection, we describe how to determine the order of the automorphism group $\text{Aut}(C)$ of a self-dual \mathbb{Z}_k -code C by modifying the computation in the previous subsection.

Let X be a generating subset of the Euclidean space \mathbb{R}^n satisfying $X = -X$. Define \overline{X} by

$$\overline{X} = \{\{x, -x\} \mid x \in X\} .$$

Let G be a subgroup of the orthogonal group $O(n, \mathbb{R})$ leaving X invariant. Let \overline{G} denote the permutation group induced by G on \overline{X} .

Lemma 2. *Let*

$$X = X_1 \cup X_2 \cup \cdots \cup X_m$$

be the decomposition of X into pairwise orthogonal subsets such that none of X_i 's are decomposable any further. Then the kernel of the canonical homomorphism $G \rightarrow \overline{G}$ is the elementary abelian subgroup of order 2^m consisting of the multiplications by -1 on some of X_i 's.

Proof. Let K denote the kernel of the canonical homomorphism $G \rightarrow \overline{G}$. Since

$$\mathbb{R}^n = \langle X_1 \rangle \perp \langle X_2 \rangle \perp \cdots \perp \langle X_m \rangle ,$$

it is clear that the multiplication by -1 on some of X_i 's is in $O(n, \mathbb{R})$, and hence in K .

Conversely, suppose $g \in K$. If $x, y \in X$, $(x, y) \neq 0$, $g(x) = x$ and $g(y) = -y$, then $(x, y) = (g(x), g(y)) = (x, -y) = -(x, y)$. This is a contradiction. Therefore, g is either 1 or -1 on each of the subspaces $\langle X_i \rangle$. \square

Lemma 3. *Let $\mathcal{F} = \{f_1, \dots, f_n\}$ be a k -frame of an integral lattice L , and let $C = \pi_{\mathcal{F}}(L)$. Then every automorphism of C is induced by an automorphism*

of L which is represented by a monomial $(\pm 1, 0)$ -matrix with respect to \mathcal{F} . In other words, there is a surjective homomorphism from $\text{Stab}_{\text{Aut}(L)}(\overline{\mathcal{F}})$ to $\text{Aut}(C)$. Moreover, this homomorphism is injective if and only if $k > 2$.

Proof. Let F denote the $n \times n$ matrix whose row vectors consist of the elements of \mathcal{F} . Then $A_k(C) \cdot \frac{1}{\sqrt{k}}F = L$. Let $P \in \text{Stab}_{\text{Aut}(L)}(\overline{\mathcal{F}})$. Then, as $\frac{1}{k}FPF^T$ is a monomial $(\pm 1, 0)$ -matrix, we can define a mapping ϕ from $\text{Stab}_{\text{Aut}(L)}(\overline{\mathcal{F}})$ to $\text{Aut}(C)$ by $P \mapsto \frac{1}{k}FPF^T \bmod k$. Clearly this mapping ϕ is a homomorphism. We claim that ϕ is surjective. Indeed, let Q be a monomial $(\pm 1, 0)$ -matrix representing an automorphism of C , regarded as a matrix with integer entries. Since $A_k(C) \cdot Q = A_k(C \cdot (Q \bmod k))$, we obtain $\frac{1}{k}F^TQF \in \text{Aut}(L)$. Setting $P = \frac{1}{k}F^TQF$, we obtain $\phi(P) = Q \bmod k$. Therefore, ϕ is surjective.

The kernel of ϕ is the set of elements P in $\text{Stab}_{\text{Aut}(L)}(\overline{\mathcal{F}})$ such that the monomial matrix $\frac{1}{k}FPF^T$ reduces to the identity matrix modulo k . If $k > 2$, then the only such monomial matrix is the identity matrix itself. If $k = 2$, then $-I$ is in the kernel, hence ϕ is not injective. \square

Lemma 4. Suppose $k > 2$. Let $\mathcal{F} = \{f_1, \dots, f_n\}$ be a k -frame of an integral lattice L . Then the automorphism group $\text{Aut}(C)$ of the code $C = \pi_{\mathcal{F}}(L)$ has order

$$2^m \cdot \# \text{Stab}_{\overline{\text{Aut}(L)}}(\overline{\mathcal{F}}),$$

where m is the number of components in the orthogonal decomposition of the set X of vectors of norm k in L , and $\overline{\text{Aut}(L)}$ is the permutation group on \overline{X} induced by $\text{Aut}(L)$.

Proof. By Lemma 3, $\text{Aut}(C)$ is isomorphic to $\text{Stab}_{\text{Aut}(L)}(\overline{\mathcal{F}})$, and the latter contains the kernel of the action of $\text{Aut}(L)$ on \overline{X} . The result then follows from Lemma 2. \square

As we described in Subsection 2.2, it is the permutation group $\overline{\text{Aut}(L)}$ on \overline{X} that is processed by computer to compute the stabilizer $\text{Stab}_{\overline{\text{Aut}(L)}}(\overline{\mathcal{F}})$. Since the number of components in the orthogonal decomposition of X can be easily computed separately, Lemma 4 then allows us to find the order of $\text{Aut}(C)$ from that of $\text{Stab}_{\overline{\text{Aut}(L)}}(\overline{\mathcal{F}})$.

2.4 Mass Formulas

Let $N_k(n)$ and $N_{k,\text{II}}(n)$ be the numbers of distinct self-dual \mathbb{Z}_k -codes and Type II \mathbb{Z}_k -codes of length n , respectively. The numbers $N_4(n)$ and $N_{4,\text{II}}(n)$ are given in [10]. The number $N_k(n)$ is given in [1] for $k = p^2$ when p is an odd prime and [14] for $k = p^3$ when p is a prime. The number $N_{8,\text{II}}(n)$ is also given in [3].

Suppose that p is an odd prime. Since any self-dual (resp. Type II) \mathbb{Z}_{2p} -code can be obtained from a pair of a binary self-dual (resp. Type II) code and a self-dual IF_p -code [8], it is easy to see that

$$N_{2p}(n) = N_2(n) \cdot N_p(n) \text{ and } N_{2p,\text{II}}(n) = N_{2,\text{II}}(n) \cdot N_p(n)$$

(see [18] for the number $N_p(n)$).

Mass formulas are useful when attempting to complete the classification of self-dual codes. Let $\mathcal{C}_k(n)$ and $\mathcal{C}_{k,\text{II}}(n)$ be the sets of inequivalent self-dual \mathbb{Z}_k -codes and Type II \mathbb{Z}_k -codes of length n , respectively. In order to check that our classification is complete, the following mass formulas are used

$$\sum_{C \in \mathcal{C}_k(n)} \frac{2^n \cdot n!}{\# \text{Aut}(C)} = N_k(n) \quad (1)$$

and

$$\sum_{C \in \mathcal{C}_{k,\text{II}}(n)} \frac{2^n \cdot n!}{\# \text{Aut}(C)} = N_{k,\text{II}}(n) . \quad (2)$$

In all of the classification results below, we have computed the sum in (1) or (2) where $\# \text{Aut}(C)$ are calculated by Lemma 4, and checked against the known formula for $N_k(n)$ or $N_{k,\text{II}}(n)$.

3 Self-dual \mathbb{Z}_4 -Codes

A self-dual \mathbb{Z}_4 -code exists for every length. The classification of self-dual \mathbb{Z}_4 -codes was done for lengths $n \leq 9$ in [5] and for lengths $n = 10, \dots, 15$ in [9]. At length 16, the classification of Type II codes was done in [16]. In this section, we

Table 3. Number of Type I \mathbb{Z}_4 -codes of length 16

$L \setminus l$	0	1	2	3	4	5	6	7	8	Total
D_8^2	0	0	2	8	31	86	160	168	85	540
\mathbb{Z}^{16}	1	1	2	4	7	8	8	3	1	35
$E_8 \oplus \mathbb{Z}^8$	0	1	3	9	22	34	35	20	5	129
$D_{12} \oplus \mathbb{Z}^4$	0	1	4	13	34	68	82	53	15	270
$E_7^2 \oplus \mathbb{Z}^2$	0	0	1	6	23	57	94	74	25	280
$A_{15} \oplus \mathbb{Z}$	0	0	0	1	5	16	34	43	19	118

Table 4. Number of self-dual \mathbb{Z}_4 -codes of length 17

$L \setminus l$	0	1	2	3	4	5	6	7	8	Total
$A_{11}E_6$	0	0	0	1	5	20	45	48	17	136
$E_8^2 \oplus \mathbb{Z}$	0	0	1	2	6	12	17	18	8	64
$D_{16} \oplus \mathbb{Z}$	0	1	1	3	7	13	19	16	9	69
$D_8^2 \oplus \mathbb{Z}$	0	0	2	9	36	107	206	220	107	687
\mathbb{Z}^{17}	1	1	2	4	7	8	9	4	1	37
$E_8 \oplus \mathbb{Z}^9$	0	1	3	9	22	37	41	25	6	144
$D_{12} \oplus \mathbb{Z}^5$	0	1	4	13	36	76	100	69	19	318
$E_7^2 \oplus \mathbb{Z}^3$	0	0	1	6	25	68	119	102	33	354
$A_{15} \oplus \mathbb{Z}^2$	0	0	0	1	6	21	50	64	29	171

Table 5. Number of self-dual \mathbb{Z}_4 -codes of length 18

$L \setminus l$	0	1	2	3	4	5	6	7	8	Total
$A_{17}A_1$	0	0	0	0	1	3	11	19	13	47
$D_{10}E_7A_1$	0	0	1	5	22	74	177	250	149	678
D_6^3	0	0	1	5	31	142	453	805	599	2036
A_9^2	0	0	0	0	2	15	80	197	198	492
$A_{11}E_6 \oplus \mathbb{Z}$	0	0	0	1	8	43	140	234	143	569
$E_8^2 \oplus \mathbb{Z}^2$	0	0	1	3	10	23	40	45	28	150
$D_{16} \oplus \mathbb{Z}^2$	0	1	2	5	12	25	41	43	24	153
$D_8^2 \oplus \mathbb{Z}^2$	0	0	2	12	58	207	511	723	452	1965
Z^{18}	1	1	2	4	7	9	12	9	3	48
$E_8 \oplus \mathbb{Z}^{10}$	0	1	3	9	24	46	65	56	20	224
$D_{12} \oplus \mathbb{Z}^6$	0	1	4	15	44	106	176	176	72	594
$E_7^2 \oplus \mathbb{Z}^4$	0	0	1	7	33	106	231	278	138	794
$A_{15} \oplus \mathbb{Z}^3$	0	0	0	1	6	25	69	107	60	268

Table 6. Number of self-dual \mathbb{Z}_4 -codes of length 19

$L \setminus l$	0	1	2	3	4	5	6	7	8	9	Total
$E_6^3O_1$	0	0	0	1	5	25	84	166	142	0	423
$A_{11}D_7O_1$	0	0	0	1	6	36	139	308	287	0	777
$A_7^2D_5$	0	0	0	1	11	79	386	1083	1191	0	2751
$A_{17}A_1 \oplus \mathbb{Z}$	0	0	0	0	1	4	15	29	23	0	72
$D_{10}E_7A_1 \oplus \mathbb{Z}$	0	0	1	6	29	102	259	397	263	0	1057
$D_6^3 \oplus \mathbb{Z}$	0	0	1	6	40	206	732	1467	1220	0	3672
$A_9^2 \oplus \mathbb{Z}$	0	0	0	0	2	18	115	343	395	0	873
$A_{11}E_6 \oplus \mathbb{Z}^2$	0	0	0	1	10	65	253	524	404	0	1257
$E_8^2 \oplus \mathbb{Z}^3$	0	0	1	3	11	26	46	55	35	0	177
$D_{16} \oplus \mathbb{Z}^3$	0	1	2	6	14	29	50	55	35	0	192
$D_8^2 \oplus \mathbb{Z}^3$	0	0	2	12	63	240	638	1005	690	0	2650
Z^{19}	1	1	2	4	7	9	13	11	5	0	53
$E_8 \oplus \mathbb{Z}^{11}$	0	1	3	9	24	48	73	74	34	0	266
$D_{12} \oplus \mathbb{Z}^7$	0	1	4	15	46	115	207	239	122	0	749
$E_7^2 \oplus \mathbb{Z}^5$	0	0	1	7	34	116	277	378	220	0	1033
$A_{15} \oplus \mathbb{Z}^4$	0	0	0	1	7	30	90	154	107	0	389

give a classification of Type I codes of length 16 and self-dual codes of lengths 17, 18 and 19.

In Tables 3, 4, 5 and 6, we list the numbers of inequivalent Type I \mathbb{Z}_4 -codes C of length 16, inequivalent self-dual \mathbb{Z}_4 -codes C of lengths 17, 18 and 19, respectively, with $A_4(C) \simeq L$ for each unimodular lattice L and for each dimension l of the residue codes $C^{(1)} = \{c \bmod 2 \mid c \in C\}$. Table 3 together with the result in [16] gives the total number of inequivalent self-dual \mathbb{Z}_4 -codes of length 16.

Proposition 1. *There are 1,372 inequivalent Type I \mathbb{Z}_4 -codes of length 16. There are 1,980 inequivalent self-dual \mathbb{Z}_4 -codes of length 17. There are 8,018 inequivalent self-dual \mathbb{Z}_4 -codes of length 18. There are 16,391 inequivalent self-dual \mathbb{Z}_4 -codes of length 19.*

4 Self-dual \mathbb{Z}_8 -Codes

A self-dual \mathbb{Z}_8 -code of length n exists if and only if n is even. The classification of self-dual \mathbb{Z}_8 -codes was done in [7] for lengths $n \leq 8$. As pointed out in [14], the classification of self-dual codes of length 6 given in [7] misses some codes. It turns out that the classification of self-dual codes of length 8 in [7] is also incomplete. In this section, we give a revised list of self-dual \mathbb{Z}_8 -codes of lengths 6 and 8. We also extend the classification to length 12.

4.1 Length 6

Let $C_{8,6,1}$, $C_{8,6,2}$ and $C_{8,6,3}$ be the \mathbb{Z}_8 -codes of length 6 with generator matrices

$$\begin{pmatrix} 200002 \\ 020020 \\ 002200 \\ 000400 \\ 000040 \\ 000004 \end{pmatrix}, \begin{pmatrix} 101776 \\ 011653 \\ 002024 \\ 000444 \end{pmatrix}, \begin{pmatrix} 110211 \\ 020024 \\ 002204 \\ 000400 \\ 000044 \end{pmatrix},$$

respectively. These codes are self-dual codes which have automorphism groups of orders $2 \cdot 1536$, $2 \cdot 24$ and $2 \cdot 64$, respectively. These three codes make a complete list of inequivalent self-dual \mathbb{Z}_8 -codes of length 6. Two codes $C_{8,6,2}$ and $C_{8,6,3}$ are missing in the classification given in [7].

4.2 Length 8

There are 9 inequivalent Type II \mathbb{Z}_8 -codes and there are 20 inequivalent Type I \mathbb{Z}_8 -codes of length 8. The orders of their automorphism groups are listed in Tables 7 and 8, respectively.

Let D_i denote the code with generator matrix $G_{8,i}$ in [7]. We note that the codes D_{12} and D_{15} are equivalent, since D_{15} is obtained from D_{12} by the

Table 7. Type II \mathbb{Z}_8 -codes of length 8

i	# Aut($C_{8,8,i}$)	i	# Aut($C_{8,8,i}$)	i	# Aut($C_{8,8,i}$)
1	$2 \cdot 168$	4	$2 \cdot 24$	7	$2 \cdot 1536$
2	$2 \cdot 24$	5	$2 \cdot 96$	8	$2 \cdot 172032$
3	$2 \cdot 10752$	6	$2 \cdot 192$	9	$2 \cdot 42$

Table 8. Type I \mathbb{Z}_8 -codes of length 8

i	# Aut($D_{8,8,i}$)						
1	$2 \cdot 6$	6	$2 \cdot 8$	11	$2 \cdot 96$	16	$2 \cdot 6$
2	$2 \cdot 32$	7	$2 \cdot 64$	12	$2 \cdot 6$	17	$2 \cdot 49152$
3	$2 \cdot 1536$	8	$2 \cdot 24$	13	$2 \cdot 32$	18	$2 \cdot 512$
4	$2 \cdot 192$	9	$2 \cdot 8$	14	$2 \cdot 1536$	19	$2 \cdot 6$
5	$2 \cdot 21$	10	$2 \cdot 192$	15	$2 \cdot 10752$	20	$2 \cdot 3$

permutation $(1, 8, 2, 6, 7, 3)$ and by changing the signs of the 6th and 8th coordinates. Moreover, we have the following pairs of equivalent codes:

$$\begin{aligned} & (D_1, C_{8,8,8}), (D_2, D_{8,8,17}), (D_3, C_{8,8,7}), (D_4, D_{8,8,14}), (D_5, D_{8,8,15}) , \\ & (D_6, C_{8,8,3}), (D_7, D_{8,8,3}), (D_8, C_{8,8,6}), (D_9, D_{8,8,11}), (D_{10}, D_{8,8,1}) , \\ & (D_{11}, D_{8,8,6}), (D_{12}, D_{8,8,5}), (D_{13}, D_{8,8,16}), (D_{14}, D_{8,8,20}), (D_{16}, D_{8,8,9}) , \\ & (D_{17}, D_{8,8,8}), (D_{18}, C_{8,8,2}), (D_{19}, C_{8,8,1}) , \end{aligned}$$

where (X, Y) means that X, Y are equivalent. Hence $C_{8,8,i}$ ($i = 4, 5, 9$) and $D_{8,8,i}$ ($i = 2, 4, 7, 10, 12, 13, 18, 19$) are missing in the classification given in [7].

The codes $C_{8,8,i}$ ($i = 4, 5, 9$) have the following generator matrices:

$$\begin{pmatrix} 10003633 \\ 01013476 \\ 00103716 \\ 00022646 \\ 00004040 \end{pmatrix}, \begin{pmatrix} 10003277 \\ 01011234 \\ 00202264 \\ 00022202 \\ 00004040 \\ 00000400 \end{pmatrix}, \begin{pmatrix} 10010125 \\ 01002151 \\ 00112574 \\ 00022426 \\ 00004000 \end{pmatrix},$$

respectively. The codes $D_{8,8,i}$ ($i = 2, 4, 7, 10, 18, 19$) have the following generator matrices:

$$\begin{aligned} & \begin{pmatrix} 10103214 \\ 01010127 \\ 00202000 \\ 00020244 \\ 00004040 \\ 00000404 \end{pmatrix}, \begin{pmatrix} 10010172 \\ 01103025 \\ 00200262 \\ 00022202 \\ 00004004 \\ 00000440 \end{pmatrix}, \begin{pmatrix} 10102330 \\ 01102103 \\ 00200204 \\ 00022044 \\ 00004000 \\ 00000444 \end{pmatrix}, \begin{pmatrix} 10010327 \\ 01010275 \\ 00202000 \\ 00020042 \\ 00004000 \\ 00000444 \end{pmatrix}, \\ & \begin{pmatrix} 10012103 \\ 02002004 \\ 00200020 \\ 00020204 \\ 00004000 \\ 00000404 \\ 00000040 \end{pmatrix}, \begin{pmatrix} 10010653 \\ 01010572 \\ 00111670 \\ 00020024 \\ 00004444 \end{pmatrix}, \end{aligned}$$

respectively. Generator matrices of codes $D_{8,8,12}, D_{8,8,13}$ can be obtained from those of $C_{8,8,9}, C_{8,8,5}$, respectively, since $D_{8,8,12} = C_{8,8,9} \cdot M$ and $D_{8,8,13} = C_{8,8,5} \cdot M$, where M is the diagonal matrix $\text{diag}(3, 1, 1, 1, 1, 1, 1, 1)$.

4.3 Lengths 10 and 12

For lengths 10 and 12, we give the number N of inequivalent self-dual codes C with $A_8(C) \simeq L$ in Table 9 for each unimodular lattice L .

Table 9. Numbers of self-dual $\mathbb{Z}Z_8$ -codes of lengths 10, 12

n	10		12	
	\mathbb{Z}^{10}	$E_8 \oplus \mathbb{Z}^2$	D_{12}	\mathbb{Z}^{12}
L	$E_8 \oplus \mathbb{Z}^2$	D_{12}	$E_8 \oplus \mathbb{Z}^4$	
N	50	42	250	194
				571

Proposition 2. *There are three inequivalent self-dual $\mathbb{Z}Z_8$ -codes of length 6. There are 20 (resp. 9) inequivalent Type I (resp. Type II) $\mathbb{Z}Z_8$ -codes of length 8. There are 92 inequivalent self-dual $\mathbb{Z}Z_8$ -codes of length 10. There are 1,015 inequivalent self-dual $\mathbb{Z}Z_8$ -codes of length 12.*

4.4 A Weaker Equivalence

In this paper, we employ monomial $(\pm 1, 0)$ -matrices as the definition of equivalence of codes. For self-dual $\mathbb{Z}Z_8$ -codes, one could use monomial $(\pm 1, \pm 3, 0)$ -matrices to define a weaker equivalence. We note that there are codes C, C' which are equivalent in this weak sense, such that $A_8(C) \not\simeq A_8(C')$. Indeed, C' could be Type I even if C is Type II.

For each of lengths 2, 4 and 6, the equivalence classes are the same under both definitions. For length 8, there are 15 classes under the weak equivalence. More specifically, the following codes are equivalent in the weak sense:

$$(D_{8,8,1}, D_{8,8,6}, C_{8,8,4}), (D_{8,8,4}, D_{8,8,11}, C_{8,8,6}) , \\ (D_{8,8,i} \ (i = 5, 8, 9, 16, 20), C_{8,8,1}, C_{8,8,2}), (D_{8,8,12}, C_{8,8,9}), (D_{8,8,13}, C_{8,8,5}) , \\ (D_{8,8,14}, C_{8,8,7}), (D_{8,8,15}, C_{8,8,3}) .$$

For length 10, we have verified that there are 37 classes under the weak equivalence. Due to the computational complexity, we were unable to find equivalence classes for length 12. However, from the number of different weight distributions, it follows that there are at least 198 equivalence classes.

5 Self-dual $\mathbb{Z}Z_k$ -Codes ($k = 6, 9, 10$)

In this section, we give a summary of our classification of self-dual $\mathbb{Z}Z_k$ -codes ($k = 6, 9, 10$) of lengths which are listed in the last column of Table 1.

A self-dual \mathbb{Z}_9 -code exists for every length. A self-dual \mathbb{Z}_6 -code of length n exists if and only if n is divisible by four and a self-dual \mathbb{Z}_{10} -code of length n exists if and only if n is even. The classification of self-dual \mathbb{Z}_6 -codes of length 4 was done in [8] incorrectly and there is a unique such code up to equivalence [15, Example 6.5]. We remark that Kitazume and Ooi [13] classified Type II \mathbb{Z}_6 -codes of length 8, and there are two such codes up to equivalence. This result, together with Table 10 gives the total number of inequivalent self-dual \mathbb{Z}_6 -codes of lengths 8 and 12. Tables 11 and 12 give the numbers N of inequivalent self-dual \mathbb{Z}_9 -codes C of lengths 9, 10, 11, 12 with $A_9(C) \simeq L$, and of self-dual \mathbb{Z}_{10} -codes C of lengths up to 10 with $A_{10}(C) \simeq L$, respectively, for each unimodular lattice L .

Table 10. Numbers of self-dual \mathbb{Z}_6 -codes of lengths 8, 12

n	8		12		
	L	\mathbb{Z}^8	D_{12}	\mathbb{Z}^{12}	$E_8 \oplus \mathbb{Z}^4$
N	3 (Type I)		25	18	35

Table 11. Numbers of self-dual \mathbb{Z}_9 -codes of lengths 9, 10, 11, 12

n	9		10		11		12				
	L	\mathbb{Z}^9	$E_8 \oplus \mathbb{Z}$	\mathbb{Z}^{10}	$E_8 \oplus \mathbb{Z}^2$	\mathbb{Z}^{11}	$E_8 \oplus \mathbb{Z}^3$	D_{12}	\mathbb{Z}^{12}	$E_8 \oplus \mathbb{Z}^4$	
N	28	7		56	27		118	153	501	425	1564

Table 12. Numbers of self-dual \mathbb{Z}_{10} -codes of lengths 2, 4, 6, 8, 10

n	2		4		6		8		10	
	L	\mathbb{Z}^2	\mathbb{Z}^4	\mathbb{Z}^6	\mathbb{Z}^8	E_8	\mathbb{Z}^{10}	$E_8 \oplus \mathbb{Z}^2$		
N	1	2	5	16 (Type I)	11 (Type II)		96	79		

We give generator matrices of the three Type I \mathbb{Z}_6 -codes of length 8:

$$\begin{pmatrix} 10000432 \\ 01001020 \\ 00100502 \\ 00012023 \end{pmatrix}, \begin{pmatrix} 10004320 \\ 01004043 \\ 00100434 \\ 00013402 \end{pmatrix}, \begin{pmatrix} 10000504 \\ 01002010 \\ 00101040 \\ 00010401 \end{pmatrix}.$$

We give generator matrices of the self-dual \mathbb{Z}_{10} -codes of lengths 2, 4, 6:

- $n = 2$: $(\begin{smallmatrix} 1 & 3 \end{smallmatrix})$.
- $n = 4$: $\begin{pmatrix} 1052 \\ 0125 \end{pmatrix}, \begin{pmatrix} 1030 \\ 0103 \end{pmatrix}$.
- $n = 6$: $\begin{pmatrix} 100520 \\ 010205 \\ 001058 \end{pmatrix}, \begin{pmatrix} 100726 \\ 010632 \\ 001243 \end{pmatrix}, \begin{pmatrix} 100182 \\ 010834 \\ 001243 \end{pmatrix}, \begin{pmatrix} 100300 \\ 010030 \\ 001003 \end{pmatrix}, \begin{pmatrix} 100508 \\ 010070 \\ 001805 \end{pmatrix}$.

6 Largest Minimum Weights

For $k = 4, 6, 8, 9$ and 10, let $d_{k,H}, d_{k,L}$ and $d_{k,E}$ be the largest minimum Hamming, Lee and Euclidean weights among self-dual \mathbb{Z}_k -codes of length n , respectively, and let $N_{k,H}, N_{k,L}$ and $N_{k,E}$ be the numbers of inequivalent self-dual \mathbb{Z}_k -codes with the minimum Hamming, Lee and Euclidean weights $d_{k,H}, d_{k,L}$ and $d_{k,E}$, respectively. In Table 13, we list the total number # of codes, $d_{k,H}, d_{k,L}$, $d_{k,E}, N_{k,H}, N_{k,L}$ and $N_{k,E}$ ($k = 4, 6, 8, 9$ and 10) for lengths which are listed in

Table 13. Largest minimum weights of self-dual \mathbb{Z}_k -codes ($k = 4, 6, 8, 9, 10$)

	Length	#	$d_{k,H}$	$N_{k,H}$	$d_{k,L}$	$N_{k,L}$	$d_{k,E}$	$N_{k,E}$
\mathbb{Z}_4	16 (Type I)	1,372	4	215	8	19	8	540
	17	1,980	4	62	6	29	8	136
	18	8,018	4	66	8	7	8	3253
	19	16,391	3	81	6	61	8	3951
\mathbb{Z}_6	8 (Type I)	3	2	3	6	1	6	3
	12	78	4	10	8	5	12	25
\mathbb{Z}_8	6	3	2	1	6	1	8	3
	8 (Type I)	20	4	5	8	1	8	20
	8 (Type II)	9	4	2	8	5	16	9
	10	92	2	25	8	4	8	92
	12	1,015	2	388	8	172	16	250
\mathbb{Z}_9	9	35	3	7	9	1	9	35
	10	83	4	7	9	1	9	83
	11	271	5	13	9	1	9	271
	12	2,490	6	259	10	72	18	501
\mathbb{Z}_{10}	2	1	2	1	4	1	10	1
	4	2	2	2	6	1	10	2
	6	5	2	5	6	3	10	5
	8 (Type I)	16	2	16	8	3	10	16
	8 (Type II)	11	4	6	8	6	20	11
	10	175	2	175	10	4	10	175

the last column of Table 1. We remark that $d_{4,H}$, $d_{4,L}$, $d_{4,E}$ were determined in [17] for lengths $n \leq 24$, and $N_{4,H}$ for lengths 17, 18 and $N_{4,L}$ for length 18 were already determined in [17] without full classification.

References

1. Balmaceda, J.M.P., Betty, R.A.L., Nemenzo, F.R.: Mass formula for self-dual codes over \mathbf{Z}_{p^2} . *Discrete Math.* 308, 2984–3002 (2008)
2. Bannai, E., Dougherty, S.T., Harada, M., Oura, M.: Type II codes, even unimodular lattices, and invariant rings. *IEEE Trans. Inform. Theory* 45, 1194–1205 (1999)
3. Betsumiya, K., Betty, R.A.L., Munemasa, A.: Mass formula for even codes over \mathbf{Z}_8 . In: Parker, M.G. (ed.) IMACC 2009. LNCS, vol. 5921, pp. 65–77. Springer, Heidelberg (2009)
4. Bosma, W., Cannon, J.: Handbook of Magma Functions. Department of Mathematics, University of Sydney, <http://magma.maths.usyd.edu.au/magma/>
5. Conway, J.H., Sloane, N.J.A.: Self-dual codes over the integers modulo 4. *J. Combin. Theory Ser. A* 62, 30–45 (1993)
6. Conway, J.H., Sloane, N.J.A.: Sphere Packing, Lattices and Groups, 3rd edn. Springer, New York (1999)
7. Dougherty, S.T., Gulliver, T.A., Wong, J.: Self-dual codes over \mathbb{Z}_8 and \mathbb{Z}_9 . *Des. Codes Cryptogr.* 41, 235–249 (2006)
8. Dougherty, S.T., Harada, M., Solé, P.: Self-dual codes over rings and the Chinese remainder theorem. *Hokkaido Math. J.* 28, 253–283 (1999)
9. Fields, J., Gaborit, P., Leon, J.S., Pless, V.: All self-dual \mathbb{Z}_4 codes of length 15 or less are known. *IEEE Trans. Inform. Theory* 44, 311–322 (1998)
10. Gaborit, P.: Mass formulas for self-dual codes over \mathbb{Z}_4 and $F_q + uF_q$ rings. *IEEE Trans. Inform. Theory* 42, 1222–1228 (1996)
11. Harada, M., Munemasa, A.: Database of Self-Dual Codes, <http://www.math.is.tohoku.ac.jp/~munemasa/selfdualcodes.htm>
12. Harada, M., Munemasa, A., Venkov, B.: Classification of ternary extremal self-dual codes of length 28. *Math. Comput.* 78, 1787–1796 (2009)
13. Kitazume, M., Ooi, T.: Classification of type II \mathbb{Z}_6 -codes of length 8. *AKCE Int. J. Graphs Comb.* 1, 35–40 (2004)
14. Nagata, K., Nemenzo, F., Wada, H.: The number of self-dual codes over \mathbf{Z}_{p^3} . *Des. Codes Cryptogr.* 50, 291–303 (2009)
15. Park, Y.H.: Modular independence and generator matrices for codes over \mathbb{Z}_m . *Des. Codes Cryptogr.* 50, 147–162 (2009)
16. Pless, V., Leon, J.S., Fields, J.: All \mathbb{Z}_4 codes of Type II and length 16 are known. *J. Combin. Theory Ser. A* 78, 32–50 (1997)
17. Rains, E.: Optimal self-dual codes over \mathbb{Z}_4 . *Discrete Math.* 203, 215–228 (1999)
18. Rains, E., Sloane, N.J.A.: Self-Dual Codes: Handbook of Coding Theory. In: Pless, V.S., Huffman, W.C. (eds.), pp. 177–294. Elsevier, Amsterdam (1998)

On Linear Codes from Maximal Curves

Stefania Fanali

Dipartimento di Matematica e Informatica - Università degli Studi della Basilicata
Viale dell'Ateneo Lucano 10 - 85100 Potenza - Italy
stefania.fanali@dmi.unipg.it

Abstract. Some linear codes associated to maximal algebraic curves via Feng-Rao construction are investigated. In several case, these codes have better minimum distance with respect to the previously known linear codes with same length and dimension.

Keywords: linear codes, algebraic geometric codes, maximal curves.

1 Introduction

The idea of constructing linear codes from algebraic curves defined over a finite field \mathbb{F}_q goes back to Goppa [10]. These codes are usually called Algebraic Geometric Codes, AG codes for short. Typically, AG codes with good parameters arise from curves with a large number N of \mathbb{F}_q -rational points with respect to their genus g . In fact, for an $[N, k]_q$ AG code code associated to a curve of genus g , the sum of its transmission rate plus its relative minimum distance is at least $1 - \frac{g-1}{N}$. An upper bound on N is given by the Hasse-Weil estimate $N \leq q + 1 + 2g\sqrt{q}$.

In 1995 Feng and Rao [3] introduced the so-called Improved AG Codes, see Section 2.2. The parameters of these codes depend on the pattern of the Weierstrass semigroup at the points of the underlying curve, and it has emerged that they can be significantly better than those of the ordinary AG codes, see [3] and [12, Section 4.3].

The aim of this paper is to investigate the parameters of the Improved AG Codes associated to some classes of maximal curves, that is, curves for which the Hasse-Weil upper bound is attained. Since maximal curves with positive genus exist only for square q , henceforth we assume that $q = q_0^2$. The main achievement of the paper is the discovery of several linear codes that apparently have better parameters with respect to the previously known ones, see Appendix. Our method is based on the explicit description of the Weierstrass semigroup at some \mathbb{F}_q -rational points of the curves under investigation.

The maximal curves that will be considered are the following.

- (A) Curves with equation $X^{2m} + X^m + Y^{q_0+1} = 0$, where $m > 2$ is a divisor of $q_0 + 1$, and $\Delta = \frac{q_0+1}{m} > 3$ is a prime [7].
- (B) Curves with equation $X^{2i+2} + X^{2i} + Y^{q_0+1} = 0$, where $\Delta = \frac{q_0+1}{2} > 3$ is a prime, and $1 \leq i \leq \Delta - 2$ [8].

- (C) Quotient curves of the Hermitian curve $Y^{q_0+1} = X^{q_0} + X$ with respect to the additive subgroups of $H = \{c \in \mathbb{F}_q \mid c^{q_0} + c = 0\}$ [5].
- (D) Curves with equation $Y^m = X^{q_0} + X$, where m is a proper divisor of $q_0 + 1$ [1].
- (E) Curves with equation $Y^{\frac{q-1}{m}} = X(X + 1)^{q_0-1}$, where m is a divisor of $q - 1$ [5].

2 Notation and Preliminaries

2.1 Curves

Throughout the paper, by a curve we mean a projective, geometrically irreducible, non-singular algebraic curve defined over a finite field. Let q_0 be a prime power, $q = q_0^2$, and let \mathcal{X} be a curve defined over the finite field \mathbb{F}_q of order q . Let g be the genus of \mathcal{X} . Henceforth, the following notation is used:

- $\mathcal{X}(\mathbb{F}_q)$ (resp. $\mathbb{F}_q(\mathcal{X})$) denotes the set of \mathbb{F}_q -rational points (resp. the field of \mathbb{F}_q -rational functions) of \mathcal{X} .
- \mathcal{H} is the Hermitian curve over \mathbb{F}_q with affine equation

$$Y^{q_0+1} = X^{q_0} + X. \quad (1)$$

- For $f \in \mathbb{F}_q(\mathcal{X})$, (f) (resp. $(f)_\infty$) denotes the divisor (resp. the pole divisor) of f .
- Let P be a point of \mathcal{X} . Then ord_P (resp. $H(P)$) stands for the valuation (resp. for the Weierstrass non-gap semigroup) associated to P . The i th non-gap at P is denoted as $m_i(P)$.

2.2 One-Point AG Codes and Improved AG Codes

Let \mathcal{X} be a curve, let P_1, P_2, \dots, P_n be \mathbb{F}_q -rational points of \mathcal{X} , and let D be the divisor $P_1 + P_2 + \dots + P_n$. Furthermore, let G be some other divisor that has support disjoint from D . The AG code $C(D, G)$ of length n over \mathbb{F}_q is the image of the linear map $\alpha : L(G) \rightarrow \mathbb{F}_q^n$ defined by $\alpha(f) = (f(P_1), f(P_2), \dots, f(P_n))$. If n is bigger than $deg(G)$, then α is an embedding, and the dimension k of $C(D, G)$ is equal to $\ell(G)$. The Riemann-Roch theorem makes it possible to estimate the parameters of $C(D, G)$. In particular, if $2g - 2 < deg(G) < n$, then $C(D, G)$ has dimension $k = deg(G) - g + 1$ and minimum distance $d \geq n - deg(G)$, see e.g. [12, Theorem 2.65]. A generator matrix M of $C(D, G)$ is

$$M = \begin{pmatrix} f_1(P_1) & \dots & f_1(P_n) \\ \vdots & \dots & \vdots \\ f_k(P_1) & \dots & f_k(P_n) \end{pmatrix},$$

where f_1, f_2, \dots, f_k is an \mathbb{F}_q -basis of $L(G)$. The dual code $C^\perp(D, G)$ of $C(D, G)$ is an AG code with dimension $n - k$ and minimum distance greater than or

equal to $\deg(G) - 2g + 2$. When $G = \gamma P$ for an \mathbb{F}_q -rational P point of \mathcal{X} , and a positive integer γ , AG codes $C(D, G)$ and $C^\perp(D, G)$ are referred to as one-point AG codes. We recall some results on the minimum distance of one-point AG codes. By [9, Theorem 3], we can assume that γ is a non-gap at P . Let

$$H(P) = \{\rho_1 = 0 < \rho_2 < \dots\},$$

and set $\rho_0 = 0$. Let f_ℓ be a rational function such that $\text{div}_\infty(f_\ell) = \rho_\ell P$, for any $\ell \geq 1$. Let $D = P_1 + P_2 + \dots + P_n$. Let also

$$h_\ell = (f_\ell(P_1), f_\ell(P_2), \dots, f_\ell(P_n)) \in \mathbb{F}_q^n. \quad (2)$$

Set

$$\nu_\ell := \#\{(i, j) \in \mathbb{N}^2 : \rho_i + \rho_j = \rho_{\ell+1}\}$$

for any $\ell \geq 0$. Denote with $C_\ell(P)$ the dual of the AG code $C(D, G)$, where $D = P_1 + P_2 + \dots + P_n$, and $G = \rho_\ell P$.

Definition 1. Let d be an integer greater than 1. The Improved AG code $\tilde{C}_d(P)$ is the code

$$\tilde{C}_d(P) := \{x \in \mathbb{F}_q^n : \langle x, h_{i+1} \rangle = 0 \text{ for all } i \text{ such that } \nu_i < d\},$$

see [12, Def. 4.22].

Theorem 1 (Proposition 4.23 in [12]). Let

$$r_d := \#\{i \geq 0 : \nu_i < d\}.$$

Then $\tilde{C}_d(P)$ is an $[n, k, d']$ -code, where $k \geq n - r_d$, and $d' \geq d$.

2.3 Maximal Curves

A curve \mathcal{X} is called \mathbb{F}_q -maximal if the number of its \mathbb{F}_q -rational points attains the Hasse-Weil upper bound, that is,

$$\#\mathcal{X}(\mathbb{F}_q) = q + 1 + 2gq_0,$$

where g is the genus of \mathcal{X} .

We recall a result on Weierstrass semigroups of maximal curves that will be used in the sequel of the paper.

Proposition 1 ([4]). Let \mathcal{X} be a maximal curve over \mathbb{F}_q . If $P \in \mathcal{X}(\mathbb{F}_q)$, then both q_0 and $q_0 + 1$ belongs to $H(P)$.

2.4 Numerical Semigroups

A numerical semigroup is a subset of the set \mathbb{N} of nonnegative integers that is closed under addition, contains 0 and whose complement in \mathbb{N} is finite. A gap of a numerical semigroup Θ is a nonnegative integer not belonging to Θ . The genus of a numerical semigroup is the number of its gaps.

For a point P of a curve \mathcal{X} , the Weierstrass semigroup $H(P)$ is a numerical semigroup whose genus coincides with the genus g of \mathcal{X} .

3 Weierstrass Semigroups for Curves (A)

Let $m > 2$ be a divisor of $q_0 + 1$, and suppose that $\Delta = \frac{q_0+1}{m} > 3$ is a prime. Let \mathcal{X}_m be the non-singular model of the plane curve over \mathbb{F}_q with affine equation

$$X^{2m} + X^m + Y^{q_0+1} = 0.$$

Proposition 2 (Section 3 in [7]). *The curve \mathcal{X}_m has the following properties.*

- (i) *The genus of \mathcal{X}_m is $g = \frac{1}{2}m(q_0 - 2) + 1$.*
- (ii) *\mathcal{X}_m is a maximal curve with*

$$q + 1 + m(q_0 - 2)q_0 + 2q_0 \geq 1 + qm$$

\mathbb{F}_q -rational points.

- (iii) *If ω is a primitive m -th root of -1 , then there exists an \mathbb{F}_q -rational point P such that*

$$\begin{aligned} -(\frac{1}{x-\omega})_\infty &= (q_0 + 1)P; \\ -(\frac{\bar{x}^{-1}\bar{y}^\Delta}{\bar{x}-\omega})_\infty &= (q_0 + 1 - \Delta)P; \\ \text{for all } n = 1, \dots, \frac{\Delta-1}{2}, (\frac{\bar{y}^n}{\bar{x}-\omega})_\infty &= (q_0 + 1 - n)P. \end{aligned}$$

Let P be as (iii) of Proposition 2. Then the Weierstrass semigroup $H(P)$ contains the following numerical semigroup

$$\Theta = \left\langle q_0 + 1 - \Delta, q_0 + 1 - \frac{\Delta - 1}{2}, q_0 + 1 - \frac{\Delta - 1}{2} + 1, \dots, q_0 + 1 \right\rangle.$$

We show that actually $H(P)$ coincides with Θ .

Proposition 3. $H(P) = \Theta$.

Proof. To prove the assertion we show that the number of gaps in Θ , that is, the number of integers in $\mathbf{N} \setminus \Theta$, is equal to the genus g of \mathcal{X}_m . Let $G = \{q_0 + 1 - \Delta, q_0 + 1 - \frac{\Delta-1}{2}, q_0 + 1 - \frac{\Delta-1}{2} + 1, q_0 + 1 - \frac{\Delta-1}{2} + 2, \dots, q_0 + 1\}$, and for $s \in \mathbf{N}_0$ let $G(s) = \{ig_1 + jg_2 | g_k \in G, i + j = s\}$. Note that, if $s < m$, then the intersection of two of these sets is always empty. In fact, the largest integer of $G(s-1)$ is $(s-1)(q_0 + 1)$, and it is smaller than the smallest integer of $G(s)$, that is $s(q_0 + 1 - \Delta)$. So, the number of gaps between $G(s-1)$ and $G(s)$ is exactly $s(q_0 + 1 - \Delta) - (s-1)(q_0 + 1) - 1 = q_0 - s\Delta$. Now we show that the number of gaps in $G(s)$ is at most $\frac{\Delta-1}{2}$. Let $v \in G(s)$. It is easily seen that $G(s)$ contains each v such that $s(q_0 + 1 - \frac{\Delta-1}{2}) \leq v \leq s(q_0 + 1)$. Moreover, if $s(q_0 + 1) - s\Delta + r\Delta - r\frac{\Delta-1}{2} \leq v \leq s(q_0 + 1) - s\Delta + r\Delta$, then

$$v = (s-r)(q_0 + 1 - \Delta) + (q_0 + 1 - \frac{\Delta-1}{2} + h) + (q_0 + 1 - \frac{\Delta-1}{2} + k),$$

where $h, k \in \{0, \dots, m-1\}$, and $r \in \{0, \dots, s-1\}$. Hence, $G(s)$ contains every integer greater than $(s-1)(q_0 + 1 - \Delta) + (q_0 + 1 - \frac{\Delta-1}{2})$, and less than $s(q_0 + 1 - \frac{\Delta-1}{2})$. For the same reason, the number of gaps in $G(m)$ is at most $\frac{\Delta-1}{2}$. In

particular, the greatest gap in $G(m)$ is less than $(m-1)(q_0+1-\Delta)+(q_0+1-\frac{\Delta-1}{2}) < 2g$, and the greatest $v \in G(m)$ is such that $v > 2g$. Therefore, we have at most

$$m\frac{\Delta-1}{2} + \sum_{i=1}^{m-1} (q_0 - i\Delta) = m\frac{\Delta-1}{2} + q_0(m-1) - \Delta\frac{m(m-1)}{2} = g$$

gaps less than $2g$. This shows that $\Theta \cap [0, 2g] = H(\gamma) \cap [0, 2g]$. To complete the proof, we need to show that Θ contains every integer greater than $2g$. This follows from the fact that if $s \geq m$, then $G(s) \cap G(s+1) \neq \emptyset$; moreover, $G(s)$ contains the gaps of $G(s+1)$, being $s(q_0+1) > (s+1)(q_0+1) - \Delta - \frac{\Delta-1}{2}$. This completes the proof.

4 Weierstrass Semigroups for Curves (B)

Let q_0 be a prime power such that $\Delta = \frac{q_0+1}{2}$ is a prime greater than 3. Let \mathcal{X}_i be the non-singular model of the curve over \mathbb{F}_q with affine equation

$$X^{2i+2} + X^{2i} + Y^{q_0+1} = 0,$$

where $1 \leq i \leq \Delta - 2$.

Proposition 4 ([8]). *The curve \mathcal{X}_i has the following properties.*

- (i) \mathcal{X}_i is maximal.
- (ii) The genus of \mathcal{X}_i is $g = q_0 - 1$.
- (iii) Let $n_1, n_2, \dots, n_{\frac{\Delta-1}{2}}$ be the integers such that $0 < n_j < \Delta$, and

$$n_j(i+1) \leq \left(\left\lfloor \frac{n_j i}{\Delta} \right\rfloor + 1 \right) \Delta,$$

for $j \in \{1, 2, \dots, \frac{\Delta-1}{2}\}$.

- There exists an \mathbb{F}_q -rational point P_1 of \mathcal{X}_i such that the Weierstrass semigroup $H(P_1)$ contains the following integers

$$q_0 + 1, q_0 + 1 - n_1, q_0 + 1 - n_2, \dots, q_0 + 1 - n_{\frac{\Delta-1}{2}}, q_0 + 1 - \Delta;$$

- there exists an \mathbb{F}_q -rational point P_2 of \mathcal{X}_i such that the Weierstrass semigroup $H(P_2)$ contains the following integers

$$q_0 + 1, q_0 + 1 - m_1, q_0 + 1 - m_2, \dots, q_0 + 1 - m_{\frac{\Delta-1}{2}}, q_0 + 1 - \Delta,$$

where $m_j = n_j i - \Delta \left\lfloor \frac{n_j i}{\Delta} \right\rfloor$;

- there exists an \mathbb{F}_q -rational point P_3 of \mathcal{X}_i such that the Weierstrass semigroup $H(P_3)$ contains the following integers

$$q_0 + 1, q_0 + 1 - k_1, q_0 + 1 - k_2, \dots, q_0 + 1 - k_{\frac{\Delta-1}{2}}, q_0 + 1 - \Delta,$$

where $k_j = \Delta \left(\left\lfloor \frac{n_j i}{\Delta} \right\rfloor + 1 \right) - n_j(i+1)$.

It is easily seen that, if $i = 1$, then $n_j = m_j = j$, and $k_j = \Delta - 2j$, for $j \in \{1, 2, \dots, \frac{\Delta-1}{2}\}$. So, from the above Proposition we have just two different Weierstrass semigroup of \mathcal{X}_1 , namely $H(P_1)$ and $H(P_3)$.

Proposition 5. *Assume that $i = 1$, and let*

$$\Theta = \left\langle q_0 + 1, (q_0 + 1) - 1, (q_0 + 1) - 2, \dots, (q_0 + 1) - \frac{\Delta-1}{2}, q_0 + 1 - \Delta \right\rangle.$$

Then $H(P_1) = \Theta$.

Proof. To prove the assertion we show that the number of gaps in Θ is equal to the genus g of \mathcal{X}_1 . Let $G = \{q_0 + 1, (q_0 + 1) - 1, (q_0 + 1) - 2, \dots, (q_0 + 1) - \frac{\Delta-1}{2}, q_0 + 1 - \Delta\}$, and for $s \in \{0, 1, 2\}$ let $G(s) = \{ig_1 + jg_2 \mid g_k \in G, i + j = s\}$. Note that $G(0) = \{0\}$, $G(1) = G$, and that the number of gaps in $G(1)$ is at most $\frac{\Delta-1}{2}$. Moreover, $G(1) \cap G(2) = \emptyset$. In fact, the largest integer of $G(1)$ is $q_0 + 1$, and it is smaller than the smallest integer of $G(2)$, that is $(q_0 + 1 - \Delta) + (q_0 + 1 - \frac{\Delta-1}{2}) = \frac{5(q_0+1)}{4} + \frac{1}{2}$. So, the number of gaps between $G(1)$ and $G(2)$ is exactly $(q_0 + 1 - \Delta) + (q_0 + 1 - \frac{\Delta-1}{2}) - (q_0 + 1) - 1 = \frac{q_0+1}{4} - \frac{1}{2}$. Now we show that $\left(\mathbf{N} \cap \left[\frac{5(q_0+1)}{4} + \frac{1}{2}, 2(q_0 + 1)\right]\right) \setminus G(2) = \emptyset$. Let $v \in G(2)$.

- If $\frac{5(q_0+1)}{4} + \frac{1}{2} \leq v \leq \frac{3(q_0+1)}{2}$, then $v = q_0 + 1 - \Delta + (q_0 + 1 - \frac{\Delta-1}{2} + h)$, where $h \in \{0, \dots, \frac{\Delta-1}{2}\}$;
- if $\frac{3(q_0+1)}{2} + 1 \leq v \leq 2(q_0 + 1)$, then $v = (q_0 + 1 - \frac{\Delta-1}{2} + h) + (q_0 + 1 - \frac{\Delta-1}{2} + k)$, where $h, k \in \{0, \dots, \frac{\Delta-1}{2}\}$.

Moreover, since that the largest integer in $G(2)$, that is $2(q_0 + 1)$, is smaller than $2g = 2(q_0 - 1)$, we have at most

$$(q_0 + 1 - \Delta - 1) + \frac{\Delta - 1}{2} + \left(\frac{q_0 + 1}{4} - \frac{1}{2}\right) = q_0 - 1 = g$$

gaps less than $2g$. This shows that $\Theta \cap [0, 2g] = H(\gamma_1) \cap [0, 2g]$. To complete the proof, we need to show that Θ contains every integer greater than $2g$. Consider now $G(s)$, for $s \geq 3$. We can observe that in $G(s)$ there is no gap, and that between $G(2)$ and $G(3)$ there is no integer. Also, it is easily seen that for $s > 2$, $G(s) \cap G(s+1) \neq \emptyset$ holds. This completes the proof.

Proposition 6. *Assume that $i = 1$, and let*

$$\Gamma = \langle q_0 + 1 - \Delta, q_0 + 1 - (\Delta - 2), q_0 + 1 - (\Delta - 4), \dots, q_0, q_0 + 1 \rangle.$$

Then $H(P_3) = \Gamma$.

Proof. We show that the number of gaps in Γ is equal to the genus g of \mathcal{X}_1 . Let $h = q_0 + 1 - \Delta$, and let $G = \{h, h + 2, h + 4, \dots, 2h - 1, 2h\}$. Clearly, Γ is generated by G . The number of gaps less than the first non-zero nongap

is $h - 1$, and the number of gaps in G is at most $\frac{h-1}{2}$. For $s \in \{0, 1, 2\}$ let $G(s) = \{ig_1 + jg_2 \mid g_k \in G, i + j = s\}$. Note that $G(0) = \{0\}$, $G(1) = G$, and that the number of gaps in $G(2)$ is at most $\frac{h-1}{2}$. In fact, for $v \in G(2)$, we have that if $v \leq 3h$, then $v = h + (h + 2i)$ for some $i \in \{1, \dots, \frac{h-1}{2}\}$, and if $3h - 1 < v \leq 4h$, then $v = (h + 2i) + (h + 2j)$ for some $i, j \in \{0, 1, \dots, \frac{h-1}{2}\}$. Since that $4h = 2(q_0 + 1) > 2g$, we have at most

$$2(h - 1) = g$$

gaps less than $2g$. Moreover, it is easily seen that Γ contains every integer greater than $2g$.

Consider the curve \mathcal{X}_i , when $q_0 = 9$, and $\Delta = 5$. We limit ourselves to the case $i = 1$, since for $i = 2$ and $i = 3$ the same Weierstrass semigroups are obtained. The curve \mathcal{X}_1 has equation

$$X^4 + X^2 + Y^{10} = 0, \quad (3)$$

its genus is $g = 8$, and the number of its \mathbb{F}_{81} -rational points is 226. By Propositions 5 and 6 there exist two \mathbb{F}_q -rational points P_1 and P_3 such that $H(P_1) = \langle 5, 8, 9 \rangle$ and $H(P_3) = \langle 5, 7, 9 \rangle$.

In Appendix, the Improved AG codes associated to the curve (3) with respect to P_3 will be referred to as codes (B1).

5 Weierstrass Semigroups for Curves (C)

Let s be a divisor of q_0 . Consider $H = \{c \in \mathbb{F}_q \mid c^{q_0} + c = 0\} < (\mathbb{F}_q, +)$, and let H_s be any additive subgroup of H with s elements. Let \mathcal{X} be the curve obtained as the image of the Hermitian curve by the following rational map

$$\varphi : \mathcal{H} \rightarrow \mathcal{X}, \quad (x, y) \mapsto (t, z) = \left(\prod_{a \in H_s} (x + a), y \right). \quad (4)$$

Proposition 7 ([5]). *The genus g of \mathcal{X} is equal to $\frac{1}{2}q_0(\frac{q_0}{s} - 1)$.*

Let \bar{P}_∞ be the only point at infinity of \mathcal{X} . This point is the image of the only infinite point P_∞ in \mathcal{H} by φ . Hence, the ramification index ϵ_{P_∞} of P_∞ is equal to $\deg(\varphi) = s$. Moreover, it is easily seen that

$$\text{ord}_{\bar{P}_\infty}(t) = \frac{1}{s} \sum_{a \in H_s} \text{ord}_{P_\infty}(x + a) = -(q_0 + 1),$$

and

$$\text{ord}_{\bar{P}_\infty}(z) = \frac{1}{s} \text{ord}_{P_\infty}(y) = -\frac{q_0}{s}.$$

Hence,

$$\left\langle \frac{q_0}{s}, q_0 + 1 \right\rangle \subseteq H(\bar{P}_\infty). \quad (5)$$

Proposition 8

$$H(\bar{P}_\infty) = \left\langle \frac{q_0}{s}, q_0 + 1 \right\rangle.$$

Proof. Note that $\frac{q_0}{s}$ and $q_0 + 1$ are coprime. Then by [12, Proposition 5.33] the genus of the semigroup generated by $\frac{q_0}{s}$ and $q_0 + 1$ is $\frac{1}{2}(\frac{q_0}{s} - 1)(q_0 + 1 - 1)$. Then the assertion follows from (5), together with Proposition 7.

Now some special cases for curves (C) are considered in greater detail.

5.1 $q_0 = 2^h$, $s = 2$, $H_s = \{0, 1\}$

Let q_0 be a power of 2, and $s = 2$. Since that q_0 is even, then $H = \mathbb{F}_{q_0}$. Let $H_s = \{0, 1\}$. Therefore, the rational map $\varphi : \mathcal{H} \rightarrow \mathcal{X}$ defined as in (4) is

$$\varphi(x, y) = (x^2 + x, y).$$

Proposition 9. \mathcal{X} has affine equation

$$Y^{q_0+1} = X^{\frac{q_0}{2}} + X^{\frac{q_0}{4}} + \dots + X^2 + X. \quad (6)$$

Proof. Let $(X, Y) \in \mathcal{H}$. We need to prove that $\varphi(X, Y)$ satisfies (6) for every point $(X, Y) \in \mathcal{H}$. To do this it is enough to observe that

$$(X^2 + X)^{\frac{q_0}{2}} + (X^2 + X)^{\frac{q_0}{4}} + \dots + (X^2 + X)^2 + (X^2 + X) = X^{q_0} + X,$$

and take into account that (X, Y) satisfies (1).

The curve with equation 6 was investigated in [6].

Proposition 10 ([6]). *There exists a point $P \in \mathcal{X}$ such that the Weierstrass semigroup at P is*

$$H(P) = \langle q_0 - 1, q_0, q_0 + 1 \rangle.$$

Therefore, taking into account Proposition 8, the curve \mathcal{X} has at least two different Weierstrass semigroups.

$q_0 = 8$. In this case the curve \mathcal{X} has equation

$$Y^9 = X^4 + X^2 + X,$$

its genus is $g = 12$, and the number of its \mathbb{F}_{64} -rational points is 257. In Appendix, we will denote by (C1a) the Improved AG codes constructed from the Weierstrass semigroup of Proposition 8, that is $H(\bar{P}_\infty) = \langle 4, 9 \rangle$, and by (C1b) those constructed from the Weierstrass semigroup of Proposition 10, $H(P) = \langle 7, 8, 9 \rangle$.

5.2 $q_0 = 2^h$, $s = \frac{q_0}{2}$, $H_s = \{a \in H \mid Tr(a) = 0\}$

Let q_0 be a power of 2, and $s = \frac{q_0}{2}$. Since that q_0 is even, then $H = \mathbb{F}_{q_0}$. Let $H_s = \{a \in H \mid Tr(a) = 0\}$, where $Tr(a) = a + a^2 + \dots + a^{\frac{q_0}{4}} + a^{\frac{q_0}{2}}$. Therefore, the rational map $\varphi : \mathcal{H} \rightarrow \mathcal{X}$ defined as in (4) is

$$\varphi(x, y) = \left(\prod_{a \in \mathbb{F}_{q_0}: Tr(a)=0} (x+a), y \right) = (Tr(x), y).$$

Proposition 11. *The curve \mathcal{X} has affine equation*

$$Y^{q_0+1} = X^2 + X. \quad (7)$$

Proof. Let $(X, Y) \in \mathcal{H}$. The assertion follows from the equation

$$(Tr(X))^2 + (Tr(X)) = X^{q_0} + X.$$

Proposition 12 ([6]). *There exists a point $P \in \mathcal{X}$ such that the Weierstrass semigroup at P is*

$$H(P) = \left\langle q_0 + 1 - \frac{q_0}{2}, q_0 + 1 - \left(\frac{q_0}{2} - 1\right), \dots, q_0, q_0 + 1 \right\rangle.$$

$q_0 = 16$. The curve \mathcal{X} has equation

$$Y^{17} = X^2 + X,$$

its genus is $g = 8$, and the number of its \mathbb{F}_{256} -rational points is 513. In Appendix, the Improved AG codes constructed from the Weierstrass semigroup of Proposition 8, that is $H(\bar{P}_\infty) = \langle 2, 17 \rangle$, will be referred to as codes (C2).

5.3 $q_0 = 16$, $s = 4$, $H_s = \mathbb{F}_4$

Let $q_0 = 16$, and $s = 4$. Since that q_0 is even, then $H = \mathbb{F}_{16}$. So, we can consider the case $H_s = \mathbb{F}_4$. The rational map $\varphi : \mathcal{H} \rightarrow \mathcal{X}$ defined as in (4) is

$$\varphi(x, y) = (x^4 + x, y).$$

Proposition 13. *The curve \mathcal{X} has affine equation*

$$Y^{17} = X^4 + X. \quad (8)$$

Moreover, \mathcal{X} has genus $g = 24$, and the number of its \mathbb{F}_{256} -rational points is 1025.

Proof. Let $(X, Y) \in \mathcal{H}$. We need to prove that $\varphi(X, Y)$ satisfies (8). To do this it is enough to observe that

$$(X^4 + X)^4 + (X^4 + X) = X^{16} + X,$$

and take into account that (X, Y) satisfies (1) for $q_0 = 16$. The second part of the assertion follows from Proposition 7.

In Appendix, the Improved AG codes constructed from the Weierstrass semigroup of Proposition 8, that is $H(\bar{P}_\infty) = \langle 4, 17 \rangle$, will be referred to as codes (C3).

5.4 $q_0 = 9$, $s = 3$, $H_s = \{x \in H \mid x^3 = \alpha x\}$, α primitive element of \mathbb{F}_9

Let $q_0 = 9$, $s = 3$. Also, let α be a primitive element of \mathbb{F}_9 . Note that $\alpha^4 = -1$. Let $H_s = \{x \in H \mid x^3 = \alpha x\}$. It is a straightforward computation to check that $H_s \subset H$. In fact, for $x \in H_s$,

$$x^9 + x = (\alpha x)^3 + x = \alpha^3 x^3 + x = \alpha^4 x + x = -x + x = 0.$$

The rational map $\varphi : \mathcal{H} \rightarrow \mathcal{X}$ defined as in (4) is

$$\varphi(x, y) = (x^3 - \alpha x, y).$$

Proposition 14. *The curve \mathcal{X} has affine equation*

$$Y^{10} = X^3 + \alpha^3 X. \quad (9)$$

Moreover, \mathcal{X} has genus $g = 9$, and the number of its \mathbb{F}_{81} -rational points is 244.

Proof. Let $(X, Y) \in \mathcal{H}$. We need to prove that $\varphi(X, Y)$ satisfies (9). To do this it is enough to observe that

$$(X^3 - \alpha X)^3 + \alpha^3 (X^3 - \alpha X) = X^9 - \alpha^4 X = X^9 + X,$$

and take into account that (X, Y) satisfies (1) for $q_0 = 9$. The second part of the assertion follows from Proposition 7.

In Appendix, the improved AG codes constructed from the Weierstrass semigroup $H(\bar{P}_\infty) = \langle 3, 10 \rangle$ will be referred to as codes (C4).

6 Weierstrass Semigroups for Curves (D)

Let m be a divisor of $q_0 + 1$, and let \mathcal{X}_m be the non-singular model of the curve over \mathbb{F}_q with affine equation

$$Y^m = X^{q_0} + X.$$

Proposition 15. *The curve \mathcal{X}_m has the following properties.*

- (i) \mathcal{X}_m is maximal.
- (ii) The genus of \mathcal{X}_m is $g = \frac{1}{2}(q_0 - 1)(m - 1)$.
- (iii) There exists precisely one \mathbb{F}_q -rational point of \mathcal{X}_m centred at the only point at infinity of the plane curve $Y^m = X^{q_0} + X$, and

$$\text{ord}_{\bar{P}_\infty}(x) = -m, \quad \text{ord}_{\bar{P}_\infty}(y) = -q_0$$

hold.

Proof. Assertion (i) and (ii) follows from [1, (IV) of Proposition 2.1]. It is straightforward to check that \mathcal{X}_m is the image of the Hermitian curve by the rational map

$$\varphi : \mathcal{H} \rightarrow \mathcal{X}_m, \quad (x, y) \mapsto (x, y^h),$$

where $h = \frac{q_0+1}{m}$. The only point at infinity \bar{P}_∞ of \mathcal{X}_m is the image of $P_\infty \in \mathcal{H}$ by φ , and it is easily seen that e_{P_∞} is equal to $\deg(\varphi) = h$. Let $f \in \bar{\mathbb{F}}_q(x, y^h)$, and let φ^* be the pull-back of φ . Then $\text{ord}_{P_\infty}(\varphi^*(f)) = e_{P_\infty} \text{ord}_{\bar{P}_\infty}(f)$. Therefore,

$$\frac{q_0+1}{m} \text{ord}_{\bar{P}_\infty}(x) = \text{ord}_{P_\infty}(x) = -(q_0+1),$$

and

$$\frac{q_0+1}{m} \text{ord}_{\bar{P}_\infty}(y) = \text{ord}_{P_\infty}(y^{\frac{q_0+1}{m}}) = \frac{q_0+1}{m} \text{ord}_{P_\infty}(y) = -\frac{q_0+1}{m} q_0.$$

Hence,

$$\text{ord}_{\bar{P}_\infty}(x) = -m, \text{ and } \text{ord}_{\bar{P}_\infty}(y) = -q_0.$$

Proposition 16. $H(\bar{P}_\infty) = \langle m, q_0 \rangle$.

Proof. Note that q_0 and m are coprime. Then by [12, Proposition 5.33] the genus of the semigroup generated by q_0 and m is $\frac{1}{2}(q_0-1)(m-1)$. Then the assertion follows from Proposition 15.

Proposition 17 ([6]). *There exists a point $P \in \mathcal{X}_m$ such that*

$$H(P) = \left\langle q_0 + 1 - \frac{q_0+1}{m}, q_0 + 1 - \left(\frac{q_0+1}{m} - 1\right), \dots, q_0 + 1 \right\rangle.$$

Now we consider the curve \mathcal{X}_m for particular values of q_0 and m .

6.1 $q_0 = 7, m = 4$

The curve \mathcal{X}_m has equation

$$Y^4 = X^7 + X,$$

its genus is $g = 9$, and the number of its \mathbb{F}_{49} -rational points is 176. In Appendix, (D1a) will denote the Improved AG codes constructed from the Weierstrass semigroup of Proposition 16, that is $H(\bar{P}_\infty) = \langle 4, 7 \rangle$, and (D1b) those constructed from the Weierstrass semigroup of Proposition 17, $H(P) = \langle 6, 7, 8 \rangle$.

6.2 $q_0 = 7, m = 2$

The curve \mathcal{X}_m has equation

$$Y^2 = X^7 + X,$$

its genus is $g = 3$, and the number of its \mathbb{F}_{49} -rational points is 92. In Appendix, (D2a) will denote the Improved AG codes constructed from the Weierstrass semigroup of Proposition 16, that is $H(\bar{P}_\infty) = \langle 2, 7 \rangle$, and (D2b) those constructed from the Weierstrass semigroup of Proposition 17, $H(P) = \langle 4, 5, 6, 7 \rangle$.

6.3 $q_0 = 8, m = 3$

The curve \mathcal{X}_m has equation

$$Y^3 = X^8 + X,$$

its genus is $g = 7$, and the number of its \mathbb{F}_{64} -rational points is 177. In Appendix, (D3a) will denote the improved AG codes constructed from the Weierstrass semigroup of Proposition 16, that is $H(\bar{P}_\infty) = \langle 3, 8 \rangle$, and (D3b) those constructed from the Weierstrass semigroup of Proposition 17, $H(P) = \langle 6, 7, 8, 9 \rangle$.

6.4 $q_0 = 9, m = 5$

The curve \mathcal{X}_m has equation

$$Y^5 = X^9 + X,$$

its genus is $g = 16$, and the number of its \mathbb{F}_{81} -rational points is 370. In Appendix, (D4a) will denote the Improved AG codes constructed from the Weierstrass semigroup of Proposition 16, that is $H(\bar{P}_\infty) = \langle 5, 9 \rangle$, and (D4b) those constructed from the Weierstrass semigroup of Proposition 17, $H(P) = \langle 8, 9, 10 \rangle$.

7 Weierstrass Semigroups for Curves (E)

Let m be a divisor of $q - 1$ and let \mathcal{X}_m be the non-singular model of the plane curve

$$Y^{\frac{q-1}{m}} = X(X + 1)^{q_0 - 1}. \quad (10)$$

Proposition 18. *The curve \mathcal{X}_m is the image of the Hermitian curve by the rational map*

$$\varphi : \mathcal{H} \rightarrow \mathcal{X}_m, \quad (x, y) \mapsto (t, z) = (x^{q_0 - 1}, y^m). \quad (11)$$

Proof. Let (X, Y) be a point in \mathcal{H} . We need to prove that $\varphi(X, Y)$ satisfies (10). This follows from

$$X^{q_0 - 1}(X^{q_0 - 1} + 1)^{q_0 - 1} = (X^{q_0} + X)^{q_0 - 1} = (Y^{q_0 + 1})^{q_0 - 1} = (Y^m)^{\frac{q-1}{m}}.$$

It is easily seen that the rational functions $t = x^{q_0 - 1}$, and $z = y^m$ have just a pole \bar{P}_∞ , which is the image by φ of the only infinite point P_∞ of \mathcal{H} . Therefore, the ramification index e_{P_∞} is equal to $\deg(\varphi)$.

Some properties of the curve \mathcal{X}_m were investigated in [5, Corollary 4.9 and Example 6.3].

Proposition 19 ([5]). *The genus of \mathcal{X}_m is equal to*

$$\frac{1}{2m}(q_0 - 1)(q_0 + 1 - d),$$

where $d = (m, q_0 + 1)$. The degree $\deg(\varphi)$ of the rational map φ is equal to m .

Proposition 20 ([5]). *Let α be a primitive m -th root of unity in \mathbb{F}_q , and let*

$$G_m = \langle \phi : \mathcal{H} \rightarrow \mathcal{H} \mid (X, Y) \mapsto (\alpha^{q_0+1}, \alpha Y) \rangle.$$

Then \mathcal{X}_m can be seen as the quotient curve of \mathcal{H} by the group G_m .

Note that a point (a, b) of the plane curve (10) is non-singular provided that $b \neq 0$. Let $P_{(a,b)}$ be the only point of \mathcal{X}_m lying over at (a, b) .

Proposition 21. *Let $b \neq 0$. Then the size of $\varphi^{-1}(P_{(a,b)})$ is equal to m .*

Proof. Clearly, $\varphi^{-1}(P_{(a,b)}) = \{(X, Y) \in \mathcal{H} \mid X^{q_0-1} = a, Y^m = b\}$. The number of roots of $Y^m - b$ is m , for all $b \neq 0$, since that $\text{char}(\bar{\mathbb{F}}_q) \nmid m$. Hence, $\#\varphi^{-1}(a, b) \geq m$, when $b \neq 0$. Now taking into account Proposition 19, we have also that $\#\varphi^{-1}(a, b) \leq m$. Then the assertion follows.

Let $(a, 0)$ be a point of the plane curve (10). If $a = 0$, then $(a, 0)$ is non-singular. Let $P_{(0,0)}$ be the only point of \mathcal{X}_m lying over $(0, 0)$. Clearly, $\varphi^{-1}(P_{(0,0)}) = \{P_0 = (0, 0) \in \mathcal{H}\}$. Hence, $e_{P_0} = \deg(\varphi) = m$. Assume now that $a \neq 0$. Then $a = -1$. We consider the set L of points of \mathcal{H} whose image by φ is a point of \mathcal{X}_m lying over $(-1, 0)$. Clearly, $L = \{(X, Y) \in \mathcal{H} \mid X^{q_0-1} = -1, Y^m = 0\} = \{(X, 0) \in \mathcal{H} \mid X^{q_0-1} + 1 = 0\}$. Let $\{P_1, P_2, \dots, P_{q_0-1}\}$ be the points in \mathcal{H} such that $Y = 0$ and $X^{q_0-1} + 1 = 0$.

Lemma 1. *Let φ be as in (11). The ramification points of φ are $P_0, P_\infty, P_1, P_2, \dots, P_{q_0-1}$. In particular, $e_{P_0} = e_{P_\infty} = m$, and $e_{P_i} = d$, for $1 \leq i \leq q_0 - 1$, where $d = (m, q_0 + 1)$.*

Proof. By the previous results, we only need to calculate e_{P_i} . Being $\mathcal{X} \cong \mathcal{H}/G_m$, the integer e_P represents the stabilizer of G_m at P , for $P \in \mathcal{H}$; i.e. $e_P = \#Stab_P(G_m)$. Note that $\#Stab_{(X,0)}(G_m) = \#\{0 \leq i < m \mid \alpha^{i(q_0+1)} = 1\} = (m, q_0 + 1)$. Hence,

$$e_{P_i} = d,$$

where $d = (m, q_0 + 1)$.

A straightforward corollary to Lemma 1 is that there are exactly $\frac{d(q_0-1)}{m}$ points of \mathcal{X}_m , say $\bar{P}_1, \bar{P}_2, \dots, \bar{P}_{\frac{d(q_0-1)}{m}}$, lying over $(-1, 0)$.

Proposition 22. *Let $D = \bar{P}_1 + \bar{P}_2 + \dots + \bar{P}_{\frac{d(q_0+1)}{m}}$. Then*

$$(z) = \frac{m}{d}D + \bar{P}_0 - q_0\bar{P}_\infty, \quad (t+1) = \frac{q_0+1}{d}D - \frac{q-1}{m}\bar{P}_\infty.$$

Proof. The assertion follows from the following straightforward computation:

•

$$\text{ord}_{\bar{P}_0}(z) = \frac{1}{m}\text{ord}_{P_0}(y^m) = \text{ord}_{P_0}(y) = 1;$$

$$\text{ord}_{\bar{P}_0}(t+1) = \frac{1}{m}\text{ord}_{P_0}(x^{q_0-1} + 1) = 0;$$

•

$$\text{ord}_{\bar{P}_\infty}(z) = \frac{1}{m} \text{ord}_{P_\infty}(y^m) = -q_0;$$

$$\text{ord}_{\bar{P}_\infty}(t+1) = \frac{1}{m} \text{ord}_{P_\infty}(x^{q_0-1} + 1) = \frac{q_0-1}{m} \text{ord}_{P_\infty}(x) = -\frac{q_0-1}{m};$$

- for $1 \leq i \leq \frac{d(q_0-1)}{m}$,

$$\text{ord}_{\bar{P}_i}(z) = \frac{1}{d} \text{ord}_{P_i}(y^m) = \frac{m}{d};$$

$$\text{ord}_{\bar{P}_i}(t+1) = \frac{1}{d} \text{ord}_{P_i}(x^{q_0-1} + 1) = \frac{q_0+1}{d}.$$

Proposition 23. Let $i, j \in \mathbf{N}_0$ such that

$$i \geq j \frac{q_0+1}{m}.$$

Then $iq_0 - j \frac{q-1}{m} \in H(\bar{P}_\infty)$.

Proof. Let $\gamma = z^i(t+1)^{-j}$ and $D = \bar{P}_1 + \bar{P}_2 + \dots + \bar{P}_{\frac{d(q_0+1)}{m}}$. Then

$$(\gamma) = \frac{im}{d}D + \bar{P}_0 - iq_0\bar{P}_\infty - \frac{j(q_0+1)}{d}D + \frac{j(q-1)}{m}\bar{P}_\infty.$$

Therefore,

$$(\gamma)_\infty = (iq_0 - j \frac{q-1}{m})\bar{P}_\infty.$$

7.1 $q_0 = 7, m = 3$

The curve \mathcal{X}_m has equation

$$Y^{16} = X(X+1)^6,$$

its genus is $g = 7$, and the number of its \mathbb{F}_{49} -rational points is 148. By Proposition 23 we have that 5, 7, 8 are non-gaps at \bar{P}_∞ . Moreover, the genus of the numerical semigroup generated by these integers is equal to the genus of \mathcal{X}_m . Hence, $H(\bar{P}_\infty) = \langle 5, 7, 8 \rangle$. In Appendix, (E1) will denote the Improved AG codes constructed from $H(\bar{P}_\infty)$.

7.2 $q_0 = 9, m = 4$

The curve \mathcal{X}_m has equation

$$Y^{20} = X(X+1)^8,$$

its genus is $g = 8$, and the number of its \mathbb{F}_{81} -rational points is 226. By Proposition 23 we have that 5, 7, 9 are non-gaps at \bar{P}_∞ . Moreover, the genus of the numerical semigroup generated by these integers is equal to the genus of \mathcal{X}_m .

Hence, $H(\bar{P}_\infty) = \langle 5, 7, 9 \rangle$. In Appendix, (E2) will denote the Improved AG codes constructed from $H(\bar{P}_\infty)$.

7.3 $q_0 = 16, m = 5$

The curve \mathcal{X}_m has equation

$$Y^{51} = X(X + 1)^{15},$$

its genus is $g = 24$, and the number of its \mathbb{F}_{256} -rational points is 1025. By Proposition 23 we have that 10, 13, 16, 17 are non-gaps at \bar{P}_∞ . Moreover, the genus of the numerical semigroup generated by these integers is equal to the genus of \mathcal{X}_m . Hence, $H(\bar{P}_\infty) = \langle 10, 13, 16, 17 \rangle$. In Appendix, (E3) will denote the Improved AG codes constructed from $H(\bar{P}_\infty)$.

References

1. Cossidente, A., Korchmáros, G., Torres, F.: Curves of large genus covered by the Hermitian curve. *Comm. Algebra* 28, 4707–4728 (2000)
2. Feng, G.L., Rao, T.R.N.: A simple approach for construction of algebraic-geometric codes from affine plane curves. *IEEE Trans. Inform. Theory* 40, 1003–1012 (1994)
3. Feng, G.L., Rao, T.R.N.: Improved geometric Goppa codes, Part I: Basic theory. *IEEE Trans. Inform. Theory* 41, 1678–1693 (1995)
4. Fuhrmann, R., Garcia, A., Torres, F.: On maximal curves. *J. Number Theory* 67, 29–51 (1997)
5. Garcia, A., Stichtenoth, H., Xing, C.P.: On subfields of the Hermitian function field. *Compositio Math.* 120, 137–170 (2000)
6. Garcia, A., Viana, P.: Weierstrass points on certain non-classic curves. *Arch. Math.* 46, 315–322 (1986)
7. Giulietti, M., Hirschfeld, J.W.P., Korchmáros, G., Torres, F.: A family of curves covered by the Hermitian curve. *Sémin. Congr., Soc. Math. France* 21, 63–78 (2009)
8. Giulietti, M., Hirschfeld, J.W.P., Korchmáros, G., Torres, F.: Curves covered by the Hermitian curve. *Finite Fields Appl.* 12, 539–564 (2006)
9. Garcia, A., Kim, S.J., Lax, R.F.: Consecutive Weierstrass gaps and minimum distance of Goppa codes. *J. Pure Appl. Algebra* 84, 199–207 (1993)
10. Goppa, V.D.: Codes associated with divisors. *Problemi Peredachi Informatsii* 13, 33–39 (1977)
11. Hirschfeld, J.W.P., Korchmáros, G., Torres, F.: *Algebraic Curves over a Finite Field*. Princeton University Press, Princeton (2008)
12. Høholdt, T., van Lint, J.H., Pellikaan, R.: Algebraic Geometry codes. In: Pless, V.S., Huffman, W.C., Brualdi, R.A. (eds.) *Handbook of Coding Theory*, vol. 1, pp. 871–961. Elsevier, Amsterdam (1998)
13. MinT. Tables of optimal parameters for linear codes, University of Salzburg, <http://mint.sbg.ac.at/>
14. Tsfasman, M.A., Vladut, S.G.: *Algebraic-geometric codes*. Kluwer, Amsterdam (1991)

Appendix: Improvements on MinT's Tables

In this Appendix, we consider the parameters of some of the codes $\tilde{C}_d(P)$ (see Definition 1), where P is a point of a curve \mathcal{X} belonging to one of the families (A)-(E).

We recall the following propagation rules.

Proposition 24 (see Exercise 7 in [14])

- (i) If there exists a q -ary linear code of length n , dimension k and minimum distance d , then for each non-negative integer $s < d$ there exists a q -ary linear code of length n , dimension k and minimum distance $d - s$.

Table 1. Improvements on [13] - $q = 49$

n	k	d	Prop.24(iii)	$n - s$	$k - s$	d	Code	n	k	d	Prop.24(iii)	$n - s$	$k - s$	d	Code
91	80	9	$s=10$	81	70	9	(D2a)(D2b)	91	50	39	$s=10$	81	40	39	(D2a)(D2b)
91	79	10	$s=10$	81	69	10	(D2a)(D2b)	91	49	40	$s=10$	81	39	40	(D2a)(D2b)
91	78	11	$s=10$	81	68	11	(D2a)(D2b)	91	48	41	$s=10$	81	38	41	(D2a)(D2b)
91	77	12	$s=10$	81	67	12	(D2a)(D2b)	91	47	42	$s=10$	81	37	42	(D2a)(D2b)
91	76	13	$s=10$	81	66	13	(D2a)(D2b)	91	46	43	$s=10$	81	36	43	(D2a)(D2b)
91	75	14	$s=10$	81	65	14	(D2a)(D2b)	91	45	44	$s=10$	81	35	44	(D2a)(D2b)
91	74	15	$s=10$	81	64	15	(D2a)(D2b)	91	44	45	$s=10$	81	34	45	(D2a)(D2b)
91	73	16	$s=10$	81	63	16	(D2a)(D2b)	91	43	46	$s=10$	81	33	46	(D2a)(D2b)
91	72	17	$s=10$	81	62	17	(D2a)(D2b)	91	42	47	$s=10$	81	32	47	(D2a)(D2b)
91	71	18	$s=10$	81	61	18	(D2a)(D2b)	91	41	48	$s=10$	81	31	48	(D2a)(D2b)
91	70	19	$s=10$	81	60	19	(D2a)(D2b)	91	40	49	$s=10$	81	30	49	(D2a)(D2b)
91	69	20	$s=10$	81	59	20	(D2a)(D2b)	91	39	50	$s=10$	81	29	50	(D2a)(D2b)
91	68	21	$s=10$	81	58	21	(D2a)(D2b)	91	38	51	$s=10$	81	28	51	(D2a)(D2b)
91	67	22	$s=10$	81	57	22	(D2a)(D2b)	91	37	52	$s=10$	81	27	52	(D2a)(D2b)
91	66	23	$s=10$	81	56	23	(D2a)(D2b)	91	36	53	$s=10$	81	26	53	(D2a)(D2b)
91	65	24	$s=10$	81	55	24	(D2a)(D2b)	147	129	12	$s=18$	129	111	12	(E1)
91	64	25	$s=10$	81	54	25	(D2a)(D2b)	147	128	13	$s=32$	115	96	13	(E1)
91	63	26	$s=10$	81	53	26	(D2a)(D2b)	147	127	14	$s=32$	115	95	14	(E1)
91	62	27	$s=10$	81	52	27	(D2a)(D2b)	147	126	15	$s=44$	103	82	15	(E1)
91	61	28	$s=10$	81	51	28	(D2a)(D2b)	147	125	16	$s=46$	101	79	16	(E1)
91	60	29	$s=10$	81	50	29	(D2a)(D2b)	147	124	17	$s=58$	89	66	17	(E1)
91	59	30	$s=10$	81	49	30	(D2a)(D2b)	147	123	18	$s=58$	89	65	18	(E1)
91	58	31	$s=10$	81	48	31	(D2a)(D2b)	147	122	19	$s=58$	89	64	19	(E1)
91	57	32	$s=10$	81	47	32	(D2a)(D2b)	147	121	20	$s=58$	89	63	20	(E1)
91	56	33	$s=10$	81	46	33	(D2a)(D2b)	147	120	21	$s=58$	89	62	21	(E1)
91	55	34	$s=10$	81	45	34	(D2a)(D2b)	147	119	22	$s=58$	89	61	22	(E1)
91	54	35	$s=10$	81	44	35	(D2a)(D2b)	147	118	23	$s=58$	89	60	23	(E1)
91	53	36	$s=10$	81	43	36	(D2a)(D2b)	147	117	24	$s=58$	89	59	24	(E1)
91	52	37	$s=10$	81	42	37	(D2a)(D2b)	147	116	25	$s=58$	89	58	25	(E1)
91	51	38	$s=10$	81	41	38	(D2a)(D2b)	147	115	26	$s=58$	89	57	26	(E1)

- (ii) If there exists a q -ary linear code of length n , dimension k and minimum distance d , then for each non-negative integer $s < k$ there exists a q -ary linear code of length n , dimension $k - s$ and minimum distance d .
 - (iii) If there exists a q -ary linear code of length n , dimension k and minimum distance d , then for each non-negative integer $s < k$ there exists a q -ary linear code of length $n - s$, dimension $k - s$ and minimum distance d .

The notation of Section 2 is kept. By Theorem 1, together with both (i) and (ii) of Proposition 24, a code $\tilde{C}_d(P)$ can be assumed to be an $[n, k, d]_q$ code with $n = \#\mathcal{X}(\mathbb{F}_q) - 1$ and $k = n - r_d$. Note that r_d can be obtained from the Weierstrass semigroup $H(P)$ by straightforward computation.

Table 2. Improvements on [13] - $q = 49$

Table 3. Improvements on [13] - $q = 64$

n	k	d	Prop.24(iii)	$n - s$	$k - s$	d	Code
176	162	10	s=29	147	133	10	(D3b)
176	159	12	s=14	162	145	12	(D3a)
176	157	14	s=14	162	143	14	(D3a)
256	232	15	s=30	226	202	15	(C1a)
256	231	16	s=30	226	201	16	(C1a)
256	230	16	s=19	237	211	16	(C1b)
256	229	18	s=30	226	199	18	(C1b)
256	228	18	s=28	228	200	18	(C1a)
256	226	20	s=28	228	198	20	(C1a)
256	225	21	s=28	228	197	21	(C1a)
256	222	24	s=28	228	194	24	(C1a)

Table 4. Improvements on [13] - $q = 81$

n	k	d	Prop.24(iii)	$n - s$	$k - s$	d	Code
225	207	12	s=24	201	183	12	(B1)(E2)
243	225	12	s=42	201	183	12	(C4)
243	223	13	s=16	227	207	13	(C4)
243	222	14	s=16	227	206	14	(C4)
243	221	15	s=16	227	205	15	(C4)
243	220	16	s=16	227	204	16	(C4)
243	218	18	s=16	227	202	18	(C4)
369	339	18	s=25	344	314	18	(D4a)(D4b)
369	337	19	s=4	365	333	19	(D4a)
369	336	20	s=36	333	300	20	(D4a)
369	334	21	s=28	341	306	21	(D4a)
369	333	23	s=66	303	267	23	(D4a)(D4b)
369	332	24	s=66	303	266	24	(D4a)(D4b)
369	330	25	s=64	305	266	25	(D4b)
369	328	27	s=64	305	264	27	(D4a)
369	327	28	s=64	305	263	28	(D4a)
369	323	32	s=64	305	259	32	(D4a)(D4b)

Table 5. Improvements on [13] - $q = 256$

n	k	d	Prop.24(iii)	$n - s$	$k - s$	d	Code
512	495	14	s=186	326	309	14	(C2)
512	494	16	s=188	324	306	16	(C2)
512	493	17	s=188	324	305	17	(C2)
512	492	18	s=188	324	304	18	(C2)
512	491	19	s=188	324	303	19	(C2)
512	490	20	s=188	324	302	20	(C2)
512	489	21	s=188	324	301	21	(C2)
512	488	22	s=188	324	300	22	(C2)
512	487	23	s=188	324	299	23	(C2)
512	486	24	s=188	324	298	24	(C2)
512	485	25	s=188	324	297	25	(C2)
512	484	26	s=188	324	296	26	(C2)
512	483	27	s=188	324	295	27	(C2)
512	482	28	s=188	324	294	28	(C2)
512	481	29	s=188	324	293	29	(C2)
512	480	30	s=188	324	292	30	(C2)
512	479	31	s=188	324	291	31	(C2)
512	478	32	s=188	324	290	32	(C2)
512	477	33	s=188	324	289	33	(C2)
512	476	34	s=188	324	288	34	(C2)
512	475	35	s=188	324	287	35	(C2)
512	474	36	s=188	324	286	36	(C2)
512	473	37	s=188	324	285	37	(C2)
512	472	38	s=188	324	284	38	(C2)
512	471	39	s=188	324	283	39	(C2)
512	470	40	s=188	324	282	40	(C2)
512	469	41	s=188	324	281	41	(C2)
512	468	42	s=188	324	280	42	(C2)
512	467	43	s=188	324	279	43	(C2)
512	466	44	s=188	324	278	44	(C2)
512	465	45	s=188	324	277	45	(C2)
512	464	46	s=188	324	276	46	(C2)
512	463	47	s=188	324	275	47	(C2)
512	462	48	s=188	324	274	48	(C2)
512	461	49	s=188	324	273	49	(C2)
512	460	50	s=188	324	272	50	(C2)
512	459	51	s=188	324	271	51	(C2)
512	458	52	s=188	324	270	52	(C2)

Table 6. Improvements on [13] - $q = 256$

n	k	d	Prop.24(iii)	$n - s$	$k - s$	d	Code
512	457	53	s=188	324	269	53	(C2)
512	456	54	s=188	324	268	54	(C2)
512	455	55	s=188	324	267	55	(C2)
512	454	56	s=188	324	266	56	(C2)
512	453	57	s=188	324	265	57	(C2)
512	452	58	s=188	324	264	58	(C2)
512	451	59	s=188	324	263	59	(C2)
512	450	60	s=188	324	262	60	(C2)
512	449	61	s=188	324	261	61	(C2)
512	448	62	s=188	324	260	62	(C2)
512	447	63	s=188	324	259	63	(C2)
512	446	64	s=188	324	258	64	(C2)
512	445	65	s=188	324	257	65	(C2)
512	444	66	s=188	324	256	66	(C2)
1024	980	27	s=247	777	733	27	(C3)
1024	979	28	s=248	776	731	28	(C3)
1024	978	30	s=413	611	565	30	(C3)
1024	976	32	s=445	579	531	32	(C3)
1024	975	33	s=477	547	498	33	(C3)
1024	974	35	s=494	530	480	35	(C3)
1024	973	36	s=494	530	479	36	(C3)
1024	972	40	s=500	524	472	40	(C3)
1024	970	41	s=498	526	472	41	(C3)
1024	969	42	s=498	526	471	42	(C3)
1024	968	43	s=498	526	470	43	(C3)
1024	967	44	s=498	526	469	44	(C3)
1024	966	45	s=498	526	468	45	(C3)
1024	965	46	s=498	526	467	46	(C3)
1024	964	47	s=498	526	466	47	(C3)
1024	963	48	s=498	526	465	48	(C3)
1024	962	49	s=498	526	464	49	(C3)
1024	961	50	s=498	526	463	50	(C3)
1024	960	51	s=498	526	462	51	(C3)
1024	959	52	s=498	526	461	52	(C3)
1024	958	53	s=498	526	460	53	(C3)
1024	957	54	s=498	526	459	54	(C3)
1024	956	55	s=498	526	458	55	(C3)

The following tables provide a list of the codes obtained in this paper which, according to the online database MinT [13], have larger minimum distance with respect to the previously known codes with same dimension and same length. The value of s in each entry means that, for each $i \leq s$, the $[n-i, k-i, d]_q$ code obtained from $\tilde{C}_d(P)$ by (iii) of Proposition 24 has better parameters than the known codes as well. For the sake of completeness, the parameters $[n-s, k-s, d]$ appear in the tables.

On Linear Cryptanalysis with Many Linear Approximations

Benoît Gérard and Jean-Pierre Tillich

INRIA project-team SECRET, France
`{benoit.gerard,jean-pierre.tillich}@inria.fr`

Abstract. In this paper we present a theoretical framework to quantify the information brought by several linear approximations of a block-cipher without putting any restriction on these approximations. We quantify here the entropy of the key given the plaintext-ciphertext pairs statistics which is a much more accurate measure than the ones studied earlier. The techniques which are developed here apply to various ways of performing the linear attack and can also been used to measure the entropy of the key for other statistical attacks. Moreover, we present a realistic attack on the full DES with a time complexity of 2^{48} for 2^{41} pairs what is a big improvement comparing to Matsui's algorithm 2 ($2^{51.9}$).

Keywords: linear cryptanalysis, multiple linear approximations, information theory.

1 Introduction

Related work

Linear cryptanalysis is probably one of the most powerful tools available for attacking symmetric cryptosystems. It was invented by Matsui [1,2] to break the DES cipher building upon ideas put forward in [3,4]. It was quickly discovered that other ciphers can be attacked in this way, for instance FEAL [5], LOKI [6], SAFER [7]. It is a known plaintext attack which takes advantage of probabilistic linear equations that involve bits of the plaintext \mathbf{P} , the ciphertext \mathbf{C} and the key \mathbf{K}

$$\Pr(\langle \pi, \mathbf{P} \rangle \oplus \langle \gamma, \mathbf{C} \rangle \oplus \langle \kappa, \mathbf{K} \rangle = b) = \frac{1}{2} + \epsilon. \quad (1)$$

Usually, ϵ is called the *bias* of the equation, π, γ and κ are linear masks and $\langle \pi, \mathbf{P} \rangle$ denotes the following inner product between $\pi = (\pi_i)_{1 \leq i \leq m}$ and $\mathbf{P} = (P_i)_{1 \leq i \leq m}$, $\langle \pi, \mathbf{P} \rangle \stackrel{\text{def}}{=} \bigoplus_{i=1}^m \pi_i P_i$. There might be several different linear approximations of this kind we have at our disposal and we let n be their number. We denote the corresponding key masks by $\kappa_i = (\kappa_i^j)_{1 \leq j \leq k}$ and the corresponding biases by ϵ_i for $i \in \{1, \dots, n\}$.

Such an attack can be divided in three parts:

- *Distillation phase*: It consists in extracting from the available plaintext-ciphertext pairs the relevant parts of the data. Basically, for each linear approximation, the attacker counts how many times $\langle \pi, \mathbf{P} \rangle \oplus \langle \gamma, \mathbf{C} \rangle$ evaluates to zero.
- *Analysis phase*: It consists in extracting from the values taken by the counters some information on the key and testing whether some key guesses are correct or not by using the linear approximation(s) (1) as a distinguisher. Typically, the output of this phase is a list of all possible subkey guesses sorted relatively to their likelihood.
- *Search phase*: It typically consists in finding the remaining key bits by exhaustive search.

In [1] Matsui used only one approximation to distinguish wrong last round keys from the right one. One year later, he refined his attack by using a second approximation obtained by symmetry [2] and by also distinguishing with them the first round key. Later Vaudenay [8] has presented a framework for statistical cryptanalysis where Matsui's attack is presented as a particular case. With Junod, he has also studied the optimal way of merging information from two (or more) approximations [9]. This kind of attack can use several approximations but the key masks must have disjoint supports. A second approach of using multiple equations is given by Kaliski and Robshaw [10]. They improved Matsui's first attack using several approximations which have the same key mask κ . Biryukov and al. suggest in [11] a way of using multiple linear approximations without putting any restriction on them. They present a theoretical framework to compute the expected rank of the good subkey guess. This framework has been used for SERPENT cryptanalysis [12,13]. Recent works by Hermelin and al. [14] give a way to compute the good subkey ranking probability law in the case of *multidimensional linear cryptanalysis*. More details on this work are given later to compare it to ours.

All these improvements have a common goal: reducing the amount of messages needed for the attack. Clearly, using several approximations should give more information than a single one.

Our contribution

The purpose of this paper is to study how much multiple linear approximations may benefit linear cryptanalysis. We aim at quantifying accurately how much information is gained on the key from the knowledge of statistical data derived from linear characteristics of type (1).

Several statistics have been proposed to study how many plaintext-ciphertext pairs we need in order to carry out successfully a linear cryptanalysis. This includes for instance the probability of guessing incorrectly a linear combination of key bits by Matsui's Algorithm 1 [2], the ranking of the right subkey in the ordered list of candidates [15,16] or the expected size of the number of keys which are more likely than the right key [11]. Some of these statistics are either not relevant for multilinear cryptanalysis or are extremely difficult to compute (such

as for instance the ranking statistics of [15,16] when we do not allow restrictions on the approximations used). This is not the case of the expected size of the number L of keys which are more likely than the right key considered in [11]. However, this kind of statistics also leads to pessimistic predictions concerning the number of plaintext-ciphertexts which are needed. To be more specific, it turns out that its prediction of the number of plaintext/ciphertext pairs ensuring that the most likely key is indeed the right key is in many cases twice the number of plaintext/ciphertexts which are really needed ! This is detailed in Proposition 3.1. We obtain the right amount by our analysis. It consists in studying instead of the expectation of L , the *entropy* $\mathcal{H}(\mathbf{K}|\mathbf{Y})$ of the key \mathbf{K} (or more generally $\mathcal{H}(\mathbf{K}'|\mathbf{Y})$, where \mathbf{K}' is a certain subkey of \mathbf{K} - for instance it can be the part of the key involved in a distinguisher attack) given the statistics \mathbf{Y} we have derived from the plaintext-ciphertext pairs.

The fact that the entropy is a much better statistic than the expectation of L is related to the following probabilistic phenomenon : this expectation is in a rather wide range of amount of plaintext-ciphertext pairs exponential in the key size k , while for most plaintext-ciphertext pairs the most likely key is the right one. This comes from the fact that rare events (of exponentially small probability) yield values of L which are exponentially large in k . In other words, while for typical plaintext-ciphertext pairs L is equal to zero, for some rare occurrences of the plaintext-ciphertext pairs L is very large, and this accounts quite heavily in the expectation of L . The entropy behaves here much better. In a certain sense, it is related to the expectation of the logarithm $\log_2(L)$. The logarithm of L varies much less than L and this why the typical size of $\log L$ coincides quite well with the expectation.

Despite the fact that it is much more desirable to estimate the entropy than the expectation of L , it might seem that this quantity is much harder to calculate. Our main result is to give here a lower bound on this quantity (see Subsections 3.1 and 3.2) which is quite sharp. The sharpness of the bound is illustrated by the results of Subsection 3.3. We apply this bound in three different scenarios: (i) the linear attack which recovers only the linear combination of the key bits, (ii) the usual linear distinguishing attack which recovers some linear combinations of the key bits of the first (and/or) last round, and (iii) the algorithm MK2 in [11]. We wish to emphasize the fact that the technique to derive the lower bound is quite general and applies in a very wide range of situations, and not only in the case where \mathbf{Y} corresponds to a function of the counters of linear approximations (see Subsection 3.1). A second useful property of this lower bound on the entropy is that it gives an upper bound on the information we gain on the \mathbf{K} when we know \mathbf{Y} which is independent of the algorithm we use afterwards to extract this information.

Complementarity with [14]

The work of Hermelin, Nyberg and Cho gives a framework for multidimensional linear cryptanalysis that does not require statistical independence between the approximations used. A set of m linearly independent approximations is chosen and the correlations of the linear combinations of those approximations are

computed. For each plaintext/ciphertext pair, the m bits vector corresponding to the m base approximation evaluations is extracted. Hence, the attacker gets an empirical probability distribution for the m bits vector. Actually, this distribution depends on the key used for encryption (usually it depends on m bits of this key). Using the correlations of the 2^m approximations, the probability distribution of the m bits vector can be computed for each possible key. Using enough pairs, the empirical distribution is likely to be the closest to the distribution provided by the correct key. The guessed key is the one with the maximum log-likelihood ratio (LLR) to the uniform distribution.

As the statistical independence hypothesis for linear approximations may not hold for many ciphers, this is an important theoretic improvement. Nevertheless, some of the results are not tight because of some other conjectures or simplifications. For instance, saying that the LLR of a wrong keys has a mean of 0 gives very pessimistic results as supposing statistical independence of LLRs does (for 8-round DES at least). Moreover, this method may not apply to some cryptanalyses (the one presented in this paper for instance). Using m base approximations leads to a time complexity of $2^m 2^d$ in the analysis phase (where d is the number of information bits to recover). In the case of the presented attack, 32968 approximations are used to recover 42 key bits. This set of approximation has a dimension of 54. Hence, the analysis time complexity is about 2^{96} what is much greater than exhaustive search. The approach presented in this paper is based on a statistical independence hypothesis. Thus, it is an orthogonal and complementary approach to the one of [14]. This approach leads to an attack with a better complexity than Matsui's algorithm 2 as soon as less than 2^{42} pairs are available (see Section 4). Using the same approximations in the framework of Hermelin and al. leads to an attack with higher complexity.

Actually, our method is based on some decoding techniques that are easily practicable in case of statistical independence of the approximations. That is why our theoretical framework seems to be the more suitable in that case. In the other hand, the work of Hermelin and al. is the more suitable when no assumption is made on statistical independence up to now.

2 The Probabilistic Model

It will be convenient to denote by $\tilde{\mathbf{K}} \stackrel{\text{def}}{=} (\tilde{K}_i)_{1 \leq i \leq n}$ the vector of linear combinations of the key bits induced by the key masks, that is

$$\tilde{K}_i \stackrel{\text{def}}{=} \bigoplus_{j=1}^k \kappa_i^j K_j.$$

A quantity will play a fundamental role in this setting : the dimension (what we will denote by d) of the vector space generated by the κ_i 's. It can be much smaller than the number n of different key masks.

We denote by Σ the set of N plaintext-ciphertext pairs. The information available after the distillation phase is modeled by

Model 1 — *The attacker receives a vector $\mathbf{Y} = (Y_i)_{1 \leq i \leq n}$ such that:*

$$\forall i \in \{1, \dots, n\}, \quad Y_i = (-1)^{\tilde{K}_i} + N_i \quad , \quad N_i \sim \mathcal{N}(0, \sigma_i^2), \quad (2)$$

where $\sigma_i^2 \stackrel{\text{def}}{=} \frac{1}{4N\epsilon_i^2}$ (N is the number of available plaintext/ciphertext pairs).

We denote by $f(\mathbf{Y} | \tilde{\mathbf{K}})$ the density function of the variable \mathbf{Y} conditioned by the value taken by $\tilde{\mathbf{K}}$ and $f_i(Y_i | \tilde{K}_i)$ denotes the density of the variable Y_i conditioned by \tilde{K}_i .

These conditional densities satisfy the independence relation

$$f(\mathbf{Y} | \tilde{\mathbf{K}}) = \prod_{i=1}^n f(Y_i | \tilde{K}_i) \quad (3)$$

The vector \mathbf{Y} is derived from Σ as follows. We first define for every i in $\{1, \dots, n\}$ and every j in $\{1, \dots, N\}$ the following quantity

$$D_i^j \stackrel{\text{def}}{=} < \pi_i, \mathbf{P}^j > \oplus < \gamma_i, \mathbf{C}^j > \oplus b_i,$$

where the plaintext-ciphertext pairs in Σ are indexed by $(\mathbf{P}^j, \mathbf{C}^j)$ and b_i is the constant appearing in the i -th linear approximation. Then for all i in $\{1, \dots, n\}$ we set up the counters D_i with $D_i \stackrel{\text{def}}{=} \sum_{j=1}^N D_i^j$ from which we build the vector of counters $\mathbf{D} = (D_i)_{1 \leq i \leq n}$. D_i is a binomial random variable which is approximately distributed as a normal law $\mathcal{N}((1/2 - \epsilon_i(-1)^{\tilde{K}_i})N, (1/4 - \epsilon_i^2)N)$. This explains why the vector $\mathbf{Y} = (Y_i)_{1 \leq i \leq n}$ is defined as:

$$Y_i \stackrel{\text{def}}{=} \frac{N - 2D_i}{2N\epsilon_i} \quad (4)$$

and why Equation (2) holds. There is some debate about the independence relation (3). This point is discussed by Murphy in [17] where he proves that even if some key masks are linearly dependent, the independence relation (3) holds asymptotically if for a fixed key the covariances $\text{cov}(D_{i_1}^j, D_{i_2}^j)$ are negligible. We have checked whether this holds in our experimental study. We had 129 linear approximations on 8-round DES with biases in the range $[1.45 \cdot 10^{-4}, 5.96 \cdot 10^{-4}]$ and we found empirical covariances in the range $[-2 \cdot 10^{-7}, 2 \cdot 10^{-7}]$ for 10^{12} samples. This corroborates the fact that the covariances are negligible and that the independence relation (3) approximately holds.

3 Bounds on the Required Amount of Plaintext-Ciphertext Pairs

3.1 An Information-Theoretic Lower Bound

The purpose of this subsection is to derive a general lower bound on the amount of uncertainty $\mathcal{H}(\mathbf{K} | \mathbf{Y})$ we have on the key given the statistics \mathbf{Y} derived from

the plaintext-ciphertext pairs. We recall that the (binary) *entropy* $\mathcal{H}(X)$ of a random variable X is given by the expression:

$$\begin{aligned}\mathcal{H}(X) &\stackrel{\text{def}}{=} -\sum_x \mathbf{Pr}(X = x) \log_2 \mathbf{Pr}(X = x) \text{ (for discrete } X) \\ &\stackrel{\text{def}}{=} -\int f(x) \log_2 f(x) dx \text{ (for continuous } X \text{ of density } f)\end{aligned}\quad (5)$$

For a couple of random variables (X, Y) we denote by $\mathcal{H}(X|Y)$ the *conditional entropy of X given Y* . It is defined by

$$\mathcal{H}(X|Y) \stackrel{\text{def}}{=} \sum_y \mathbf{Pr}(Y = y) \mathcal{H}(X|Y = y),$$

where $\mathcal{H}(X|Y = y) \stackrel{\text{def}}{=} -\sum_x \mathbf{Pr}(X = x|Y = y) \log_2 \mathbf{Pr}(X = x|Y = y)$ when X and Y are discrete variables and when \mathbf{Y} is a continuous random variable taking its values over \mathbb{R}^n it is given by

$$\mathcal{H}(X|\mathbf{Y}) = \int_{\mathbb{R}^n} \mathcal{H}(X|\mathbf{Y} = \mathbf{y}) f(\mathbf{y}) d\mathbf{y},$$

where $f(\mathbf{y})$ is the density of the distribution of \mathbf{Y} at the point \mathbf{y} . A related quantity is the *mutual information* $\mathcal{I}(X; Y)$ between X and Y which is defined by

$$\mathcal{I}(X; Y) \stackrel{\text{def}}{=} \mathcal{H}(X) - \mathcal{H}(X|Y). \quad (6)$$

It is straightforward to check [18] that this quantity is symmetric and that

$$\mathcal{I}(X; Y) = \mathcal{I}(Y; X) = \mathcal{H}(Y) - \mathcal{H}(Y|X). \quad (7)$$

Since K is a discrete random variable and Y is a continuous one, it will be convenient to use the following formula for the mutual information where the conditional distributions of Y given K has density $f(Y|K)$.

$$\mathcal{I}(K; Y) = \sum_k \mathbf{Pr}(K = k) \int f(y|k) \log \frac{f(y|k)}{\sum_k f(y|k)} dy. \quad (8)$$

We will be interested in deriving a lower bound on $\mathcal{H}(\mathbf{K}'|\mathbf{Y})$ when $\mathbf{K}' = (K'_1, \dots, K'_n)$ is a subkey derived from \mathbf{K} which satisfies:

(i) (conditional independence assumption)

$$f(\mathbf{Y} | \mathbf{K}') = \prod_{i=1}^n f(Y_i | K'_i), \quad (9)$$

where $f(\mathbf{Y}|\mathbf{K}')$ is the density function of the variable \mathbf{Y} conditioned by the value taken by \mathbf{K}' and $f_i(Y_i | K'_i)$ denotes the density of the variable Y_i conditioned by K'_i .

(ii) The subkey \mathbf{K}' may take $2^{k'}$ values and all are equally likely.

With these assumptions we have the following result

Lemma 1

$$\mathcal{I}(\mathbf{K}'; \mathbf{Y}) \leq \sum_{i=1}^n \mathcal{I}(K'_i; Y_i) \quad (10)$$

$$\mathcal{H}(\mathbf{K}' | \mathbf{Y}) \geq k' - \sum_{i=1}^n \mathcal{I}(K'_i; Y_i). \quad (11)$$

The proof of this lemma can be found in the appendix. It will be used in what follows in various scenarios for linear attacks, but it can obviously be used to cover many other cryptographic attacks. This lower bound is in general quite sharp as long as it is non-trivial, i.e when $k' \geq \sum_{i=1}^n \mathcal{I}(K'_i; Y_i)$. We will prove this for Attack 1 in what follows but this can also be done for the other cases.

3.2 Application to Various Scenarios

Attack 1: In this case, we do not use the linear equations as distinguishers but only want to recover the $\langle \kappa_i, \mathbf{K} \rangle$'s. This corresponds in the case of a single equation to Matsui's attack 1 and in the case of multiple equations to the attack MK1 in [11]. We have here

$$\begin{aligned} K'_i &= \tilde{K}_i = \langle \kappa_i, \mathbf{K} \rangle \\ Y_i &= \frac{N - 2D_i}{2N\epsilon_i}. \end{aligned}$$

Variables \mathbf{K}' and \mathbf{Y} satisfy the required conditional independence assumption (see Equation 3) and a straightforward calculation using Formula (8) yields

$$\mathcal{I}(K'_i; Y_i) = \mathbf{Cap}(\sigma_i^2)$$

where

$$\mathbf{Cap}(\sigma^2) \stackrel{\text{def}}{=} 1 - \frac{\sigma e^{-\frac{1}{2\sigma^2}}}{\sqrt{8\pi}} \int_{-\infty}^{\infty} e^{-\frac{u^2\sigma^2}{8}} e^{\frac{u}{2}} \log_2(1 + e^{-u}) du.$$

and therefore by applying Lemma 1 we obtain

$$\mathcal{H}(\mathbf{K}' | \mathbf{Y}) \geq d - \sum_{i=1}^n \mathbf{Cap}(\sigma_i^2) \quad (12)$$

Attack 2: This attack corresponds to cryptanalyses using a distinguisher such as [12,19]. Approximations of the form (1) are applied to a reduced cipher say the cipher peeled off by the first and the last round what is usually the case. Here, we focus on the subkeys used for the first ($\mathbf{K}_{\text{first}}$) and the last rounds (\mathbf{K}_{last}). The idea is to encrypt and decrypt the pairs with each possible value for $\mathbf{K}_{\text{first}}$ and \mathbf{K}_{last} and then to observe the bias obtained. The candidate that gives the greater bias is then chosen. Notice that we do not take care of the information given by the $\langle \kappa_i, \mathbf{K} \rangle$. This may be the case when cryptanalyzing ciphers for which it is difficult to find the key masks of linear approximations.

The $\langle \pi_i, \mathbf{P} \rangle$'s and the $\langle \gamma_i, \mathbf{C} \rangle$'s might not depend on all the bits of $\mathbf{K}_{\text{first}}$ and \mathbf{K}_{last} . We denote by $\hat{\mathbf{K}}_i$ the vector composed of the bits of $\mathbf{K}_{\text{first}}$ and \mathbf{K}_{last} on which the $\langle \pi_i, \mathbf{P} \rangle$'s and the $\langle \gamma_i, \mathbf{C} \rangle$'s depend on. We define \mathbf{K}' by the vector $(\hat{\mathbf{K}}_i)_{i=1}^n$ and assume that it may take $2^{\hat{k}}$ values. The aim is to recover \mathbf{K}' based on the values of the counters D_i^z for i in $\{1, \dots, n\}$ and z ranging over all possible values for \mathbf{K}' . These counters are defined similarly as in Section 2 with the difference being that we use the value $\mathbf{K}' = z$ for deriving the relevant couples (\mathbf{P}, \mathbf{C}) . The statistics $\mathbf{Y} = (Y_i)_{1 \leq i \leq n}$ we consider in this case is given by $Y_i \stackrel{\text{def}}{=} (Y_i^z)_z$ with

$$Y_i^z = \frac{|N - 2D_i^z|}{2N\epsilon_i}.$$

The conditional independence relation (3) is also satisfied in this case. With the help of Lemma 1, we can write $\mathcal{H}(\mathbf{K}'|\mathbf{Y}) \geq \hat{k} - \sum_{i=1}^n \mathcal{I}(K'_i; Y_i)$. We can again use Lemma 1 and obtain $\mathcal{I}(K'_i; Y_i) \leq \sum_z \mathcal{I}(K'_i; Y_i^z)$. The variable Y_i^z has density r_i if z corresponds to the right choice for K' and w_i otherwise, where $r_i(t) = \varphi_i^1(t) + \varphi_i^{-1}(t)$, $w_i(t) = 2\varphi_i^0(t)$ for nonnegative t with $\varphi_i^\alpha(t) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[-\frac{(t-\alpha)^2}{2\sigma_i^2}\right]$ being the density of a normal variable of expectation α and variance σ_i^2 . A straightforward application of Formula (8) gives

$$\mathcal{I}(K'_i; Y_i^z) = \int_0^\infty \frac{r_i(t)}{2^{\hat{k}}} \log\left(\frac{r_i(t)}{s_i(t)}\right) dt + \int_0^\infty (1 - 2^{-\hat{k}})w_i(t) \log\left(\frac{w_i(t)}{s_i(t)}\right) dt, \quad (13)$$

with $s_i(t) \stackrel{\text{def}}{=} 2^{-\hat{k}}r_i(t) + (1 - 2^{-\hat{k}})w_i(t)$. We denote this quantity by I_i and we finally obtain

$$\mathcal{H}(\mathbf{K}'|\mathbf{Y}) \geq \hat{k} - 2^{\hat{k}} \sum_{i=1}^n I_i.$$

Attack 3: This corresponds to the attack MK2 in [11] which is a variation of the previously seen distinguisher attack. In this case, we wish to find simultaneously the $\hat{\mathbf{K}}_i$'s defined in Attack 2 and the vector $\tilde{\mathbf{K}}$ defined in Attack 1. In this case, we let $\mathbf{K}'_i = (\hat{\mathbf{K}}_i, \tilde{K}_i)$ and define $\mathbf{K}' \stackrel{\text{def}}{=} (\mathbf{K}'_i)_{1 \leq i \leq n}$. We assume that $2^{k'}$ is the number of all possible values for \mathbf{K}' and that $2^{\hat{k}}$ is the number of all possible values for $\hat{\mathbf{K}}$. Here, we define the relevant statistics $\mathbf{Y} = (Y_i)_{1 \leq i \leq n}$ by $Y_i = (Y_i^z)_z$ where z ranges over all possible values for $\hat{\mathbf{K}}$ and where

$$Y_i^z = \frac{N - 2D_i^z}{2N\epsilon_i}.$$

We have again the desired independence relation (3) and as in the previous example we can use Lemma 1 twice to obtain

$$\mathcal{H}(\mathbf{K}'|\mathbf{Y}) \geq k' - \sum_{i=1}^n \mathcal{I}(\mathbf{K}'_i; \mathbf{Y}_i) \geq k' - 2^{\hat{k}} \sum_{i=1}^n \mathcal{I}(\mathbf{K}'_i; Y_i^z)$$

A straightforward application of Formula (8) yields

$$\mathcal{I}(\mathbf{K}'_i; Y_i^z) = \int_{-\infty}^{\infty} \frac{\varphi_i^1(t)}{2^k} \log \left(\frac{\varphi_i^1(t)}{\psi_i(t)} \right) dt + \int_{-\infty}^{\infty} (1 - 2^{-\hat{k}}) \varphi_i^0(t) \log \left(\frac{\varphi_i^0(t)}{\psi_i(t)} \right) dt,$$

with $\psi_i(t) \stackrel{\text{def}}{=} (1 - 2^{-\hat{k}})\varphi_i^0(t) + 2^{-\hat{k}-1}[\varphi_i^{-1}(t) + \varphi_i^1(t)]$ and $\varphi_i^\alpha(t)$ defined as in Attack 2.

3.3 An Upper Bound

One might wonder whether or not the bounds given in the previous subsection are sharp or not. It is clear that these lower bounds become negative when the number of pairs is large enough and that they are worthless in this case (since entropy is always nonnegative). However in all three cases it can be proved that as long the bound is non trivial it is quite sharp. We will prove this for the lower-bound (12). Similar techniques can be used for the other bounds but it would be too long to include them in this paper. To prove that (12) is sharp we will consider the case when

$$\sum_{i=1}^n \mathbf{Cap}(\sigma_i^2) \approx d$$

If the lower-bound is sharp, one might be tempted to say that the conditional entropy of \mathbf{K}' given \mathbf{Y} should be close to 0 which would mean that \mathbf{K}' is determined from \mathbf{Y} with probability close to 1. This is of course not always true, but it is the case *for most choices* of the coefficients κ_i^j . To give a precise meaning to this statement we will first consider what happens when the κ_i^j 's are chosen at random.

Theorem 1. *Assume that the κ_i^j are chosen chosen uniformly at random and that $\sum_{i=1}^n \mathbf{Cap}(\sigma_i^2) \geq d + \delta n$ for some constant $\delta > 0$. Let P_{err} be the probability that the most likely value for \mathbf{K}' given \mathbf{Y} is not the right one. There exists a constant A such that*

$$P_{\text{err}} \leq \frac{A}{\delta^2 n} + 2^{-\delta n/2}.$$

The probability P_{err} is taken over \mathbf{Y} but also over the choices of the κ_i^j 's. It says nothing about a particular choice of the κ_i^j 's. However it implies the aforementioned assertion about most choices of the κ_i 's. Let us be more specific by bringing in $P_{\text{err}}(\mathcal{C})$ which is the probability that the most likely key given \mathbf{Y} is not the right one when the subspace of dimension d of the possible values for $\tilde{\mathbf{K}}$ is \mathcal{C} . A bound on P_{err} implies that for most choices of the κ_i 's (and hence of \mathcal{C}) $P_{\text{err}}(\mathcal{C})$ is small by using the following lemma

Lemma 2. *Assume that $P_{\text{err}} \leq \epsilon$. Then for any $t > 0$:*

$$\Pr_{\mathcal{C}} (P_{\text{err}}(\mathcal{C}) \geq t\epsilon) \leq \frac{1}{t}$$

Proof. Let us define $P \stackrel{\text{def}}{=} \Pr_{\mathcal{C}}(P_{\text{err}}(\mathcal{C}) \geq t\epsilon)$. Then, we observe that $P_{\text{err}} = \sum_{\mathcal{C}} P_{\text{err}}(\mathcal{C}) \Pr(\mathcal{C}) \geq Pt\epsilon$. This implies that $P \leq \frac{1}{t}$. ■

Remark: The notation $\Pr_{\mathcal{C}}$ means here that the probability is taken over the choices for \mathcal{C} . It actually denotes the proportion of choices for \mathcal{C} which lead to the specified event inside the probability.

3.4 Entropy vs. Expected Number of $\tilde{\mathbf{K}}$'s More Likely Than the Right One

The aim of this subsection is to emphasize the fact that in a certain range of values of N (which is the number of plaintext-ciphertext pairs) the expected size \mathcal{E} of the list of the $\tilde{\mathbf{K}}$'s which are more likely than the right one gives pessimistic estimates of the amount of plaintext-ciphertext pairs we need to mount an attack. Actually, the *gain* g of a type 1 attack defined in [11] relies on this statistic \mathcal{E} . Here, we compare this *gain* with the capacity defined in Subsection 3.2. In order to achieve top ranking for a d -bits key (that is the correct key is at the top of the list), the gain has to be equal to d and Theorem 1 shows that for $\sum_{i=1}^n \mathbf{Cap}(\sigma_i^2) \approx d$ the probability of top ranking is close to 1. The comparison shows that the estimate derived from \mathcal{E} is twice bigger than the one derived from our entropy approach, as stated in Proposition 3.1. Proposition 3.1 holds for $N \cdot \epsilon_i^2 = o(1)$. This is often the case in multiple linear cryptanalysis where many approximations are used to drop the data complexity below the value required for a single approximation, that is $N = O(\epsilon^{-2})$.

Proposition 3.1 — Suppose that N is in a range where $\forall i, N\epsilon_i^2 = o(1)$. Using our entropy approach, the estimate for the data complexity required to achieve top ranking on a d -bit key is

$$N \approx \frac{d \ln(2)}{2 \sum_{i=1}^n \epsilon_i^2} (1 + o(1)).$$

The one obtained using the formula derived from the gain in [11] is

$$N \approx \frac{d \ln(2)}{\sum_{i=1}^n \epsilon_i^2} (1 + o(1)).$$

Proof.

It can be found in [20, ex. 4.12] that $\sum_{i=1}^n \mathbf{Cap}(\sigma_i^2) = \frac{2N \sum_{i=1}^n \epsilon_i^2}{\ln(2)} (1 + o(1))$, if for all $i, N \cdot \epsilon_i^2 = o(1)$. The corresponding estimate for N is $N \approx \frac{d \ln(2)}{2 \sum_{i=1}^n \epsilon_i^2} (1 + o(1))$. The formula for the gain in [11] is:

$$g \approx -\log_2 \left[2 \cdot \Phi \left(-\sqrt{2N \cdot \sum_{i=1}^n \epsilon_i^2} \right) \right]. \quad (14)$$

The following estimate can be found in [21, p. 175]. For large x , $\ln(\Phi(-x)) = -x^2/2(1 + o(1))$. We can apply this to (14) and find $N \approx \frac{d \ln(2)}{\sum_{i=1}^n \epsilon_i^2} (1 + o(1))$. ■

4 Experimental Results

To corroborate the theoretical results presented in this paper, we performed some experiments. First, we confirm the tightness of the bound on entropy by comparing it to the empirical entropy computed for a toy example (namely a type 1 attack on the 8-round DES). Then, we performed a realistic type 1 attack on the full 16-round DES, ranked the subkeys with respect to their likelihoods and checked whether or not the rank of the right subkey is among the $2^{\mathcal{H}(\mathbf{K}'|\mathbf{Y})}$ most likely subkeys. The results we obtained confirmed that choosing lists of this size is indeed relevant. Finally, in order to emphasize the power of multiple linear cryptanalysis, we compare this type 1 attack using many approximations to Matsui's type 3 attack using the optimal ranking statistic suggested by Junod [15]. This is first time that such an attack is performed.

Accuracy of the bound on entropy

Concerning the bound on entropy given in 1, we checked our results on 8-round DES. For those simulations, we used a group of 76 linear approximations involving 13 key bits to perform a type 1 attack. The quality of the lower-bound (12) can be verified by estimating empirically the entropy. Figure 1 displays the empirical conditional entropy of \mathbf{K}' given \mathbf{Y} for these equations as a function of $\log_2(N)$, where N is the number of available plaintext-ciphertext pairs. There is an excellent agreement between the lower bound and the empirical entropies up

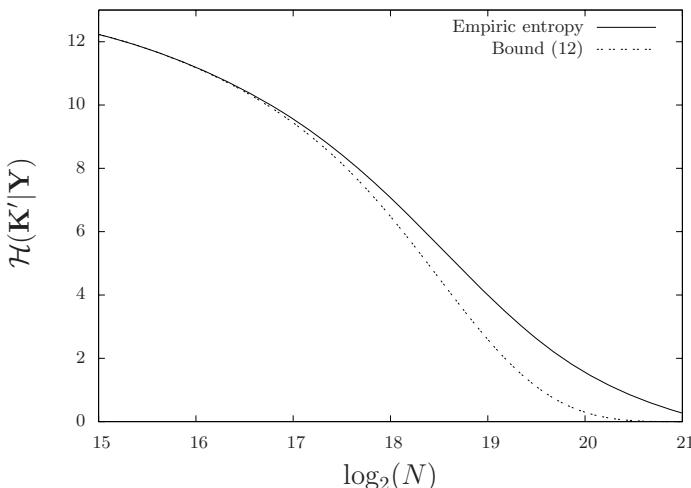


Fig. 1. Comparison between lower bound and empirical value of entropy

to when we approach the critical value of N for which the lower bound is equal to zero. This kind of lower bound is really suited to the case when the amount of plaintext/ciphertext pairs is some order of magnitude below this critical value. This is typically the case when we want to decrease the amount of data needed at the expense of keeping a list of possible candidates for \mathbf{K}' .

A realistic type 1 attack on the full 16-round DES

Our aim in this experiment was to confirm that most of the time the right value of $\mathbf{K}' = (\kappa_i, \mathbf{K})_{1 \leq i \leq n}$ belongs to the list of $2^{\mathcal{H}(\mathbf{K}'|\mathbf{Y})}$ most likely candidates. We performed here the whole type 1 attack, with the exception of the search phase which is not relevant for our purpose. In [22], the analysis phase uses a soft decision decoding algorithm for Reed Muller codes over the Gaussian channel with erasures. This decoding algorithm can be efficiently performed using a fast Walsh-Hadamard transform. Generating the list and sorting it are thus two operations with the same complexity $O(d2^d)$ where d is the dimension of the space spanned by κ 's. This implies that to speed up the analysis phase, we have to use approximations that lead to a small d . In the case when the set of approximations does not have any structure, the analysis phase can be efficiently performed using a general decoding algorithm for random linear codes such as for instance the stochastic resonance decoding algorithm from Valembois [23]. There is still no proof of its complexity but it is quite simple to implement and actually efficient. The study of this decoding algorithm is out of the scope of this article but is a nice subject we wish to work on.

Using a Branch & Bound Algorithm, we found 74086 linear approximations on the 16-round DES with biases higher than $2^{-28.84}$ (the biases are obtained by using the piling-up lemma). The space spanned by these κ 's turned out to be 56. This is too much to use directly the fast Walsh-Hadamard transform. We choose to consider a subset of these approximations (32968 out of 74086 spanning a vector space of dimension 42) which can be divided in 4 groups, each of them consisting of key masks κ_i spanning a vector space of small dimension d . We sum-up information about these groups in Figure 2.

Group	N	Nb. input masks	Nb. output masks	d
G1	12384	1500	82	19
G2	12384	82	1500	19
G3	4100	64	82	13
G4	4100	82	64	13

Fig. 2. Characteristics of the groups of approximations

The symmetry comes from the fact that enciphering or deciphering with the DES is the same algorithm (using subkeys in reverse order). We observe that the number of different masks in a group is much lower than the number of approximations. This will help us in speeding up the distillation phase using a trick similar to the one mentioned in [24]. Notice that some key bits are common

to some groups. We performed a fast Walsh Hadamard transformation on each group separately and use a heuristic to combine the information for each group to compute the rank of the correct key inside the list of candidates sorted with respect to their likelihood. This is detailed in the Appendix of [26].

The number N of available pairs was chosen to be small enough so that we can generate the data and perform the distillation phase in reasonable time. On the other hand, if we want our experiments to be relevant, we must get at least 1 bit of information about the key. These considerations lead us to choose $N = 2^{39}$ for which we get 2 bit of information out of 42 on the subkey.

We performed the attack 19 times. This attack recovers 42 bits of the key. For 2^{39} pairs, the information on the key is of 2 bits. The entropy on the key is thus 40 bits. Our theoretical work suggests to take a list of size $2^{\mathcal{H}(\mathbf{K}'|\mathbf{Y})} = 2^{40}$ to have a good success probability. Our experiments corroborate this. The worst rank over the 19 experiments is $2^{40.88}$ and the rank exceeded 2^{40} in only 3 experiments out of 19. The (ordered) list of ranks for the 19 experiments is:

$$\begin{aligned} & 2^{31.34}, 2^{33.39}, 2^{34.65}, 2^{35.24}, 2^{36.56}, 2^{37.32}, 2^{37.72}, 2^{37.99}, 2^{38.11}, 2^{38.52}, 2^{38.97}, \\ & 2^{39.04}, 2^{39.19}, 2^{39.27}, 2^{39.53}, 2^{39.85}, 2^{40.28}, 2^{40.82}, 2^{40.88}. \end{aligned}$$

Comparison with Matsui's attack

The attack from [2] uses two approximations on 14-round DES with biases $1.19.2^{-21}$ in a type 3 attack. This kind of attack uses approximations with much better biases than a type 1 attack because they involve only 14 rounds instead of the full 16 rounds.

Despite this fact, we show here that the gap between 14-round approximations and 16-round approximations can be filled by using many approximations in type 1 attack. We demonstrate here that for a rather large range of number of plaintext/ciphertext pairs N , a type 1 attack has a better complexity than the best version Matsui's type 3 attack [15]. This is first time that such a result is shown on the full DES.

Figure 3 gives the formulas used to compute the complexity of the two attacks. Due to space constraints, we do not detail how we obtained the distillation and analysis phase complexities but they are essentially a direct application of the tricks of [24] and the work of [19]. We denote by ν the **XOR** operation complexity and θ the DES enciphering complexity (including key schedule). From the same amount of data, our attack obtains more information on the key. It improves the final search complexity at the cost of increasing the distillation phase complexity. To measure the gain of using a type 1 attack, we have to estimate the ratio ν/θ .

Attack	Distillation	Analysis	Search
Matsui's [15]	$N \cdot 2 \cdot 46 \cdot \nu$	$12 \cdot 2^{12} \cdot \nu$	$2^{\mathcal{H}(\mathbf{K}' \mathbf{Y})} \cdot \theta$
Our	$N \cdot (82 + 82 + 64 + 64) \cdot 44 \cdot \nu$	$2^{26} \cdot \nu$	$2^{\mathcal{H}(\mathbf{K}' \mathbf{Y})} \cdot \theta$

Fig. 3. Complexities of the different steps

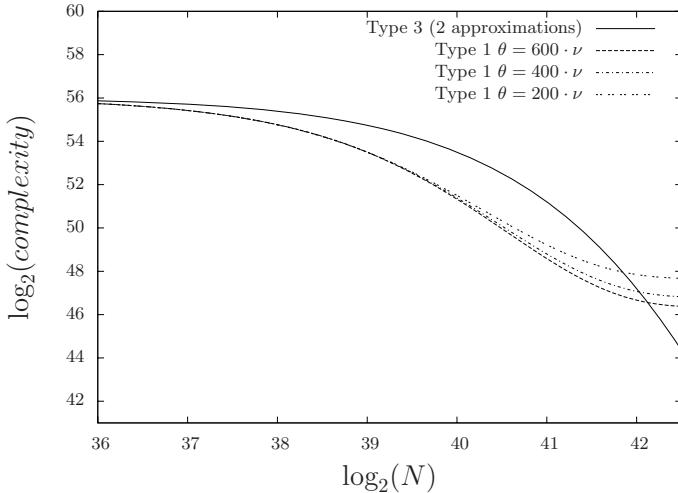


Fig. 4. Complexities of Matsui’s type 3 attack and our type 1 attack in terms of DES evaluations

The lower this ratio is, the more we gain using multilinear type 1 attack. For a standard implementation, $600 \cdot \nu$ is a good estimate of θ . We computed the complexities of the two attacks in terms of DES evaluations (θ) and plotted it as functions of the number of pairs in Figure 4. We restrict the plot to the value of N where type 1 attack competes with type 3 and we can see that this attack is better for N less than 2^{42} . We also plotted the complexities of the type attack for $\theta = 400 \cdot \nu$ and $\theta = 200 \cdot \nu$ to show that type 1 attack still competes with type 3 whenever enciphering is very efficient. Notice that with these estimates of θ the complexity for Matsui’s attack remains the same as long as N is less than 2^{42} .⁵

Remark on Matsui’s attack complexity

In [15], the author points out the fact that his Theorem 1 is pessimistic regarding the expected average rank of the good key. For 2^{43} pairs, the empirical complexity seems to be less than 2^{41} with high probability. Actually, the bound (12) suggests a complexity of precisely 2^{41} in this case (see Figure 4). This is a good illustration of the phenomenon mentioned in Section 3.4. The average rank of the good key is pessimistic because in some extremely rare cases the rank is sufficiently high to influence the mean. This observation, together with the complexity of computing multidimensional probability laws in a general case, may confirm the interest of the approach presented in this paper.

5 Conclusion and Further Work

We have presented here a rather general technique in Lemma 1 to derive a sharp lower bound on the entropy of a key given (independent) statistics.

We have applied it here to various linear cryptanalytic attacks, but the scope of this tool is much broader and it would be interesting to apply it for other classes of statistical attacks.

We performed a realistic type 1 attack on full 16-round DES using 32968 approximations and 2^{39} plaintext/ciphertext pairs that confirmed our theoretical results.

Moreover, theoretical results predict that for 2^{41} pairs, the DES can be broken with high probability with complexity close to 2^{48} while Matsui's attack 2 needs $2^{51.9}$ DES computations.

This work entails some further research interests.

It would be interesting to compare our theoretical results with some others [11,25] for some particular type 3 attack.

Another interesting thing would be to perform a type 1 attack on another cipher (SERPENT for instance) to see if, for recent ciphers, type 1 attacks still can compete type 3 attacks.

A deep study of different decoding algorithms for the analysis phase is necessary as much as a precise complexity analysis of distillation phase complexity for type 1 attack (maybe using ideas from [19]).

References

1. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, 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*. LNS, vol. 839, pp. 1–11. Springer, Heidelberg (1994)
3. Tardy-Corfdir, A., Gilbert, H.: A Known Plaintext Attack of FEAL-4 and FEAL-6. In: Feigenbaum, J. (ed.) *CRYPTO 1991*. LNCS, vol. 576, pp. 172–181. Springer, Heidelberg (1992)
4. Matsui, M., Yamagishi, A.: A New Method for Known Plaintext Attack of FEAL Cipher. In: Rueppel, R.A. (ed.) *EUROCRYPT 1992*. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993)
5. Ohta, K., Aoki, K.: Linear Cryptanalysis of the Fast Data Encipherment Algorithm. In: Desmedt, Y.G. (ed.) *CRYPTO 1994*. LNCS, vol. 839, pp. 12–16. Springer, Heidelberg (1994)
6. Tokita, T., Sorimachi, T., Matsui, M.: Linear Cryptanalysis of LOKI and s2DES. In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) *ASIACRYPT 1994*. LNCS, vol. 917, pp. 293–303. Springer, Heidelberg (1995)
7. Murphy, S., Piper, F., Walker, M., Wild, P.: Likelihood Estimation for Block Cipher Keys. Technical report, Information Security Group, University of London, England (1995)
8. Vaudenay, S.: An Experiment on DES Statistical Cryptanalysis. In: CCS 1996, pp. 139–147. ACM, New York (1996)
9. Junod, P., Vaudenay, S.: Optimal key ranking procedures in a statistical cryptanalysis. In: Johansson, T. (ed.) *FSE 2003*. LNCS, vol. 2887, pp. 235–246. Springer, Heidelberg (2003)
10. Kaliski, B.S., Robshaw, M.J.B.: Linear Cryptanalysis Using Multiple Approximations. In: Desmedt, Y.G. (ed.) *CRYPTO 1994*. LNCS, vol. 839, pp. 26–39. Springer, Heidelberg (1994)

11. Biryukov, A., Cannière, C.D., Quisquater, M.: On Multiple Linear Approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004)
12. 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)
13. Collard, B., Standaert, F.X., Quisquater, J.J.: Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 382–397. Springer, Heidelberg (2008)
14. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional Linear Cryptanalysis of Reduced Round Serpent. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 203–215. Springer, Heidelberg (2008)
15. Junod, P.: On the Complexity of Matsui’s Attack. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 199–211. Springer, Heidelberg (2001)
16. Selçuk, A.: On Probability of Success in Linear and Differential Cryptanalysis. *J. Cryptol.* 21, 131–147 (2008)
17. Murphy, S.: The Independence of Linear Approximations in Symmetric Cryptology. *IEEE Transactions on Information Theory* 52, 5510–5518 (2006)
18. Cover, T., Thomas, J.: Information theory. Wiley series in communications. Wiley, Chichester (1991)
19. 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)
20. Richardson, T., Urbanke, R.: Modern coding theory (2008)
21. Feller, W.: An introduction to probability theory and its applications, 3rd edn., vol. 1. John Wiley and Sons Inc, New York (1968)
22. Fourquet, R., Loidreau, P., Tavernier, C.: Finding Good Linear Approximations of Block Ciphers and its Application to Cryptanalysis of Reduced Round DES. In: WCC 2009, pp. 501–515 (2009)
23. Valembois, A.: Détection, Reconnaissance et Décodage des Codes Linéaires Binaires. PhD thesis, Université de Limoges (2000)
24. Biham, E., Dunkelman, O., Keller, N.: Linear Cryptanalysis of Reduced Round Serpent. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 219–238. Springer, Heidelberg (2002)
25. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional Extension of Matsui’s Algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
26. Gérard, B., Tillich, J.-P.: On Linear Cryptanalysis with Many Linear Approximations (full version). Cryptology ePrint Archive, Report 2009/463 (2009), <http://eprint.iacr.org/>

A Proofs

A.1 Proof of Lemma 1

Let us use Equation (7) and write in two different ways the mutual information between \mathbf{K}' and \mathbf{Y} : $\mathcal{I}(\mathbf{K}'; \mathbf{Y}) = \mathcal{H}(\mathbf{K}') - \mathcal{H}(\mathbf{K}'|\mathbf{Y}) = \mathcal{H}(\mathbf{Y}) - \mathcal{H}(\mathbf{Y}|\mathbf{K}')$. From this we deduce that

$$\begin{aligned} \mathcal{I}(\mathbf{K}'; \mathbf{Y}) &= \mathcal{H}(\mathbf{Y}) - \mathcal{H}(\mathbf{Y}|\mathbf{K}') \\ &= \mathcal{H}(Y_1, \dots, Y_n) - \mathcal{H}(Y_1, \dots, Y_n|\mathbf{K}'). \end{aligned} \tag{15}$$

Here Equation (15) is a consequence of the fact that the a priori distribution over \mathbf{K}' is the uniform distribution and the entropy of a discrete random variable which is uniformly distributed is obviously nothing but the logarithm of the number of values it can take. Moreover (see [18, Theorem 2.6.6])

$$\mathcal{H}(Y_1, \dots, Y_n) \leq \mathcal{H}(Y_1) + \dots + \mathcal{H}(Y_n). \quad (16)$$

On the other hand, by the chain rule for entropy [18, Theorem 2.5.1]:

$$\mathcal{H}(Y_1, \dots, Y_n | \mathbf{K}') = \mathcal{H}(Y_1 | \mathbf{K}') + \mathcal{H}(Y_2 | Y_1, \mathbf{K}') + \dots + \mathcal{H}(Y_n | Y_1, Y_2, \dots, Y_{n-1}, \mathbf{K}'). \quad (17)$$

We notice now that $\mathcal{H}(Y_i | \mathbf{K}', Y_1 \dots Y_{i-1})$ can be written as

$$\sum_{\mathbf{k}} \int_{\mathbb{R}^{i-1}} \mathcal{H}(Y_i | \mathbf{K}' = \mathbf{k}, Y_1 = y_1, \dots, Y_{i-1} = y_{i-1}) f(y_1, \dots, y_{i-1} | \mathbf{K}' = \mathbf{k}) \mathbf{Pr}(\mathbf{K}' = \mathbf{k}) dy_1 \dots dy_{i-1}, \quad (18)$$

where the sum is taken over all possible values \mathbf{k} of \mathbf{K}' and $f(y_1, \dots, y_{i-1} | \mathbf{K}' = \mathbf{k}) \mathbf{Pr}(\mathbf{K}' = \mathbf{k})$ is the density of the distribution of the vector (Y_1, \dots, Y_{i-1}) given the value \mathbf{k} of \mathbf{K}' at the point (y_1, \dots, y_{i-1}) . From conditional independence assumption (9) we deduce that $\mathcal{H}(Y_i | \mathbf{K}' = \mathbf{k}, Y_1 = y_1, \dots, Y_{i-1} = y_{i-1}) = \mathcal{H}(Y_i | K'_i)$. By summing in Expression (18) over y_1, \dots, y_{i-1} and all possible values of $K'_1, \dots, K'_{i-1}, K'_{i+1}, \dots, K'_n$ we obtain that

$$\mathcal{H}(Y_i | \mathbf{K}', Y_1, \dots, Y_{i-1}) = \frac{1}{2} \mathcal{H}(Y_i | K'_i = 0) + \frac{1}{2} \mathcal{H}(Y_i | K'_i = 1) = \mathcal{H}(Y_i | K'_i = k_i) \quad (19)$$

Plugging in this last expression in Expression (17) we obtain that

$$\mathcal{H}(Y_1, \dots, Y_n | K'_1, \dots, K'_n) = \mathcal{H}(Y_1 | K'_1) + \dots + \mathcal{H}(Y_n | K'_n). \quad (20)$$

Using this last equation and Inequality (16) in (15) we finally deduce that

$$\begin{aligned} \mathcal{I}(\mathbf{K}'; \mathbf{Y}) &\leq \mathcal{H}(Y_1) + \dots + \mathcal{H}(Y_n) - \mathcal{H}(Y_1 | K'_1) - \dots - \mathcal{H}(Y_n | K'_n) \\ &\leq \sum_{i=1}^n \mathcal{H}(Y_i) - \mathcal{H}(Y_i | K'_i) \leq \sum_{i=1}^n \mathcal{I}(K'_i; Y_i). \end{aligned} \quad (21)$$

The lower bound on the entropy follows from equality (7) that can be written as $\mathcal{H}(\mathbf{K}' | \mathbf{Y}) = \mathcal{H}(\mathbf{K}') - \mathcal{I}(\mathbf{K}'; \mathbf{Y}) = k' - \mathcal{I}(\mathbf{K}'; \mathbf{Y})$.

A.2 Proof of Theorem 1

The proof of this theorem follows closely standard proofs of the direct part of Shannon's channel capacity theorem [18], however most of the proofs given for this theorem are asymptotic in nature and are not suited to our case. There are proofs which are not asymptotic, but they are tailored for the case where all the σ_i 's are equal and are rather involved. We prefer to follow a slightly different path here. The first argument we will use is an explicit form of the joint AEP (Asymptotic Equipartition Property) theorem.

For this purpose, we denote by (\mathbf{X}, \mathbf{Y}) a couple of random variables where $\mathbf{X} = (X_i)_{1 \leq i \leq n}$ is uniformly distributed over $\{0, 1\}^n$ and $\mathbf{Y} = (Y_i)_{1 \leq i \leq n}$ is the

output of the Gaussian channel described in Section 2 when \mathbf{X} is sent through it. This means that

$$Y_i = (-1)^{X_i} + N_i, \quad (22)$$

where the N_i are independent centered normal variables of variance σ_i^2 .

Let us first bring in the following definition.

Definition 1. For $\epsilon > 0$, we define the set T_ϵ of ϵ -jointly typical sequences of $\{0, 1\}^n \times \mathbb{R}^n$ by $T_\epsilon \stackrel{\text{def}}{=} \bigcup_{\mathbf{x} \in \{0, 1\}^n} \{\mathbf{x}\} \times T_\epsilon(\mathbf{x})$ with

$$T_\epsilon(\mathbf{x}) \stackrel{\text{def}}{=} \{\mathbf{y} \in \mathbb{R}^n : |-\log_2(f(\mathbf{y})) - \mathcal{H}(\mathbf{Y})| < n\epsilon \quad (23)$$

$$|-\log_2(f(\mathbf{y}|\mathbf{x})2^{-n}) - \mathcal{H}(\mathbf{X}, \mathbf{Y})| < n\epsilon\} \quad (24)$$

where $f(\mathbf{y})$ is the density distribution of \mathbf{Y} and $f(\mathbf{y}|\mathbf{x})$ is the density distribution of \mathbf{Y} given that \mathbf{X} is equal to \mathbf{x} .

The entropies of \mathbf{Y} and (\mathbf{X}, \mathbf{Y}) are given by the following expressions

Lemma 3

$$\begin{aligned} \mathcal{H}(\mathbf{Y}) &= \sum_{i=1}^n \mathbf{Cap}(\sigma_i^2) + \frac{1}{2} \log_2(2\pi e \sigma_i^2) \\ \mathcal{H}(\mathbf{X}, \mathbf{Y}) &= n + \sum_{i=1}^n \frac{1}{2} \log_2(2\pi e \sigma_i^2) \end{aligned}$$

Proof. Notice that with our model the Y_i 's are independent. Therefore $\mathcal{H}(\mathbf{Y}) = \sum_{i=1}^n \mathcal{H}(Y_i)$. Moreover, by the very definitions of entropy and mutual information: $\mathcal{H}(Y_i) = \mathcal{H}(Y_i|X_i) + \mathcal{I}(X_i; Y_i)$; X_i is uniformly distributed over $\{0, 1\}$ and therefore by the definition of the capacity of a Gaussian channel and the fact that the capacity attains its maximum for a binary input which is uniformly distributed we have $\mathcal{I}(X_i; Y_i) = \mathbf{Cap}(\sigma_i^2)$. On the other hand $\mathcal{H}(Y_i|X_i)$ is obviously the same as $\mathcal{H}(N_i)$. The calculation of this entropy is standard (see [18]) and gives

$$\mathcal{H}(N_i) = \frac{1}{2} \log_2(2\pi e \sigma_i^2) \quad (25)$$

By putting all these facts together we obtain the expression for $\mathcal{H}(\mathbf{Y})$. Concerning the other entropy, with similar arguments we obtain

$$\begin{aligned} \mathcal{H}(\mathbf{X}, \mathbf{Y}) &= \mathcal{H}(\mathbf{X}) + \mathcal{H}(\mathbf{Y}|\mathbf{X}) \\ &= n + \sum_{\mathbf{x} \in \{0, 1\}^n} \frac{1}{2^n} \mathcal{H}(\mathbf{Y}|\mathbf{X} = \mathbf{x}) \\ &= n + \sum_{\mathbf{x} \in \{0, 1\}^n} \frac{1}{2^n} \mathcal{H}(N_1, \dots, N_n) \\ &= n + \sum_{i=1}^n \frac{1}{2} \log_2(2\pi e \sigma_i^2) \end{aligned}$$

■

T_ϵ ” stands for “typical set” since it is highly unlikely that (\mathbf{X}, \mathbf{Y}) does not belong to T_ϵ :

Lemma 4. *There exists a constant A such that*

$$\Pr((\mathbf{X}, \mathbf{Y}) \notin T_\epsilon) \leq \frac{A}{\epsilon^2 n}.$$

Before giving the proof of this lemma we will first give an interpretation of entropy which provides an explanation of why the probability of falling outside the typical set becomes smaller as n increases.

Lemma 5. *Let $U_i \stackrel{\text{def}}{=} -\log_2 f_i(Y_i)$ where f_i is given by*

$$f_i(y) \stackrel{\text{def}}{=} \frac{1}{2\sqrt{2\pi\sigma_i^2}} \left(e^{-\frac{(y-\mu)^2}{2\sigma_i^2}} + e^{-\frac{(y+\mu)^2}{2\sigma_i^2}} \right).$$

We also denote by $V_i \stackrel{\text{def}}{=} -\log_2 \left(\frac{g_i(Y_i - (-1)^{X_i})}{2} \right)$ where g_i is the density distribution of a centered Gaussian variable of variance σ_i^2 .

$$\begin{aligned} -\log_2(f(\mathbf{Y})) - \mathcal{H}(\mathbf{Y}) &= \sum_{i=1}^n U_i - \mathbb{E} \left(\sum_{i=1}^n U_i \right) \\ -\log_2(f(\mathbf{Y}|\mathbf{X})2^{-n}) - \mathcal{H}(\mathbf{X}, \mathbf{Y}) &= \sum_{i=1}^n V_i - \mathbb{E} \left(\sum_{i=1}^n V_i \right) \end{aligned}$$

Proof. For the first equation we just have to notice that

$$-\log_2(f(\mathbf{Y})) = -\log_2(\prod_{i=1}^n f_i(Y_i)) = -\sum_{i=1}^n \log_2(f_i(Y_i)) = \sum_{i=1}^n U_i$$

and that $\mathcal{H}(\mathbf{Y}) = \mathbb{E}(-\log_2 f(\mathbf{Y}))$, which follows directly from the definition of the entropy given in (5). The second equation can be obtained in a similar way. ■

This implies that in order to estimate the probability that a point falls outside the typical set we have to estimate the probability that the deviation between a sum of n independent random variables and its expectation is at least of order ϵn . In our case, it can be proven that for fixed ϵ , this probability is exponentially small in n . However, we prefer to give a much weaker statement which is also easier to prove and which uses only Chebyschev’s inequality, which we recall here

Lemma 6. *Consider a real random variable X of variance $\text{var}(X)$. We have for any $t > 0$:*

$$\Pr(|X - \mathbb{E}(X)| \geq t) \leq \frac{\text{var}(X)}{t^2}. \quad (26)$$

To use this inequality we have to estimate the variances of the U_i 's and the V_i 's. It can be checked that

Lemma 7. *There exists a constant A such that for any i we have*

$$\text{var}(V_i) \leq A \quad \text{and} \quad \text{var}(U_i) \leq A.$$

Proof. Can be found in [26]. ■

We are ready now to prove Lemma 4:

Proof. We start the proof by writing

$$\begin{aligned} \mathbf{Pr}((\mathbf{X}, \mathbf{Y}) \notin T_\epsilon) &= \mathbf{Pr}(|-\log_2(f(\mathbf{Y})) - \mathcal{H}(\mathbf{Y})| \geq n\epsilon) \cup \{|-\log_2(f(\mathbf{Y}|\mathbf{X})2^{-n}) - \mathcal{H}(\mathbf{X}, \mathbf{Y})| \geq n\epsilon\} \\ &\leq \mathbf{Pr}(|-\log_2(f(\mathbf{Y})) - \mathcal{H}(\mathbf{Y})| \geq n\epsilon) + \mathbf{Pr}(|-\log_2(f(\mathbf{Y}|\mathbf{X})2^{-n}) - \mathcal{H}(\mathbf{X}, \mathbf{Y})| \geq n\epsilon) \\ &= \mathbf{Pr}(|U - \mathbb{E}(U)| \geq n\epsilon) + \mathbf{Pr}(|V - \mathbb{E}(V)| \geq n\epsilon) \end{aligned}$$

with $U \stackrel{\text{def}}{=} \sum_{i=1}^n U_i$ and $V \stackrel{\text{def}}{=} \sum_{i=1}^n V_i$. We use now Chebyschev's inequality (Lemma 26) together with the upper-bounds $\text{var}(U) = \sum_{i=1}^n \text{var}(U_i) \leq nA$ and $\text{var}(V) = \sum_{i=1}^n \text{var}(V_i) \leq nA$ to obtain $\mathbf{Pr}((\mathbf{X}, \mathbf{Y}) \notin T_\epsilon) \leq \frac{2A}{n\epsilon^2}$. ■

Moreover, not only is it unlikely that (\mathbf{X}, \mathbf{Y}) does not fall in T_ϵ , but the Euclidean volume (which we denote by "Vol") of this set is not too large:

Lemma 8

$$\sum_{\mathbf{x} \in \{0,1\}^n} \text{Vol}(T_\epsilon(\mathbf{x})) \leq 2^{\mathcal{H}(\mathbf{X}, \mathbf{Y}) + \epsilon n}$$

Proof. Let us notice that

$$\begin{aligned} 1 &= \sum_{\mathbf{x} \in \{0,1\}^n} \frac{1}{2^n} \int_{\mathbb{R}^n} f(\mathbf{y}|\mathbf{x}) d\mathbf{y} \geq \sum_{\mathbf{x} \in \{0,1\}^n} \frac{1}{2^n} \int_{T_\epsilon(\mathbf{x})} f(\mathbf{y}|\mathbf{x}) d\mathbf{y} \\ &\geq \sum_{\mathbf{x} \in \{0,1\}^n} \text{Vol}(T_\epsilon(\mathbf{x})) 2^{-\mathcal{H}(\mathbf{X}, \mathbf{Y}) - \epsilon n} \end{aligned}$$

where the last inequality follows from (24) ■

We will use this result to show that

Proposition 1. *If $(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$ is a couple of independent random variables, where $\tilde{\mathbf{X}}$ is uniformly distributed and $\tilde{\mathbf{Y}}$ has the same distribution as \mathbf{Y} , then $\mathbf{Pr}((\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \in T_\epsilon) \leq 2^{-C+2n\epsilon}$ with $C \stackrel{\text{def}}{=} \sum_{i=1}^n \text{Cap}(\sigma_i^2)$.*

Proof. We evaluate $\mathbf{Pr}((\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \in T_\epsilon)$ as follows

$$\mathbf{Pr}((\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \in T_\epsilon) = \sum_{\mathbf{x} \in \{0,1\}^n} \frac{1}{2^n} \int_{T_{\mathbf{x}}(\epsilon)} f(\mathbf{y}) d\mathbf{y} \leq \sum_{\mathbf{x} \in \{0,1\}^n} \frac{1}{2^n} \text{Vol}(T_{\mathbf{x}}(\epsilon)) 2^{-\mathcal{H}(\mathbf{Y}) + \epsilon n}$$

The last inequality follows from (23) in the definition of the typical set. We use now Lemma 8 to obtain

$$\mathbf{Pr}((\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \in T_\epsilon) \leq \frac{1}{2^n} 2^{\mathcal{H}(\mathbf{X}, \mathbf{Y}) + \epsilon n} 2^{-\mathcal{H}(\mathbf{Y}) + \epsilon n} \leq 2^{-n + \mathcal{H}(\mathbf{X}; \mathbf{Y}) - \mathcal{H}(\mathbf{Y}) + 2\epsilon n}$$

By using the expressions for $\mathcal{H}(\mathbf{X}, \mathbf{Y})$ and $\mathcal{H}(\mathbf{Y})$ given in Lemma 3 we deduce $-n + \mathcal{H}(\mathbf{X}, \mathbf{Y}) - \mathcal{H}(\mathbf{Y}) = -\sum_{i=1}^n \mathbf{Cap}(\sigma_i^2)$. This finishes the proof. ■

These results can be used to analyze the following typical set decoder, which takes as inputs a vector \mathbf{y} in \mathbb{R}^n which is the output of the Gaussian channel described in Section 2 and a real parameter ϵ , and outputs either ‘‘Failure’’ or a possible key $\tilde{\mathbf{k}} \in \{0, 1\}^n$.

```
TYPICAL SET DECODER( $\mathbf{y}, \epsilon$ )
1 counter  $\leftarrow 0$ 
2 for all possible values  $\mathbf{k}$  of  $\tilde{\mathbf{K}}$ 
3   do if  $\mathbf{y} \in T_{\mathbf{k}}(\epsilon)$ 
4     then counter  $\leftarrow$  counter + 1
5     result  $\leftarrow \mathbf{k}$ 
6 if counter = 1
7   then return result
8   else return failure
```

This algorithm is therefore successful if and only if \mathbf{y} is in the typical set of the right key and if there is no other value \mathbf{k} for $\tilde{\mathbf{K}}$ for which \mathbf{y} belongs to the typical set associated to \mathbf{k} . Let us now finish the proof of Theorem 1.

Proof. Let \mathbf{k} be right value of $\tilde{\mathbf{K}}$ and let \mathcal{C} be the set of possible values of $\tilde{\mathbf{K}}$. The probability P_{err} that the typical decoder fails is clearly upper-bounded by

$$P_{\text{err}} \leq \mathbf{Pr}_{\mathbf{y}, \mathcal{C}}(\overline{T_{\mathbf{k}}(\epsilon)}) + \sum_{\mathbf{k}' \in \mathcal{C}, \mathbf{k}' \neq \mathbf{k}} \mathbf{Pr}_{\mathbf{y}, \mathcal{C}}(T_{\mathbf{k}'}(\epsilon)) \quad (27)$$

where $\overline{T_{\mathbf{k}}(\epsilon)}$ denotes the complementary set of $T_{\mathbf{k}}(\epsilon)$. On the one hand

$$\mathbf{Pr}_{\mathbf{y}, \mathcal{C}}(\overline{T_{\mathbf{k}}(\epsilon)}) = \mathbf{Pr}((\mathbf{X}, \mathbf{Y}) \notin T_{\epsilon}) \leq \frac{A}{\epsilon^{2n}}.$$

by Lemma 4, and on the other hand for $\mathbf{k}' \neq \mathbf{k}$:

$$\begin{aligned} \sum_{\mathbf{k}' \in \mathcal{C}, \mathbf{k}' \neq \mathbf{k}} \mathbf{Pr}_{\mathbf{y}, \mathcal{C}}(T_{\mathbf{k}'}(\epsilon)) &\leq \sum_{\mathbf{k}' \in \mathcal{C}} \mathbf{Pr}_{\mathbf{y}, \mathcal{C}}(T_{\mathbf{k}'}(\epsilon)) = 2^r \mathbf{Pr}\left((\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) \in T_{\epsilon}\right) \\ &\leq 2^{r - \sum_{i=1}^n \mathbf{Cap}(\sigma_i^2) + 2\epsilon n} \end{aligned}$$

by Proposition 1. By plugging in these two upper bounds in the union bound (27) we obtain $P_{\text{err}} \leq \frac{A}{\epsilon^{2n}} + 2^{r - \sum_{i=1}^n \mathbf{Cap}(\sigma_i^2) + 2\epsilon n} \leq \frac{A}{\epsilon^{2n}} + 2^{-\delta n + 2\epsilon n}$. We finish the proof by choosing $\epsilon = \frac{\delta}{4}$. ■

Bivium as a Mixed-Integer Linear Programming Problem

Julia Borghoff*, Lars R. Knudsen, and Mathias Stolpe

DTU Mathematics,
Technical University of Denmark
`{J.Borghoff,Lars.R.Knudsen,M.Stolpe}@mat.dtu.dk`

Abstract. Trivium is a stream cipher proposed for the eSTREAM project. Raddum introduced some reduced versions of Trivium, named Bivium A and Bivium B. In this article we present a numerical attack on the Biviums. The main idea is to transform the problem of solving a sparse system of quadratic equations over $GF(2)$ into a combinatorial optimization problem. We convert the Boolean equation system into an equation system over \mathbb{R} and formulate the problem of finding a 0-1-valued solution for the system as a mixed-integer programming problem. This enables us to make use of several algorithms in the field of combinatorial optimization in order to find a solution for the problem and recover the initial state of Bivium. In particular this gives us an attack on Bivium B in estimated time complexity of $2^{63.7}$ seconds. But this kind of attack is also applicable to other cryptographic algorithms.

1 Introduction

In 2004 the eSTREAM project within ECRYPT called for secure and fast stream ciphers. There were two categories: one for software oriented ciphers and one for hardware oriented ciphers. In the hardware category the focus was on stream ciphers for hardware applications with restricted resources such as limited storage, gate count or power consumption. In 2008 the eSTREAM project ended and amongst the recommended hardware oriented stream ciphers was Trivium. Due to its elegant design and simple structure (see Section 2 for details) Trivium has been a target for many cryptanalysts but so far no attack faster than exhaustive key search is known. In order to understand the structure of Trivium better and to invent possible attacks, Raddum introduced two reduced versions of Trivium called Bivium A and Bivium B [8]. These variants have been successfully attacked by an approach using SAT-solvers [3] and by an approach using special graphs [8]. One possible way to recover the initial state of Bivium is to solve a sparse system of quadratic and linear equations over $GF(2)$. It is known that solving a system of non-linear Boolean equations is an NP-hard problem. In this paper we introduce a new approach for solving this system of equations.

* The author is supported by a grant from the Danish research council for Technology and Production Sciences grant number 274-07-0246.

We convert the Boolean equations into equations which hold over the reals or respectively over the integers. Starting from this system we generate a system of linear equalities and inequalities which describes the initial state of Bivium. These constraints yield together with an arbitrary objective function a mixed-integer programming (MIP) problem and we can use methods of combinatorial optimization to solve the problem. Unfortunately it is difficult to give an estimate of the running time of these algorithms and it is hard to predict whether a problem is solvable in reasonable time or not because the running time might be exponential in the worst case [7]. For that reason it is necessary to ascertain the running times by experiments.

2 Description of Bivium

Trivium [4] is one of the hardware-oriented stream ciphers which are recommended by the eSTREAM project. Trivium has a 288 bit initial state which is divided into three nonlinear feedback shift registers (NFSR). The 80-bit key is loaded into the first register and an 80 bit IV into the second. After $4 \cdot 288$ clockings (steps) of the algorithm which do not produce any output, a key stream bit is produced by an XOR of six state bits (two from each register). Raddum [8] introduces two small-scale variants of Trivium called Bivium A and Bivium B (the latter is often referred to as Bivium). Both variants are obtained by removing the last NFSR which yields an internal state of size 177. The following pseudo code specifies how to generate one bit z of the key stream for Bivium B where s_i denotes the i th state bit:

$$\begin{aligned} t_1 &\leftarrow s_{66} + s_{93} \\ t_2 &\leftarrow s_{162} + s_{177} \\ z &\leftarrow t_1 + t_2 \\ t_1 &\leftarrow t_1 + s_{91}s_{92} + s_{171} \\ t_2 &\leftarrow t_2 + s_{175}s_{176} + s_{69} \\ (s_1, s_2, \dots, s_{93}) &\leftarrow (t_2, s_1, \dots, s_{92}) \\ (s_{94}, s_{95}, \dots, s_{177}) &\leftarrow (t_1, s_{94}, \dots, s_{176}) \end{aligned}$$

Bivium A only differs in the key stream bit equation from Bivium B. In Bivium A the key stream bit $z = t_2$ which means that the key stream bit only depends directly on the second register. It is important to note that not only is the state size smaller for Bivium compared to Trivium but also the linear equation or the key stream equation is simpler because it only depends on respectively four and two bits for Bivium B respectively Bivium A.

By introducing a new variable for each update of the NFSRs [8] we obtain a full description of an internal state after observing 177 key stream bits by equations of the form:

$$s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} = z \quad (1)$$

$$s_{66} \oplus s_{93} \oplus (s_{91} \wedge s_{92}) \oplus s_{171} \oplus s_{178} = 0 \quad (2)$$

$$s_{162} \oplus s_{177} \oplus (s_{175} \wedge s_{176}) \oplus s_{69} \oplus s_{179} = 0. \quad (3)$$

177 key stream bits yield a fully determined system of 399 equations in 399 unknowns. By using more than 177 key stream bits we can easily obtain an overdetermined system.

Solving a system of random quadratic equation over GF(2) is known to be an NP-hard problem. In order to find a solution despite this fact we will convert the problem into a problem over the reals or respectively the integers and use a method from the field of combinatorial optimization in order to solve the system. Here the small number of monomials per equation will be an advantage as we will see later.

3 Mixed-Integer Programming

Combinatorial and integer optimization deals with the problem of minimizing (or maximizing) a function of several variables subject to equality and inequality constraints and integrality restrictions on some or all of the variables. A linear mixed-integer programming problem (MIP) is a problem of the form

$$\min_x \{c^T x | Ax \leq b, x \in \mathbb{Z}^k \times \mathbb{R}^l\}$$

where c is an n -vector, A is an $m \times n$ -matrix ($n = k + l$) and b is an m -vector. This means that we minimize a linear function subject to linear equality and inequality constraints. Additionally some of the variables are restricted to integer values while the other variables are real-valued.

The set S of all $x \in \mathbb{Z}^k \times \mathbb{R}^l$ which satisfies the linear constraints $Ax \leq b$

$$S = \{x \in \mathbb{Z}^k \times \mathbb{R}^l, Ax \leq b\}$$

is called a feasible set. An element $x \in S$ is called a feasible point. The MIP is called feasible if $S \neq \emptyset$ and infeasible if $S = \emptyset$. The function $z = c^T x$ is the objective function that we want to minimize.

An MIP has either an optimal solution, is unbounded, or is infeasible. If there exists for any $w \in \mathbb{R}$ an $x \in S$ such that $c^T x < w$, the MIP is unbounded. A solution for a MIP is optimal if a feasible point $x_S \in S$ exists with $c^T x_S \leq c^T x$ for all $x \in S$.

Special cases of MIP are the linear programming problem (LP)

$$\min_x \{c^T x | Ax \leq b, x \in \mathbb{R}^l\}$$

where all variables are continuous and the pure integer programming problem (IP)

$$\min_x \{c^T x | Ax \leq b, x \in \mathbb{Z}^k\}$$

where all variables are integer-valued. We are mainly interested in another special case of MIP called 0-1-MIP where the integer variables are replaced by binary variables.

3.1 Modeling

Usually there is more than one way to model an MIP or an IP. In integer programming, formulating a 'good' model is of crucial importance for solving the problem [10]. The first question in formulating a model is usually defining variables but in the case of Bivium the obvious choice is to take the internal state bits as variables. We will also introduce some additional variables depending on the model we use. The second question is finding a good objective function. As we will see later we are interested in a feasible point and not in an optimal solution for our problem. Hence, we have a lot of freedom to choose the objective function. The main problem is to find a good formulation for $S = \{x \in \mathbb{Z}^k \times \mathbb{R}^l, Ax \leq b\}$. Here it is often easy to find A and b which yield a valid formulation for S but this description of the feasible set might not be the best one for actually solving the problem. The reason for this is that integer optimization algorithms such as branch-and-cut need a lower bound on the objective function and they determine this bound often by using relaxations. One possible relaxation of the problem is to solve the corresponding linear program $z_{LP} = \min_x \{c^T x : Ax \leq b, x \in \mathbb{R}^n\}$. The feasible set S of the MIP is a subset of the feasible set P of the LP ($S \subset P = \{x \in \mathbb{R}^n : Ax \leq b\}$). The smaller the set P is, the more precise are the bounds for the MIP and the efficiency of most algorithms is dependent on these bounds.

4 Conversion Methods

We are interested in using mixed-integer linear programming problems and the corresponding algorithms to solve the systems of equations we get from encryption schemes. These systems of equations usually describe the secret key or the internal state of the encryption scheme, that is, if we are able to solve the system of equations we have recovered the secret key or an internal state. But the equations we get are defined over $GF(2)$. The algorithms which are used to solve integer or mixed-integer linear programming problems work over the reals using some integrality restrictions. Therefore we want to represent the Boolean functions as polynomials over the reals. One method to do this is the Standard Conversion Method [2]:

Definition 1 (Standard Conversion)

The Standard Conversion is defined by a mapping $t : \{\text{FALSE}, \text{TRUE}\} \rightarrow \{0, 1\}$ with

$$t(s) = \begin{cases} 0 & \text{if } s = \text{FALSE} \\ 1 & \text{if } s = \text{TRUE} \end{cases}$$

and

$$\begin{aligned} s_1 \wedge s_2 &\Rightarrow x_1 \cdot x_2 \\ s_1 \oplus s_2 &\Rightarrow x_1 + x_2 - 2x_1x_2 \end{aligned}$$

where $x_i = t(s_i)$ for $i = 1, 2$.

A second approach is not to convert each operation but the equation as whole. This is called the Adapted Standard Conversion (ASC) [6]. We map $\{\text{FALSE}, \text{TRUE}\} \rightarrow \{0, 1\}$ and evaluate the equations over the reals without changing their structure (we just replace XOR by addition over the reals and AND by multiplication) for all values for which the Boolean equation holds. We will get different results and for each result we get an equation over the reals by subtracting the evaluation result from the left-hand side of the equation. It follows that only one of these new real-valued equations can be true. We multiply each equation by an exclusion variable and add the constraint that the sum of the exclusion variables is 1 to achieve this.

Example 1. We consider the equation

$$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 = 0.$$

This equation holds for all $(x_1, x_2, x_3, x_4, x_5) \in \{0, 1\}^5$ with even Hamming weight. By evaluating the function as a function over the reals we get as results 0, 2, 4. This means one of the following equations is true over the reals:

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 + x_5 &= 0, \\ x_1 + x_2 + x_3 + x_4 + x_5 - 2 &= 0, \\ x_1 + x_2 + x_3 + x_4 + x_5 - 4 &= 0. \end{aligned}$$

It is obvious that not all equations can be true at the same time, therefore we introduce exclusion variables x_{e_1}, x_{e_2} and x_{e_3} .

$$\begin{aligned} x_{e_1}(x_1 + x_2 + x_3 + x_4 + x_5) &= 0 \\ x_{e_2}(x_1 + x_2 + x_3 + x_4 + x_5 - 2) &= 0 \\ x_{e_3}(x_1 + x_2 + x_3 + x_4 + x_5 - 4) &= 0 \\ x_{e_1} + x_{e_2} + x_{e_3} &= 1 \end{aligned}$$

Using the ASC the basis structure of the equations can be kept, the degree increases by only one, but the number of variables and equations is increased by three in our example.

In the next section we will show how we can use the Standard Conversion method and the basic idea of the ASC method to convert the Boolean equation system of Bivium A into a system of linear real or integer equations respectively.

5 Bivium A as a Mixed-Integer Linear Programming Problem

In this section we will explain step by step how to model Bivium A as a mixed-integer linear programming problem. We will use the equations which describe

the initial state [8] to define the feasible set of the mixed-integer problem. This means that we formulate the problem of finding the internal state of Bivium A given a sufficiently long part of the key stream as a feasibility problem rather than as an optimization problem. As mentioned in Section 3.1 it is very important how to model the MIP. In the following subsections we will describe two different ideas how to model Bivium A as an MIP, one model is based on the Standard Conversion Method and the other one uses the idea of the Adapted Standard Conversion Method. Afterwards we will compare the results of these different models.

The starting point for both linearisations is the system of Boolean equations which describe Bivium A. For each clocking of the algorithm we get three equations of the form

$$s_{162} \oplus s_{177} = z \quad (4)$$

$$s_{66} \oplus s_{93} \oplus (s_{91} \wedge s_{92}) \oplus s_{171} \oplus s_{178} = 0 \quad (5)$$

$$s_{162} \oplus s_{177} \oplus (s_{175} \wedge s_{176}) \oplus s_{69} \oplus s_{179} = 0 \quad (6)$$

After 177 clockings of the algorithm we have a fully determined system of 399 equations in 399 unknowns. (For the last 66 clockings we do not need to introduce new variables, thus we do not need the quadratic equations). For each further clocking we get an overdetermined system because we get three equations and introduce two new variables [8].

We want to use this fully or overdetermined system to describe the feasible set of an MIP. We will show two ways to do this.

5.1 Linearisation Using the Standard Conversion Method

Using the Standard Conversion Method the degree of the real equation is equal to the number of variables involved in the Boolean equation. We want to keep the degree of the equation as small as possible because later we have to replace terms of higher order by new variables in order to get linear constraints. We also want to keep the number of monomials per equation small because the less complex the constraints are, the more likely it is that we can solve the MIP. Therefore the first step is to split up the equations by introducing some new auxiliary variables.

Splitting. First we replace $s_{162} \oplus s_{177}$ by the key stream bit z in the quadratic equation. This is a good opportunity to reduce the number of variables in this equation and the complexity of the resulting constraint. This is different from the case of Bivium B and Trivium because of the more complex key stream equation. Then we introduce new variables in such a way that there are not more than four variables per equation. Using a small trick of rewriting the equations we

can achieve that the real equations have at most degree two after applying the Standard Conversion Method. We split the system in the following way:

$$\begin{aligned}s_{162} \oplus s_{177} &= z \\r_1 &= s_{66} \oplus s_{93} \\r_2 &= s_{91} \wedge s_{92} \\r_1 \oplus r_2 &= s_{171} \oplus s_{178} \\r_3 &= s_{175} \wedge s_{176} \\z \oplus r_3 &= s_{69} \oplus s_{179}\end{aligned}$$

This means we introduce three splitting variables for each clocking (except for the last 66 clockings). Starting from a fully determined system with 399 equations this splitting yields a system of 732 equations in 732 unknowns. These are still Boolean equations. Hence the next step is to convert the system of Boolean equations into a system over the reals.

Conversion Using the Standard Conversion Method. The only requirement we have is that the solution of the Boolean system is also a solution of the real system. We can ignore the additional non-binary solutions of the real system. Thus it does not matter whether we convert each side of the equation and subtract the resulting terms afterwards or write the equation of the form $p(x) = \text{TRUE}/\text{FALSE}$ and convert the Boolean polynomial $p(x)$ as a whole using the Standard Conversion Method into a polynomial over the reals. The advantage of splitting the equation into two parts is that the degree of the resulting polynomial is at most two. This yields equations over the reals of the following form:

$$\begin{aligned}x_{162} + x_{177} - 2x_{162}x_{177} &= z \\ \tilde{r}_1 - x_{66} - x_{93} + 2x_{66}x_{93} &= 0 \\ \tilde{r}_2 - x_{91}x_{92} &= 0 \\ \tilde{r}_1 + \tilde{r}_2 - 2\tilde{r}_1\tilde{r}_2 - x_{171} - x_{178} + 2x_{171}x_{178} &= 0 \\ \tilde{r}_3 - x_{175}x_{176} &= 0 \\ (1 - 2z)\tilde{r}_3 - x_{69} - x_{179} + 2x_{69}x_{179} &= -z\end{aligned}$$

where $x_i = t(s_i)$ and $\tilde{r}_i = t(r_i)$.

We still have equations of degree 2. All these constraints are equality constraints. The last step is to replace the quadratic terms by new variables which will be forced to take the correct values by additional constraints.

Linearisation. In order to linearise the equations we introduce a new binary variable y for each quadratic term $x_i x_j$. We want this new variable to be zero if x_i or x_j is zero and to be one if and only if x_i and x_j are both one. We can achieve this by adding the following inequalities to the system of constraints:

$$y \leq x_i \tag{7}$$

$$y \leq x_j \tag{8}$$

$$x_i + x_j - 1 \leq y \tag{9}$$

The inequality constraints (7) and (8) make sure that $y = 0$ if $x_i = 0$ or $x_j = 0$ and (9) ensures that $y = 1$ if $x_i = 1$ and $x_j = 1$. We introduce one new variable and three inequality constraints for each quadratic term. The equality constraints are of the form

$$\begin{aligned}x_{162} + x_{177} - 2y_1 &= z \\ \tilde{r}_1 - x_{66} - x_{93} + 2y_2 &= 0 \\ \tilde{r}_2 - y_3 &= 0 \\ \tilde{r}_1 + \tilde{r}_2 - 2y_4 - x_{171} - x_{178} + 2y_5 &= 0 \\ \tilde{r}_3 - y_6 &= 0 \\ (1 - 2z)\tilde{r}_3 - x_{69} - x_{179} + 2y_7 &= -z\end{aligned}$$

This linearisation yields a system of 732 equality and 2529 inequality constraints in 1575 variables (again starting from a fully determined system with 399 variables in 399 unknowns).

5.2 Linearisation Using the Basic Idea of the ASC

In this linearisation method we use the fact that we want to use the resulting system of equalities and inequalities in a mixed-integer programming problem. This means that we can restrict some (or all) variables to be integers. Consequently we do not have to ensure that the equation holds over the reals but over the integers if we convert a Boolean equation.

The main idea how to convert a Boolean equation into an equation over the integers is taken from the Adapted Standard Conversion. We interpret the Boolean equation as an equation over the integers by replacing XOR by addition and AND by multiplication. Then we evaluate the integer equation for all solutions of the Boolean equation as we did in the ASC method (Here we map TRUE to 1 and FALSE to 0). Afterwards we subtract these results from the left-hand side of the equation. We observe that all results are multiples of 2. That means instead of generating several equations and introducing exclusion variables we just introduce one new integer-valued variable in a linear term and get one equation.

Example 2. Consider the Boolean equation

$$s_1 \oplus s_2 \oplus (s_3 \wedge s_4) \oplus s_5 \oplus s_6 = 0 \quad (10)$$

If we evaluate the corresponding real polynomial $x_1 + x_2 + x_3x_4 + x_5 + x_6$ for all solutions of (10) we get 0, 2, 4 as results. That means that a solution of (10) is a solution to the following equation over the integers

$$x_1 + x_2 + x_3x_4 + x_5 + x_6 - 2y = 0 \text{ where } y \in \{0, 1, 2\}.$$

The degree is the same and the number of monomials per equation is increased only by one. We call this conversion method *Integer Adapted Standard Conversion* (IASC).

It is assumed that the smaller the number of monomial per equation is, the easier it is to solve the resulting mixed-integer problem. For this reason we start by splitting the equations by introducing auxiliary variables. We introduce $t_1 = s_{66} \oplus s_{93}$ and also replace $s_{162} \oplus s_{177}$ by the key stream bit z in (6). We apply the Standard Conversion Method to equations with no more than three monomials. In this case the number of monomials per equation and the number of new variables is the same as when using the IASC. But by replacing a quadratic term by a new variable we get three constraints instead of only the restriction that the variable is binary. That means we get stronger constraints by using the Standard Conversion in these cases. For equations with more than three monomials we use the IASC. This yields equations of the form:

$$\begin{aligned}x_{162} + x_{177} - 2x_{162}x_{177} &= z \\t_1 - x_{66} - x_{93} + 2x_{66}x_{93} &= 0 \\t_1 + x_{91}x_{92} + x_{171} + x_{178} - 2y_1 &= 0 \\x_{175}x_{176} + x_{69} + x_{179} - 2y_2 &= z\end{aligned}$$

where $y_1 \in \{0, 1, 2\}$ and $y_2 \in \{0, 1\}$. Finally we replace the quadratic terms by new variables and add the corresponding inequality constraints (7)-(9) (as in Section 5.1.) Starting from a fully determined Boolean system this yields a system of 2040 equalities and inequalities in 1020 variables.

5.3 Bivium A as a Feasibility Problem

From the linearisation we obtain a system of linear equalities and inequalities over the reals respectively over the integers which describes the internal state of Bivium A. Since the number of variables exceeds the number of equality constraints we cannot use Gauss elimination to solve the system. Therefore we use these constraints to describe the feasible set of a linear mixed-integer programming problem. Let

$$\min_x \{c^T x : Ax \leq b, x \in \{0, 1\}^{k_1} \times \mathbb{Z}^{k_2} \times \mathbb{R}^l\} \quad (11)$$

be the mixed-integer programming problem describing Bivium A where A and b are given by the constraints obtained from converting and linearising the Boolean equations of Bivium A and by the upper and lower bounds on the variables. If we can find a feasible binary/integer-valued solution for this MIP for an arbitrary objective function, this solution can be converted into a solution for the Boolean system. Hence it is not important to find a minimal solution but a feasible point. It is possible that more than one feasible point exists, it is also possible that the Boolean system does not have a unique solution. But it is likely that an overdetermined Boolean system has a unique solution and the corresponding feasible region contains only one element.

Furthermore we can observe that most of the variables are dependent on the initial state variables (all except the variables introduced by IASC in Section 5.2). This means that we do not have to restrict all variables to be

integers nor binary. It is sufficient to force the initial state variable to be binary (and the IASC-variables to be integers).

In the next section we will discuss which function might be a good objective function, how many additional key stream equations we should generate to obtain an overdetermined system and which variables should be restricted to be binary or integers. Furthermore we will compare the two different linearisation methods.

6 Results

In this section we present our observations and results from experiments with various variants of Bivium A and Bivium B as a mixed-integer programming problems. We mainly focus on MIPs which use the constraints obtained using the Standard Conversion Method and linearisation. Here we consider some variants of Bivium A with a smaller state size, see Table 1, to compare the increase of the solution time to the number of variables. We also tried to find the optimal number of additional key stream equations, a good objective function and the variables which should be restricted to be binary. Later we will compare this approach to MIPs using constraints obtained by using the IASC.

For all experiments we use the commercial linear optimization tool CPLEX by ILOG [1]. CPLEX has a user's choice for emphasis on feasibility or optimality. We choose emphasis on feasibility because we are not interested in optimality and stop after we found the first solution because we assume that there is only one solution. We use CPLEX version 9.130 on a Sun Fire E25K SF 12K with shared processors. On this machine 2^{24} Bivium simulations over 5×177 steps take $2^{14.5}$ seconds. This means we can approximately search through $2^{9.5}$ keys per second.

Table 1. Variants of Bivium A with smaller state size

Name	Step 1	Step 2	Step 3	Bivium A
state size	118	133	147	177
key stream bits required	158	177	196	236
variables	1530	1718	1894	2283
equalities	728	817	901	1086
constraints	3254	3652	4027	4854

6.1 Using Standard Conversion

For finding the right parameter (objective function, binary variables, additional equations) we ran most of the tests for the smallest variant Bivium A Step 1. This variant has internal state size 118 and keeps the structure of Bivium A as much as possible. We confirmed these parameters for the variants with larger state size by running some spot tests.

$$\begin{aligned}
t_1 &\leftarrow s_{44} + s_{62} \\
t_2 &\leftarrow s_{108} + s_{118} \\
z &\leftarrow t_1 + t_2 \\
t_1 &\leftarrow t_1 + s_{60}s_{61} + s_{114} \\
t_2 &\leftarrow t_2 + s_{116}s_{117} + s_{46} \\
(s_1, s_2, \dots, s_{62}) &\leftarrow (t_2, s_1, \dots, s_{61}) \\
(s_{63}, s_{64}, \dots, s_{118}) &\leftarrow (t_1, s_{63}, \dots, s_{117})
\end{aligned}$$
Fig. 1. Pseudo code Bivium A Step 1

Binary vs. Continuous. As mentioned before using the Standard Conversion Method to convert the Boolean equations into real equations, all new introduced variables (the new introduced state variables, as well as the auxiliary variables and the linearisation variables) are dependent on the initial state variables. This means that forcing the initial variables to take binary values will also force all other variables to take binary values. This raises the question, whether it is an advantage to have binary restrictions only for the initial state variables instead of for all? A branch-and-bound algorithm has to determine a variable for branching in each step. In the binary case this variable will be assigned to 0 and 1 to grow the search tree further in one node. Intuition tells us that it is better to choose variables for branching which automatically force many other

Table 2. Comparison of running time in seconds for Bivium A Step 1 between the MIP with binary restriction on the initial state and on all variables. The two last rows of the upper table contain the average running time in seconds and the standard deviation for a sample using a fully determined system. The lower table contains the average running time and the standard deviation for a sample using an overdetermined system.

No.	mixed	binary
1	385	1583
2	550	890
3	170	478
4	157	277
5	629	1297
6	42	27
7	1209	620
8	213	1011
9	256	286
10	484	1979
average	548	783
std	393	563
	mixed	binary
average	493	930
std	272	289

variables to take binary values. This can be achieved by only restricting these variables to be binary or by giving the algorithm an order in which it should consider the variables. We can confirm our intuition by experiments, see Table 2. Table 2 shows the running time for 10 typical instances of the problem. An 80 bit key is chosen at random to generate a fully determined Boolean equation system. This system is used to formulate a linear mixed-integer programming problem as explained before. As an objective function the sum over all variables is used. The both last rows of the upper table in Table 2 give the average and the standard deviation taken over a larger sample. The lower table contains the average and standard deviation over a sample where the Boolean system is overdetermined. We can see that the average running time is cut by 30% to almost 50% when we compare the case where all variables are restricted to be binary to the case where only the initial state variables have binary restrictions. Even though the running times vary a lot for both cases the standard deviation is smaller for the mixed problems. Furthermore we can note a decrease of the standard deviation when we consider overdetermined problems. Considering these improvements it seems to be a good strategy to restrict only the initial state variables to be binary. All other variables are continuous variables.

Objective Function. As mentioned before, Bivium A is formulated as a feasibility problem and the objective function does not influence the solution in a way that matters for us. That means we can choose an arbitrary objective function. Even if the objective function is not important to get the correct solution of the problem it is important for the performance of many mixed-integer algorithms. In branch-and-bound algorithms the bounding function estimates the best value of the objective function obtainable by growing the tree one node further. This value is an important factor in the process of choosing the next node in the search tree. The closer the value of the bounding function to the objective function the better. Mixed-integer programming algorithms often use the corresponding linear programming problem (LP) as bounding function but we do not go into detail here. In the corresponding LP all variables are continuous i.e., we have no integrality or binary restrictions.

The only restriction for the objective function we have is that it must be linear. Natural choices are

1. the zero-function
2. the sum over the initial state variables
3. the sum over all variables which are introduced by the original Boolean system i.e. the sum over all state variables
4. the sum over all variables
5. the sum over all but the initial state variables¹

Already a few tests showed that the zero-function is not a good objective function in practice. This is not surprising because the constant zero-function gives

¹ Suggested by one of the reviewers.

Table 3. MIP Bivium A Step 1 with different objective function (fully determined system). The rows labeled with numbers 1 to 9 contain the run time in seconds corresponding to the chosen objective function. The row labeled with average show the average run time in seconds and the last row gives us the standard deviation.

No.	$\sum_{i=1}^{118} x_i$	$\sum_{i=1}^{270} x_i$	$\sum_{i=1}^{\#variables} x_i$	$\sum_{i=118}^{\#variables} x_i$
1	510	365	262	202
2	477	635	198	270
3	180	1325	791	723
4	495	931	621	512
5	627	1151	411	168
6	145	506	244	180
7	544	342	939	394
8	387	1057	416	542
9	613	527	641	493
10	972	779	136	382
average	573	678	449	463
std	290	320	232	286

the algorithm no information and does not show any improvement. We focused on the four remaining candidates for the objective function. Our experiments show, see Table 3, that we achieve the best running times if we use the sum over all variables as the objective function. Here again the first 10 rows contain the running times of 10 typical instances of the mixed integer programming problem we get from a randomly chosen keys and a corresponding overdetermined Boolean system. In our formulation of the MIP only the initial state variables are restricted to be binary. The last two rows contain the average running time and standard deviation taken over a larger sample. As mentioned before the sum over all variables turns out to be the best objective function in this sample and is used in all further experiments. But the 5th candidate, the sum over all non-state variables, is almost as good and we expect that these two functions will yield similar results because the functions have similar properties.

For a key chosen at random approximately one third of the variables take the value one, while if we only look at the initial state variables or the variables introduce by the original Boolean system approximately half of them take the value one. Using the sum of certain variables as objective function means looking for a solution where the Hamming weight of these variables is minimal. For the second and third candidates the expected value of the objective function for the correct point is the same as for a random point. But for the fourth candidate, the sum over all variables, the expected value for the correct solution is only one third of the number of variables which is significantly less than half of the number of variables which is the expected value for a random point. Here minimizing the objective function leads us in the right direction. Also the last candidate has the property that minimizing it leads to the correct solution.

The observation that the sum over all variables is the best objective function raises the question if the MIP is solved faster if the Hamming weight of the

solution is low. We tested this on an overdetermined system of Bivium A (59 additional key stream equations and the corresponding quadratic equations) and our tests show that problems with a low weight solution are in average solved much faster, see Table 4. The hypothesis of our experiments is:

Table 4. Running time Bivium A for low weight solution (All values are averages over a small samples.)

HW initial state	HW solution	time in sec
10	209,4	0,2
20	346,8	17,2
40	496,8	780,3
60	617,0	9113,6
80	657,4	9342,0
100	770,9	29135,7
120	868,8	27777,7

The sum over all variables $\sum_{i=1}^m x_i$ where m is the number of variables, is a good choice for the objective function.

We use this objective function in the remaining experiments.

Overdetermined System. The fully determined or quadratic Boolean system (means n equations in n unknowns where $n = 399$ in the case of Bivium A) has possibly more than one solution. For an overdetermined system it is likely that the solution is unique. Fortunately it is very easy to generate an overdetermined Boolean system. After we have generated a fully determined system each additional key stream bit gives us three equations and two new variables. Then 5 to 10 additional key stream bits are already sufficient to get a unique solution in most cases. The advantage of adding even more equations is that the feasible set will get smaller and hence the algorithm will become faster. On the other hand more constraints come into play which will slow down the algorithm. The question is how many additional key stream equations and corresponding quadratic

Table 5. Average running times for overdetermined systems

Bivium A Step 1

add. key stream bits	+5	+10	+20	+40	+59
average time in sec	743	548	719	438	2063

Bivium A Step 2

add. key stream bits	+10	+40	+44	+67
average time in sec	3493	2446	2209	3767

Bivium A

add. key stream bits	+10	+59	+89
average time in sec	25759	15309	21950

Boolean equations we should add. Here the hypothesis of our experiments is, see Table 5:

Generate one third more key stream equations and the corresponding quadratic equations to define the system.

Results on Bivium A Using Standard Conversion. We ran tests for all variants of Bivium A where the key was chosen at random. We used the following settings which are good according to our experiments before:

- objective function: the sum over all variables;
- generate an overdetermined system by generating one third additional key stream equations;
- restrict the initial state variables to be binary, all other variables are continuous.

We summarize the results in Table 6.

Table 6. Overview over Bivium A with different state size

Name	Step 1	Step 2	Step 3	Bivium A
state size	118	133	147	177
key stream bits require	158	177	196	236
variables	1530	1718	1894	2283
equalities	728	817	901	1086
constraints	3254	3652	4027	4854
time in sec.	555	2110	2535	15267

We are able to break Bivium A in less than 4.5 hours on average. This shows us that our approach is faster than Raddum's [8] (about a day) but slower than using MiniSAT [3] (reported to be 21 sec).

Results on Bivium B Using the Standard Conversion

In the same manner as we for Bivium A we can convert Bivium B into a mixed-integer programming problem. We use the same parameters as we did for Bivium A, i.e. the objective function is the sum over all variables, we generate an overdetermined system by adding 59 additional key stream and corresponding state update equations and we restrict only the first 177 variables to be binary. This yields an MIP in 2821 variables and 5865 constraints, 1388 of these are equality constraints. We start by considering variants of Bivium B with smaller state size (59 and 118 bits). Furthermore we reduce the complexity of the problem by guessing bits. When we run all experiments in parallel the expected running time of the algorithm is the time it needs to find the solution for the correct guess times 2^n where n is the number of guessed bits. Inspired by

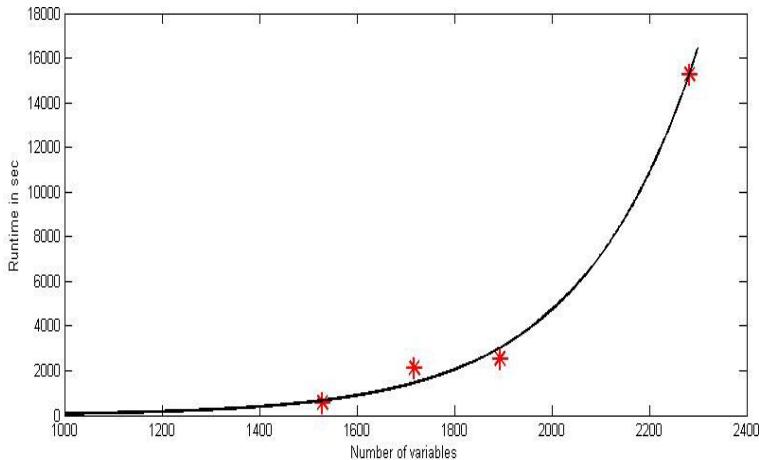


Fig. 2. Running time for Bivium A with different state sizes (exponential compensating curve)

[9] we tried two different guessing strategies. We guessed bits at the beginning or at the end of the initial state. It turned out that guessing the last bit is a better strategy. For the variant of Bivium B with internal state size 118 the average time to find the solution after we guessed the first 30 bits correct is 31635 sec while the average time for guessing the last 30 bits correct is just 6334 sec. Here we have an improvement by a factor of five. For that reason we will guess the last bits of the initial state in all further experiments. Here we can also see that for a variant of Bivium B with state size 118 we have to guess 30 variables. This means the problem has a high complexity. One way to simplify the problem is to reduce the size of the feasible region by adding additional constraints. We tried two different kinds of additional constraints

- The AND-gate constraints

The quadratic Boolean update functions contain a quadratic term (an AND-gate) which we replace by a new variable later. These AND-gates contain successive variables. This means if one AND-gate or one of the variables which replace the AND-gate is zero than also the following or the prior AND-gate/variable has to be zero. This leads to the following deterministic constraint:

$$r_1 + r_3 - r_2 \leq 1$$

where $r_1 = x_1x_2$, $r_2 = x_2x_3$ and $r_3 = x_3x_4$ are variables which replace AND-gates.

- Probabilistic constraints

The initial state of Bivium B is produced by the key setup. Thus we can assume that it behaves like a random bit stream. This means that 10 consecutive bits contain at least one "1" and one "0" with high probability.

Therefore we can add the probabilistic constraints

$$\sum_{k=i}^{i+9} x_i \leq 9$$

and

$$-\sum_{k=i}^{i+9} x_i \leq -1.$$

If these constraints are not true we won't find a integer feasible solution.

Table 7 lists the running times for 10 instances of the problem as well as the average running time which is taken over a larger sample. As we can see here the additional constraints do not improve the running time, actually they impair it. One reason could be that we increase the number of constraints too much, maybe adding just a few new constraints would improve the running time (no tests on that so far). Furthermore we can see that the variance for the results on Bivium B with 50 bits guessed is quite high. So far there is no explanation for the big differences in the running time but a general observation is that the more variables we have the higher is the standard deviation.

We can determine the initial state for Bivium B in $2^{63.7}$ seconds (assuming that we run the tests for all guesses in parallel). Our simulations showed that we can search through 2^{24} keys in $2^{14.5}$ seconds. That means that the complexity of our approach corresponds to searching through $2^{73.2}$ keys. This is not a very impressive result compared to the result using Raddum splitting algorithm [8] which takes 2^{56} seconds, this corresponds to $2^{69.3}$ keys or the even better result

Table 7. Timing results on Bivium B in seconds (*including cost for bit guessing). The numbers in the first row indicate how many bits are preassigned, 'Prob' means we added the Probabilistic constraints, 'AND-Gate' means we added the AND-Gate constraints, 'All' means we added the probabilistic and the AND-Gate constraints. The rows labeled with 1 to 10 contain the run time in seconds excluding the cost for bit guessing.

	50 given	55 given	55+All	55 + prob	55+ And-gate
1	1073	104	378	295	382
2	1325	550	670	272	591
3	213	27	777	1422	1711
4	97592	3818	230	1150	742
5	79935	213	4579	999	2194
6	31311	1308	2333	1834	1667
7	7642	452	336	1280	2623
8	486	949	339	890	566
9	745	364	1303	1060	1734
10	8434	434	802	445	851
average *	$2^{63.7}$	$2^{65.2}$	$2^{65.2}$	$2^{65.06}$	$2^{65.5}$

using MiniSAT [3] which takes $2^{42.7}$ seconds to attack Bivium or correspondingly a search through 2^{56} keys.

6.2 Using IASC

We describe the feasible set of the MIP by the constraints we get from using the conversion and linearisation described in Section 5.2. As objective function we use the sum over all but the variables introduced by IASC, i.e. we sum over all binary variables. We restrict the initial state variables to be binary and the variables introduced by IASC to their integer values. From Table 8 we can see that the running time for this approach is worse than for the first one. This means that even if the approach using IASC has less variables and constraints than the approach using Standard Conversion the running time is worse. One reason for that might be that parameters which yield good results in our first approach are not good for the second one. Another more likely reason is that we have more variables with restrictions in the approach using IASC, nearly three times as many as using the Standard Conversion approach. This means that the algorithm has to consider more variables when it chooses the next branching variable. Some of these variables are not binary but integer-valued. That means here are even more possibilities to fulfill this equation and therefore the constraint is weaker. However, this approach might still be interesting if we consider Bivium B. Here the lower complexity of the resulting constraints could

Table 8. Running time in seconds for some test on Bivium A using IASC

no.	+10 key stream bits	+59 key stream bits
1	243241	50395
2	211305	27853
3	6572	8912
4	35296	12545
5	9966	6760
6	29650	39950
7	52230	10596
8	183083	1050
9	130254	111693
av	100180	27652

Table 9. Parameter of the two approaches for Bivium A

Method	IASC	SC
state size	177	177
variables	1773	2283
equalities	746	1086
constraints	2984	4854
variables with restrictions	517	177

be an advantage even if we get more variables with restrictions. There are no tests on Bivium B so far.

7 Conclusion

We showed two ways of transforming Bivium into a mixed-integer linear programming problem. One way uses the Standard Conversion Method to convert Boolean equations into equations over \mathbb{R} , the other way uses the Integer Adapted Standard Conversion to convert the Boolean equations into equations over \mathbb{Z} . These two methods are also applicable to any other Boolean equation system. The best results for Bivium A are achieved by using the Standard Conversion method with an estimated time complexity of $2^{13.9}$ seconds or approximately 4.5 hours. Using the Integer Adapted Standard Conversion we get an attack with running time of $2^{14.76}$ seconds or approximately 8 hours. Solving the MIP which corresponds to Bivium B converted using the Standard Conversion has an average time complexity of $2^{64.5}$ seconds which corresponds to a search through $2^{73.2}$ keys. Comparing the results on variants of Bivium B with a smaller state size to Bivium A shows that not only the increased number of variables in Bivium B but also the more complex structure of the equations are responsible for the increase of the running time. Moving from Bivium B to Trivium would double the number of variables in the corresponding MIP. Also, the key stream equation of Trivium involves 6 variables and is hence more complicated than the key stream equation of Bivium B. Therefore the mixed-integer programming problem as presented in this article can not in its current form be considered as a threat for Trivium.

Acknowledgments. We thank the anonymous reviewers for their helpful comments.

References

1. ILOG CPLEX, <http://www.ilog.com/products/cplex/>
2. Beigel, R.: The polynomial method in circuit complexity. In: Structure in Complexity Theory Conference, pp. 82–95 (1993)
3. McDonald, C., Charnes, C.: An algebraic analysis of Trivium ciphers based on the boolean satisfiability problem. Cryptology ePrint Archive, Report 2007/129 (2007), <http://eprint.iacr.org/2007/129.pdf>
4. De Cannière, C., Preneel, B.: Trivium - a stream cipher construction inspired by block cipher design principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006)
5. De Cannière, C., Preneel, B.: Trivium specifications. ECRYPT eSTREAM Project (2006), http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf
6. Lamberger, M., Nad, T., Rijmen, V.: Numerical solvers in cryptanalysis (extended abstract). In: Second Workshop on Mathematical Cryptology (2008)

7. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization. Prentice-Hall, Inc., Englewood Cliffs (1982)
8. Raddum, H.: Cryptanalytic results on Trivium. eSTREAM report 2006/039 (2006), <http://www.ecrypt.eu.org/stream/triviump3.html>
9. Eibach, T., Pilz, E., Völkel, G.: Attacking Bivium using SAT solvers. In: Kleine Büning, H., Zhao, X. (eds.) SAT 2008. LNCS, vol. 4996, pp. 63–76. Springer, Heidelberg (2008)
10. Wolsey, L.A., Nemhauser, G.L.: Integer and Combinatorial Optimization. Wiley Interscience, Hoboken (1999)

Security of Cyclic Double Block Length Hash Functions

Ewan Fleischmann, Michael Gorski, and Stefan Lucks

Bauhaus-University Weimar, Germany

{ewan.fleischmann,michael.gorski,stefan.lucks}@uni-weimar.de

Abstract. We provide a proof of security for a huge class of double block length hash function that we will call CYCLIC-DM. Using this result, we are able to give a collision resistance bound for ABREAST-DM, one of the oldest and most well-known constructions for turning a block cipher with n -bit block length and $2n$ -bit key length into a $2n$ -bit cryptographic hash function. In particular, we show that when ABREAST-DM is instantiated using a block cipher with 128-bit block length and 256-bit key length, any adversary that asks less than $2^{124.42}$ queries cannot find a collision with success probability greater than $1/2$. Surprisingly, this about 15 years old construction is one of the few constructions that have the desirable feature of a near-optimal collision resistance guarantee.

We are also able to derive several DBL constructions that lead to compression functions offering an even higher security guarantee and more efficiency than ABREAST-DM (*e.g.* share a common key). Furthermore we give a practical DBL construction that has the highest security guarantee of all DBL compression functions currently known in literature. We also provide a (relatively weak) analysis of preimage resistance for CYCLIC-DM.

Keywords: cryptographic hash function, block cipher based, proof of security, double-block length, ideal cipher model, Cyclic-DM, Abreast-DM.

1 Introduction

A cryptographic hash function is a function which maps an input of arbitrary length to an output of fixed length. It should satisfy at least collision-, preimage- and second-preimage resistance and is one of the most important primitives in cryptography [25].

Block Cipher-Based Hash Functions. Since their initial design by Rivest, MD4-family hash functions (*e.g.* MD4, MD5, RIPEMD, SHA-1, SHA2 [3,28,29,31,32]) have dominated cryptographic practice. But in recent years, a sequence of attacks on these type of functions [8,12,40,41] has led to a generalized sense of concern about the MD4-approach. The most natural place to look for an alternative is in block cipher-based constructions, which in fact predate the MD4-approach [24].

Another reason for the resurgence of interest in block cipher-based hash functions is due to the rise of size restricted devices such as RFID tags or smart cards: A hardware designer has to implement only a block cipher in order to obtain an encryption function as well as a hash function. But since the output length of most practical encryption functions is far too short for a collision resistant hash function, *e.g.* 128-bit for AES, one is mainly interested in sound design principles for *double block length* (DBL) hash functions [2]. A DBL hash-function uses a block cipher with n -bit output as the building block by which it maps possibly long strings to $2n$ -bit ones.

Our Contribution. Four, somewhat 'classical' DBL hash functions are known: MDC-2, MDC-4, ABREAST-DM and TANDEM-DM [4,5,23]. Security bounds for MDC-2 and TANDEM-DM had been shown [38,10] at EUROCRYPT'07 and FSE'09. In [18], Lee and Kwon independently provided a proof of security of ABREAST-DM and discussed some generalizations (without stating proofs). Since our results are tighter (both preimage and collision resistance) and far more general, we will give them here in full detail. We also provide a rigorous formal analysis of our security bounds. Using our generalization, CYCLIC-DM, we are able to derive a DBL compression function that offers the highest security guarantee of all compression functions currently known in literature.

Outline. The paper is organized as follows: Section 2 includes formal notations and definitions as well as a review of related work. In Section 3, we discuss ABREAST-DM as a special case of CYCLIC-DM: we show that any adversary asking less than $2^{124.42}$ oracle queries has negligible advantage in finding a collision for the ABREAST-DM compression function. The proof for this special case is given in the full version of the paper [11]. Section 4 now defines the compression function CYCLIC-DM and gives security bounds in terms of collision resistance and preimage resistance. Section 5 discusses how we can use the results from the previous section to derive new DBL compression functions that have the highest security guarantee of all currently known DBL compression functions. In Section 6 we discuss our results and conclude.

2 Preliminaries

2.1 Iterated DBL Hash Function Based on Block Ciphers

Ideal Cipher Model. A block cipher is a keyed family of permutations consisting of two paired algorithms $E : \Omega \times \mathcal{K} \rightarrow \Omega$ and $E^{-1} : \Omega \times \mathcal{K} \rightarrow \Omega$ where Ω is the set of plaintexts/ciphertexts, and \mathcal{K} the set of keys. If $\Omega = \{0, 1\}^n$ and $\mathcal{K} = \{0, 1\}^k$, we will call it an (n, k) -block cipher. Let $\text{BC}(\Omega, \mathcal{K})$ be the set of all such block ciphers. Now, for any one fixed key $K \in \mathcal{K}$, decryption $E_K^{-1} = E^{-1}(\cdot, K)$ is the inverse function of encryption $E_K = E(\cdot, K)$, so that $E_K^{-1}(E_K(X)) = X$ holds for any input $X \in \Omega$.

Most of the attacks on hash functions based on block ciphers do not utilize the internal structure of the block ciphers. The security of such hash functions

is usually analyzed in the *ideal cipher model* [2,9,20]. In the ideal cipher model the underlying primitive, the block cipher E , is modeled as a family of random permutations $\{E_K\}$ whereas the random permutations are chosen independently for each key K , *i.e.* formally E is selected randomly from $\text{BC}(\mathcal{X}, \mathcal{K})$.

DBL Compression Functions. Iterated DBL hash functions with two block cipher calls in their compression function are discussed in this article. A hash function $H : \{0,1\}^* \rightarrow \mathcal{X}^2$ can be built by iterating a compression function $F : \Omega^2 \times \{0,1\}^b \rightarrow \Omega^2$ as follows: Split the padded message M into b -bit blocks M_1, \dots, M_l , fix (G_0, H_0) , apply $(G_i, H_i) = F(G_{i-1}, H_{i-1}, M_i)$ for $i = 1, \dots, l$ and finally set $H(M) := (G_l, H_l)$. Let the compression function F be such that

$$(G_i, H_i) = F(G_{i-1}, H_{i-1}, M_i),$$

where $G_{i-1}, H_{i-1}, G_i, H_i \in \Omega$ and $M_i \in \{0,1\}^b$. We assume that the compression function F consists of F_T , the top row, and F_B , the bottom row. Each of the component functions F_B and F_T performs exactly *one* call to the block cipher and can be defined as follows:

$$\begin{aligned} G_i &= F_T(G_{i-1}, H_{i-1}, M_i) = E(X_T, K_T) \oplus Z_T, \\ H_i &= F_B(G_{i-1}, H_{i-1}, M_i) = E(X_B, K_B) \oplus Z_B, \end{aligned}$$

where X_T, K_T, Z_T and X_B, K_B, Z_B are uniquely determined by G_{i-1}, H_{i-1}, M_i . We define the rate r of a block cipher based compression/hash function F by

$$r = \frac{|M_i|}{(\text{number of block cipher calls in } F) \times n}.$$

The key scheduler rate r_{key} is defined as

$$r_{key} = \frac{1}{\text{number of key scheduler operations per compression function}}.$$

It follows that $r_{key} = 1$ if $K_T = K_B$ and $r_{key} = 1/2$ otherwise. They both are a measure of efficiency for such block cipher based constructions. Note that there is currently a discussion in literature on how to measure this efficiency 'correctly'.

2.2 Defining Security – Collision Resistance of a Compression Function

Insecurity is quantified by the success probability of an optimal resource-bounded adversary. The resource is the number of queries to the ideal cipher oracles E or E^{-1} . For a set S , let $z \xleftarrow{R} S$ represent random sampling from S under the uniform distribution. For a probabilistic algorithm \mathcal{M} , let $z \xleftarrow{R} \mathcal{M}$ mean that z is an output of \mathcal{M} and its distribution is based on the random choices of \mathcal{M} .

An adversary is a computationally unbounded but always-halting collision-finding algorithm \mathcal{A} with access to an oracle $E \in \text{BC}(\mathcal{X}, \mathcal{K})$. We can assume

(by standard arguments) that \mathcal{A} is deterministic. The adversary may make a *forward* query $(X, K, ?)_{fwd}$ to discover the corresponding value $Y = E_K(X)$, or the adversary may make a *backward* query $(?, K, Y)_{bwd}$, so as to learn the corresponding value $X = E_K^{-1}(Y)$ for which $E_K(X) = Y$. Either way the result of the query is stored in a triple (X_i, K_i, Y_i) and the *query history*, denoted \mathcal{Q} , is the tuple (Q_1, \dots, Q_q) where $Q_i = (X_i, K_i, Y_i)$ is the result of the i -th query made by the adversary and where q is the total number of queries made by the adversary. Without loss of generality, it is assumed that \mathcal{A} asks at most only once on a triplet of a key K_i , a plaintext X_i and a ciphertext Y_i obtained by a query and the corresponding reply.

The goal of the adversary is to output two different triplets (G, H, M) and (G', H', M') such that $F(G, H, M) = F(G', H', M')$. Since E is assumed to be an ideal cipher, we impose the reasonable condition that the adversary must have made all queries necessary to compute $F(G, H, M)$ and $F(G', H', M')$. We will in fact dispense the adversary from having to output these two triplets, and simply determine whether the adversary has been successful or not by examining its query history \mathcal{Q} . Formally, we say that $\text{COLL}(\mathcal{Q})$ holds if there is such a collision and \mathcal{Q} contains all the queries necessary to compute it.

Definition 1. (*Collision resistance of a compression function*) Let F be a blockcipher based compression function, $F : \Omega^2 \times \{0, 1\}^b \rightarrow \Omega^2$. Fix an adversary \mathcal{A} . Then the advantage of \mathcal{A} in finding collisions in F is the real number

$$\mathbf{Adv}_F^{\text{COLL}}(\mathcal{A}) = \Pr[E \xleftarrow{R} \text{BC}(\mathcal{X}, \mathcal{K}); ((G, H, M), (G', H', M')) \xleftarrow{R} \mathcal{A}^{E, E^{-1}} : ((G, H, M) \neq (G', H', M')) \wedge F(G, H, M) = F(G', H', M')].$$

For $q \geq 1$ we write

$$\mathbf{Adv}_F^{\text{COLL}}(q) = \max_{\mathcal{A}} \{\mathbf{Adv}_F^{\text{COLL}}(\mathcal{A})\}$$

where the maximum is taken over all adversaries that ask at most q oracle queries (i.e. E and E^{-1} queries).

2.3 Related Work

Schemes with non-optimal or unknown collision resistance. Preneel *et al.* [30] discussed the security of SBL hash functions against several generic attacks. They concluded that 12 out of 64 hash functions are secure against the attacks. However, formal proofs were first given by Black *et al.* [2] about 10 years later. Their most important result is that 20 hash functions – including the 12 mentioned above – are optimally collision resistant. Knudsen *et al.* [21] discussed the insecurity of DBL hash functions with rate 1 composed of (n, n) -block ciphers. Hohl *et al.* [16] analyzed the security of DBL compression functions with rate 1 and 1/2. Satoh *et al.* [36] and Hattoris *et al.* [13] discussed DBL hash functions with rate 1 composed of $(n, 2n)$ -block ciphers. MDC-2 and MDC-4

[17,1,5] are (n, n) -block cipher based DBL hash functions with rates $1/2$ and $1/4$, respectively. Steinberger [38] proved that for MDC-2 instantiated with, *e.g.*, AES-128 no adversary asking less than $2^{74.9}$ can usually find a collision. Nandi *et al.* [27] proposed a construction with rate $2/3$ but it is not optimally collision resistant. In [22], Knudsen and Muller presented some attacks against it. At EUROCRYPT'08 and CRYPTO'08, Steinberger [34,35] proved some security bounds for fixed-key (n, n) -block cipher based hash functions, *i.e.* permutation based hash functions, that all have small rates and low security guarantees. None of these schemes/techniques mentioned so far are known to have birthday-type collision resistance.

Schemes with Birthday-Type Collision Resistance. Merkle [26] presented three DBL hash functions composed of DES with rates of at most 0.276. They are optimally collision resistant in the ideal cipher model. Hirose [14] presented a class of DBL hash functions with rate $1/2$ which are composed of two different and independent $(n, 2n)$ -block ciphers that have birthday-type collision resistance. At FSE'06, Hirose [15] presented a rate $1/2$ and $(n, 2n)$ -block cipher based DBL hash function that has birthday-type collision resistance. He essentially stated that for his compression function, no adversary can find a collision with probability greater than $1/2$ if no more than $2^{124.55}$ queries are asked (see [10, App. B] for details on this). At FSE'09, Fleischmann et. al. [10] showed that for TANDEM-DM, no adversary asking less than $2^{120.4}$ queries can find a collision with probability greater than $1/2$. In [18], Lee and Kwon independently provided a security bound for ABREAST-DM (numerically slightly weaker compared to our result in this article, $2^{124.42}$). They also stated a generalized Theorem (without proof) and gave a compression function with a security guarantee of up to 2^{125} queries which does not have the desirable feature of a common key for both encryption functions. Our results will be formally tighter in all cases compared to [18].

3 Security of ABREAST-DM

3.1 Compression Function

The ABREAST-DM hash function was proposed at EUROCRYPT '92 by Xuejia Lai and James L. Massey [23]. It incorporates two Davies-Meyer (DM) single block length compression functions [25] which are used side-by-side. The compression function is illustrated in Figure 1 and is formally given in Definition 2.

Definition 2. Let $F^{\text{ADM}} : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ be a compression function such that $(G_i, H_i) = F^{\text{ADM}}(G_{i-1}, H_{i-1}, M_i)$ where $G_i, H_i, M_i, G_{i-1}, H_{i-1} \in \{0, 1\}^n$. F^{ADM} consists of a $(n, 2n)$ -block cipher E as follows:

$$\begin{aligned} G_i &= G_{i-1} \oplus E_{H_{i-1}|M_i}(G_{i-1}) \\ H_i &= H_{i-1} \oplus E_{M_i|G_{i-1}}(\overline{H}_{i-1}), \end{aligned}$$

where \overline{H} denotes the bit-by-bit complement of H .

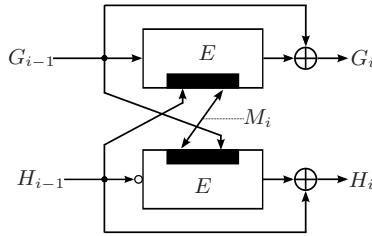


Fig. 1. The compression function F^{ADM} of ABREAST-DM, the small circle 'o' denotes a bit-by-bit complement

The compression function F^{ADM} requires two invocations of the block cipher E to produce an output. Note that these two block cipher invocations can be computed in parallel. Normally, E would be assumed to be AES-256 and therefore $n = 128$.

3.2 Security Results

Our discussion will result in proofs for the following bounds as given by Theorems 1 and 2.

Theorem 1. (*Collision Resistance*) Let $F := F^{\text{ADM}}$ as in Definition 2 and n, q be natural numbers with $q < 2^{n-2.58}$. Then

$$\mathbf{Adv}_F^{\text{COLL}}(q) \leq 18 \left(\frac{q}{2^{n-1}} \right)^2.$$

The following corollary explicitly states what this Theorem means for $n = 128$.

Corollary 1. For the compression function ABREAST-DM, instantiated with AES-256¹ any adversary asking less than $q = 2^{124.42}$ (backward or forward) oracle queries cannot usually find a collision.

Theorem 2. (*Preimage Resistance*) Let $F := F^{\text{ADM}}$ be as in Definition 2. For every $N' = 2^n - q$ and $q > 1$

$$\mathbf{Adv}_F^{\text{INV}}(q) \leq 2q/(N')^2.$$

The proof of Theorem 1 is a special case of Theorem 4 and will be omitted here. It is given in the full version of the paper [11], the proof of Theorem 2 is a simple corollary of Theorem 5 and will also be omitted. Using Theorem 1 and simple calculus, it is easy to see that the compression function is asymptotically optimal for $n \rightarrow \infty$ since

$$\lim_{n \rightarrow \infty} \mathbf{Adv}_F^{\text{COLL}}(q) = \frac{q^2}{2^{2n}}.$$

¹ Formally, we model the AES-256 block cipher as an ideal block cipher.

This bound is not meaningful even for $q \approx 2^{n-2.58}$ since $18q^2/2^{2n-2}$ would be larger than one. Note that the power of the arguments stems from the fact that we can tightly upper bound the number of compression functions that an adversary can compute given an upper bound of queries mounted by an adversary.

4 Security of Cyclic Hash Functions

In this section, we will generalize the definitions and techniques of the previous section.

4.1 Cyclic Compression Functions

We will now define the notion of a cyclic compression function F^{CYC} .

Definition 3. Let $(\Omega, *)$ be a group, $N = |\Omega|$. Let $F^{\text{CYC}} : \Omega^2 \times \{0, 1\}^b \rightarrow \Omega^2$ be a compression function such that $(G_i, H_i) = F^{\text{CYC}}(G_{i-1}, H_{i-1}, M_i)$ where $G_{i-1}, H_{i-1}, G_i, H_i \in \Omega$ and $M_i \in \{0, 1\}^b$, $b > 0$. Let $E \in BC(\Omega, \Omega \times \{0, 1\}^b)$ be a block cipher; ρ and σ permutations on the set $\Omega^2 \times \{0, 1\}^b$ and π^T, π^B permutations on Ω . Let $Z := (G_{i-1}, H_{i-1}, M_i) \in \Omega^2 \times \{0, 1\}^b$. Then $X^T, X^B \in \Omega$, $K^T, K^B \in \Omega \times \{0, 1\}^b$ such that $(X^T, K^T) = \rho(Z)$ and $(X^B, K^B) = \sigma(\rho(Z))$. Now F^{CYC} consists of a E as follows:

$$\begin{cases} G_i = E_{K^T}(M^T) * \pi^T(X^T) \\ H_i = E_{K^B}(M^B) * \pi^B(X^B) \end{cases}$$

where the computation leading to G_i is informally called the 'top row', and for H_i called 'bottom row'.

The compression function F^{CYC} is visualized in Figure 2.

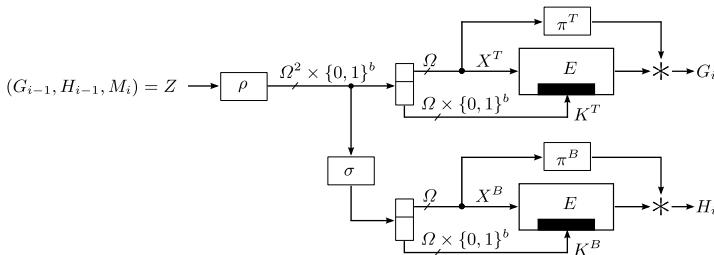


Fig. 2. Cyclic Compression Function $(G_i, H_i) = F^{\text{CYC}}(Z)$, $Z = (G_{i-1}, H_{i-1}, M_i)$

Since the properties of the permutation σ are highly relevant for the proof, we will discuss them now. The following Definitions 4, 5 are in some way the heart of this discussion. They lay the groundwork for defining cyclic double block length compression functions in the first place and provide for a main notion that we will use, the *order of an element* and the *order of a mapping*.

Definition 4. Let σ be a bijective mapping on a set \mathcal{S} where $\mathcal{S} := \Omega^2 \times \{0, 1\}^b$. Let ID be the identity mapping on \mathcal{S} . The function σ^k is defined as $\sigma^k := \sigma \circ \sigma^{k-1}$ for $k > 0$ and $\sigma^0 := \text{ID}$.

- (i) Fix some element $s \in \mathcal{S}$. The order of s is defined to be $|s| = \min_{r \geq 1}(\sigma^r(s) = s)$, i.e. $|s|$ is minimal (but > 0) such that $\sigma^{|s|}(s) = s$.
- (ii) If there is a $c \in \mathbb{N}_{\geq 1}$ such that $\forall \tilde{s} \in \mathcal{S} : |\tilde{s}| = c$, we say the order of the mapping σ , denoted by $|\sigma|$, is equal to c , i.e. $|\sigma| = c$. If there is no such c , then $|\sigma| := 0$.

Definition 5. Let F^{CYC} , ρ and σ be as in Definition 3. If $|\sigma| \geq 2$, then F^{CYC} is called a cyclic double block length (CDBL) compression function with cycle length $|\sigma|$.

Properties of CDBL Compression Functions. Now we will discuss the main properties of F^{CYC} . It is easy to see that a CDBL compression function with cycle length 1 is not reasonable as this would essentially F^{CYC} render a *single block length* compression function. A cycle length of 1 would imply $\sigma = \text{ID}$ and therefore $G_i = H_i$ for all $i \geq 1$. The values of the initial vector (G_0, H_0) are nonetheless free to be different. Single block length hash functions have already been thoroughly analyzed in [2,30,37].

Lemma 1. Let F^{CYC} be as in Definition 5.

- (i) Any oracle query for the top- or bottom row of F^{CYC} uniquely determines the oracle query in the bottom- or top row.
- (ii) The queries used in F^{CYC} for the bottom- and top row are always different, i.e. there are no fixed-points.

Proof. The proof of (i) is trivial and (ii) is a consequence of $|\sigma| > 1$. (□)

(Counter-)Examples. In the following, we will give some examples of known DBL constructions and discuss how they match Definition 3.

ABREAST-DM. To match Definition 3, we choose $\Omega = \{0, 1\}^n$, $b = n$, $\pi^T = \text{ID}$, $\pi^B(X) = \overline{X}$, $\rho(G, H, M) = \text{ID} = (G, H, M)$ and $\sigma(G, H, M) = (\overline{H}, M, G)$. It is easy to see that ABREAST-DM has a cycle length of $|\sigma| = c = 6$.

Hirose's FSE'06 Proposal. A description of F^{Hirose} is given in Appendix A. In this case we choose $\Omega = \{0, 1\}^n$, $b = n$, $\pi^T = \pi^B = \text{ID}$, $\rho = \text{ID}$ and $\sigma(G_{i-1}, H_{i-1}, M_i) = (G_{i-1} \oplus \text{const}, H_{i-1}, M_i)$ in order to map with the definition of F^{CYC} . It is easy to see that $|\sigma| = 2$. In [10, Appendix B] it is shown that for F^{Hirose} no adversary asking less than $2^{124.55}$ queries cannot find a collision probability greater than $1/2$ given that F^{Hirose} was instantiated with a $(128, 256)$ cipher as, e.g., AES-256.

TANDEM-DM. This compression function can be seen as a counter-example. A description of the compression function F^{TDM} is given in Appendix B. Its security was analyzed by Fleischmann et al. at FSE'09 [10] where it was shown that no adversary asking less than $2^{120.4}$ queries can find a collision with probability greater than $1/2$, given that F^{TDM} was instantiated with a $(128, 256)$ cipher as, e.g., AES-256. As the compression function feeds in the ciphertext of the top row into the bottom row, it cannot be represented as an instantiation of F^{CYC} since the definition does not allow any ciphertext feedback.

4.2 Security Results

Our discussion will result in proofs for the following bounds as given by Theorems 3, 4 and 5.

Theorem 3. (*Collision Resistance for $|\sigma| = 2$*) Let $F := F^{\text{CYC}}$ be a cyclic compression function with cycle length $c = |\sigma| = 2$ as in Definition 5. If $\pi^T = \pi^B$, then $a = 1$, else $a = 2$. Then, for any $q > 1$ and $2q < N$,

$$\mathbf{Adv}_F^{\text{COLL}}(q) \leq \frac{2aq^2}{(N - 2q)^2} + \frac{2q}{N - 2q}.$$

Theorem 4. (*Collision Resistance for $|\sigma| > 2$*) Let $F := F^{\text{CYC}}$ be a cyclic compression function with cycle length $c = |\sigma| > 2$ as in Definition 5. Then, for any $q > 1$ and $cq < N$,

$$\mathbf{Adv}_F^{\text{COLL}}(q) \leq \frac{c^2}{2} \left(\frac{q}{N - cq} \right)^2.$$

Theorem 5. (*Preimage Resistance*) Let $F := F^{\text{CYC}}$ be a cyclic compression function as in Definition 5. Then, for any $q > 1$ and $q < N$,

$$\mathbf{Adv}_F^{\text{INV}}(q) \leq 2q/(N - q)^2.$$

Applications and examples of these Theorems will be discussed in Section 5. The proof of Theorem 3 is given in Appendix C. The proof of Theorem 5 is essentially due to Fleischmann et. al. [10, Thm. 2] and can be found in Appendix D. The proof of Theorem 4 is given in Section 4.3.

Remarks. Naturally, the question arises what to conclude if only a 'generalization' of Definition 4 is applicable, i.e. if not all elements s share the same order. In theory, one can imagine a case where some elements have an order of 2, some of 3 and some have an order of 4. Since for any $|\sigma| > 2$, Theorem 4 states a monotonically decreasing upper bound, we can easily assume the 'worst case' bound and apply this theorem for this maximum cycle length *if* all elements have a minimum order of 3. If some elements also have an order of 2, one has to take into account the result given in Theorem 3. The final bound for the advantage can be finally obtained by taking the maximum of all bounds that result from the order of the elements. Simply, this is due to the fact that a query that results in a 'query cycle' of length a cannot be used in any other query cycle since any query cycle is a closed set.

4.3 Collision Resistance – Proof of Theorem 4

Overview. In this section we will generalize the proof of Theorem 1. We will analyze if the queries made by the adversary contain the means for constructing a collision of the compression function F^{CYC} . We upper bound the probability of the adversary making a query that can be used as the *final* query to complete a collision.

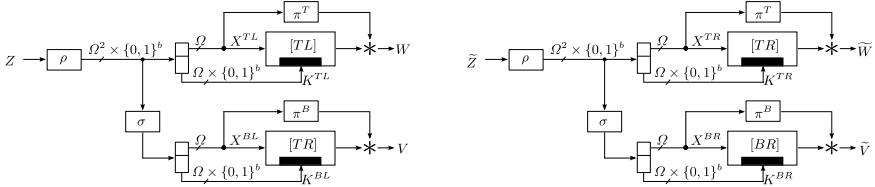


Fig. 3. Notations used for a collision of CYCLIC-DM: $\text{COLL}^{\text{CYC}}(\mathcal{Q})$, in this case $W = \tilde{W}$ and $V = \tilde{V}$ but $Z \neq \tilde{Z}$

The cycle in CYCLIC-DM. Assume that the adversary mounts a query $Q_{ci} = (X_{ci}, K_{ci}, Y_{ci})$, where $X_{ci}, Y_{ci} \in \Omega$, $K_{ci} \in \Omega \times \{0,1\}^b$, $Y_{ci} = E_{K_{ci}}(X_{ci})$. The query index $c \cdot i$ for the i -th query of the adversary is – similar as in the case of ABREAST-DM – due the the $c - 1$ free queries the adversary is given for any mounted query. First assume that the query is used in the top row. Let $U_1 = (X_{ci}, K_{ci}) \in \Omega \times (\Omega \times \{0,1\}^b) = \Omega^2 \times \{0,1\}^b$ and $U_2 = (X_{ci+1}, K_{ci+1}) = \sigma(U_1)$ where $X_{ci+1} \in \Omega$ and $K_{ci+1} \in \Omega \times \{0,1\}^b$. The adversary is given for free the corresponding query in the bottom row $Q_{ci+1} = (X_{ci+1}, K_{ci+1}, Y_{ci+1})$, $Y_{ci+1} = E_{K_{ci+1}}(X_{ci+1})$. Given these two queries, the adversary is able to compute one output of the compression function $(W_1, V_1) = F^{\text{CYC}}(\rho^{-1}(U_1))$. The adversary can ‘reuse’ the query Q_{ci+1} in the top row as a starting point to compute a new result of F^{CYC} . We now give the adversary for free the corresponding bottom row query, Q_{ci+2} , assuming that Q_{ci+1} is used in the top row. For this query Q_{ci+2} , we have $U_3 = (X_{ci+2}, K_{ci+2}) = \sigma(U_2)$ where $X_{ci+2} \in \Omega$ and $K_{ci+2} \in \Omega \times \{0,1\}^b$ and $Y_{ci+2} = E_{K_{ci+2}}(X_{ci+2})$. Our main observation is that $U_3 = \sigma(U_2) = \sigma^2(U_1)$.

Let $c = |\sigma|$ denote the cycle length of F^{CYC} . This process can be continued. The adversary is given for free the queries $Q_{ci+3}, \dots, Q_{ci+c-1}$ as is shown in Table 1 in more detail. A cycle is formed since $U_c = \sigma(U_{c-1}) = \dots = \sigma^c(U_1) = U_1$ and therefore $Q_{ci+c} = Q_{ci}$. The queries forming the cycle are visualized in Figure 4.

Details. Fix a set Ω , numbers b, q and an adversary \mathcal{A} asking q backward and forward queries to its oracle in total. Let $\text{COLL}^{\text{CYC}}(\mathcal{Q})$ be the event that the adversary is able to construct a collision of F^{CYC} using the queries in \mathcal{Q} . The term ‘last query’ means the latest query made by the adversary and is always given index $c \cdot i$ and denoted as Q_{ci} . We will examine the adversary mounted queries ($d = 0$) and the free queries ($d = 1, 2, \dots, c - 1$), $(X_{ci+d}, K_{ci+d}, ?)_{fwd}$ or $(?, K_{ci+d}, Y_{ci+d})_{bwd}$ one at a time as the adversary gets hold of them. A query

Table 1. Starting with query ci , $(X_{ci}, K_{ci}, ?)_{fwd}$ or $(?, K_{ci}, Y_{ci})_{bwd}$, the adversary is given $c - 1$ forward queries $ci + 1, ci + 2, \dots, ci + c - 1$ for free. In total, he is able to compute c results of F^{CYC} by using these c queries. The notations used in the table are given in the text.

$F^{\text{CYC}}(\cdot)$	Query #	Plaintext	Chaining Value
$\rho^{-1}(U_1)$	ci (*)	X_{ci}	$W_1 = Y_{ci} * \pi^T(X_{ci})$
	$ci + 1$	X_{ci+1}	$V_1 = Y_{ci+1} * \pi^B(X_{ci+1})$
$\rho^{-1}(\sigma(U_1))$	$(ci + 1)$	X_{ci+1}	$W_2 = Y_{ci+1} * \pi^T(X_{ci+1})$
	$ci + 2$	X_{ci+2}	$V_2 = Y_{ci+2} * \pi^B(X_{ci+2})$
$\rho^{-1}(\sigma^2(U_1))$	$(ci + 2)$	X_{ci+2}	$W_3 = Y_{ci+2} * \pi^T(X_{ci+2})$
	$ci + 3$	X_{ci+3}	$V_3 = Y_{ci+3} * \pi^B(X_{ci+3})$
\vdots	\vdots	\vdots	\vdots
$\rho^{-1}(\sigma^{c-2}(U_1))$	$(ci + c - 2)$	X_{ci+c-2}	$W_{c-1} = Y_{ci+c-2} * \pi^T(X_{ci+c-2})$
	$ci + c - 1$	X_{ci+c-1}	$V_{c-1} = Y_{ci+c-1} * \pi^B(X_{ci+c-1})$
$\rho^{-1}(\sigma^{c-1}(U_1))$	$(ci + c - 1)$	X_{ci+c-1}	$W_c = Y_{ci+c-1} * \pi^T(X_{ci+c-1})$
	(ci)	X_{ci}	$V_c = Y_{ci} * \pi^B(X_{ci})$

$Q_m = (X_m, K_m, Y_m)$ is successful, if it can be used to form a collision using other queries contained in the query history \mathcal{Q}_m as indicated in Figure 3.

We now upper bound $\Pr[\text{COLL}^{\text{CYC}}(\mathcal{Q})]$ by exhibiting predicates $\text{WIN}_0(\mathcal{Q}), \dots, \text{WIN}_{q-1}(\mathcal{Q})$ such that $\text{COLL}^{\text{CYC}} \implies \text{WIN}_0(\mathcal{Q}) \vee \dots \vee \text{WIN}_{q-1}(\mathcal{Q})$. Then, $\Pr[\text{COLL}^{\text{CYC}}(\mathcal{Q})] \leq \text{WIN}_0(\mathcal{Q}) + \dots + \text{WIN}_{q-1}(\mathcal{Q})$. Since the adversary mounts q queries in total we informally say that $\text{WIN}_i(\mathcal{Q})$, $0 \leq i \leq q - 1$ holds if the adversary finds a collision after mounting the i -th query, $0 \leq i \leq q - 1$, using at least two of the following queries $Q_{ci}, \dots, Q_{ci+c-1}$ conditioned on the fact that the adversary has not been successful before. For simplicity, we assume again that the free queries are always given in 'ascending' order as given in Table 1.

Definition 6. We say that a pair of queries (a, b) is successful in \mathcal{Q}_c , if the query Q_a is used in the top row, Q_b in the bottom row in the computation of a compression function F^{CYC} and there exists a pair of queries $Q_j, Q_k \in \mathcal{Q}_c$ such that a collision of F^{CYC} can be computed:

$$\pi^T(X_a) \oplus Y_a = \pi^T(X_j) \oplus Y_j \quad \text{and} \quad \pi^B(X_b) \oplus Y_b = \pi^B(X_k) \oplus Y_k.$$

Definition 7. Let $d = 0, \dots, c - 1$, $d' = d + 1 \bmod c$, $\tilde{d} = \max(d, d')$. We say $\text{COLLFIT}_i^d(\mathcal{Q})$ if (i) the pair of queries $(ci + d, ci + d')$ is successful in $\mathcal{Q}_{ci+\tilde{d}}$ and (ii) the adversary had not been successful for $0 \leq t \leq d - 1$: $\neg \text{COLLFIT}_i^t(\mathcal{Q})$.

The predicates $\text{WIN}_i(\mathcal{Q})$ are defined as follows:

Definition 8

$$\text{WIN}_i(\mathcal{Q}) = \neg \left(\bigvee_{0 \leq j \leq i-1} \text{WIN}_j(\mathcal{Q}) \right) \wedge (\text{COLLFIT}_i^1(\mathcal{Q}) \vee \dots \vee \text{COLLFIT}_i^c(\mathcal{Q}))$$

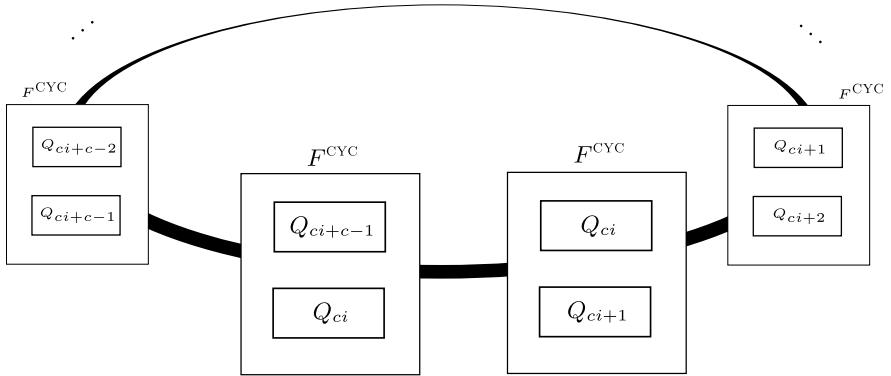


Fig. 4. A Cycle: An adversary uses the c queries to compute the complete output of c compression functions F^{CYC}

We now show that our case analysis is complete.

Lemma 2. $\text{COLL}^{\text{CYC}}(\mathcal{Q}) \implies \text{WIN}_0(\mathcal{Q}) \vee \dots \vee \text{WIN}_{q-1}(\mathcal{Q})$.

This proof is omitted. It is given in the full version of the paper.

Since $\Pr[\text{COLL}^{\text{CYC}}(\mathcal{Q})] \leq \sum_{j=0}^{q-1} \text{WIN}_j(\mathcal{Q})$ it follows that

$$\Pr[\text{COLL}^{\text{CYC}}(\mathcal{Q})] \leq \sum_{i=0}^{q-1} \sum_{d=0}^{c-1} \text{COLLFI}_i^d(\mathcal{Q}). \quad (1)$$

We will now upper bound $\Pr[\text{COLLFI}_i^d(\mathcal{Q})]$.

Lemma 3. Let $0 \leq i \leq q-1$ and $0 \leq d \leq c-1$. Then

$$\Pr[\text{COLLFI}_i^d(\mathcal{Q})] \leq \frac{ci}{(N-ci)^2}.$$

Proof. Let $d' = d + 1 \bmod c$. The output of the compression function F^{CYC} , (W, V) , is uniquely determined by the queries $Q_{ci+d} = (X_{ci+d}, K_{ci+d}, Y_{ci+d})$ and $Q_{ci+d'} = (X_{ci+d'}, K_{ci+d'}, Y_{ci+d'})$,

$$W = Y_{ci+d} * \pi^T(X_{ci+d}) \quad \text{and} \quad V = Y_{ci+d'} * \pi^B(X_{ci+d'}).$$

Note that both W and V are randomly determined by the answer of the oracle as in discussed in the full version of the paper. The use of permutations π^T and π^B do not change these arguments.

To form a collision, two queries Q_j, Q_k are needed that can be chosen from at most $c(i+1)$ queries in $\mathcal{Q}_{c(i+1)-1}$. The adversary can use them to compute the output of $< c(i+1)$ compression functions F^{CYC} . Therefore,

$$\Pr[\text{COLLFI}_i^d(\mathcal{Q})] \leq \frac{c(i+1)}{(N-c(i+1))^2}. \quad (\square)$$

Using (1) we get the following upper bound for any $q \geq 1$ and $N > cq$

$$\begin{aligned} \Pr[\text{COLL}^{\text{CYC}}(\mathcal{Q})] &\leq \sum_{i=0}^{q-1} \sum_{d=0}^{c-1} \frac{c(i+1)}{(N - c(i+1))^2} \leq \sum_{i=1}^q \sum_{d=0}^{c-1} \frac{ci}{(N - ci)^2} \\ &\leq \sum_{i=1}^q \frac{c^2 i}{(N - ci)^2} \leq \frac{c^2 \cdot q^2 \cdot \frac{1}{2}}{(N - cq)^2} \leq \frac{c^2}{2} \left(\frac{q}{N - cq} \right)^2. \end{aligned}$$

This completes our proof of Theorem 4. ■

5 Building More Efficient and Secure DBL Compression Functions

Table 2 contains all efficient double block length compression functions known from literature that have provably birthday-type collision resistance. Except for TANDEM-DM, they are all in the class of CYCLIC-DM. The threshold value ' α ' gives the least amount of queries any adversary must ask in order to have more than a chance of 0.5 in finding a collision for the compression function assuming a plain-/ciphertext length of 128 bit of the block cipher.

5.1 Add/k-DM (Cycle Length 2^k)

Luckily, there does exist a very elegant and efficient method to instantiate a compression function with cycle length $c = 2^k$ for any $k \geq 1$. This construction is very similar to Hirose's FSE'06 proposal. It is shown in Figure 5 and formally given in Definition 9.

Table 2. List of all known efficient double block length compression functions. 'Common Key' indicates whether both block cipher calls use the same key for their encryption operations, 'Parallel' indicates whether both encryption operations are independent of each other and can therefore be computed in parallel.

Cycle length	Threshold α	Example(s)	Common Key	Parallel
2	$2^{124.55}$	Hirose FSE'06 [15] ADD/1-DM, Section 5.1	yes yes	yes yes
3	$2^{125.42}$	Section 5.2	yes	yes
4	$2^{125.0}$	ADD/2-DM, Section 5.1	yes	yes
4 ¹	$2^{125.0}$	Lee/Kwon, [18]	no	yes
6	$2^{124.42}$	ABREAST-DM, Section 3.2	no	yes
2^k ($k \geq 2$)	2^{127-k}	ADD/k-DM, Section 5.1	yes	yes
—	$2^{120.4}$	TANDEM-DM, FSE'09 [10]	no	no

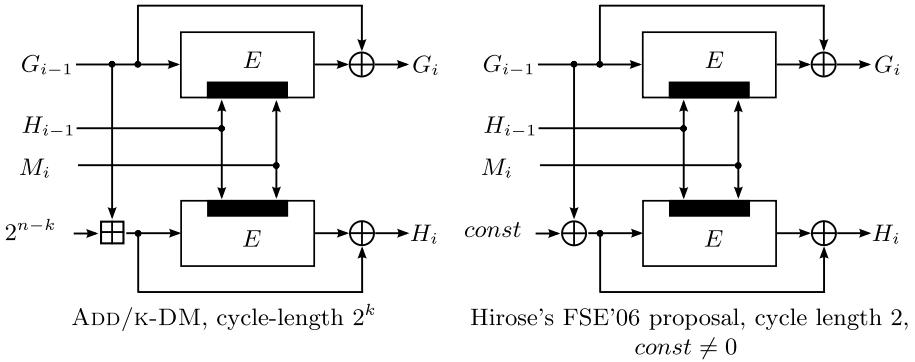


Fig. 5. Left: Cyclic Compression Function with cycle length 2^k , $k > 1$. Right: (for comparison) Hirose's FSE'06 proposal with a cycle length of 2.

Definition 9. Let $F^{\text{ADD/K}} : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ be a compression function such that $(G_i, H_i) = F^{\text{ADD/K}}(G_{i-1}, H_{i-1}, M_i)$ where $G_i, H_i, M_i \in \{0, 1\}^n$ and let $k \in \mathbb{N}$ such that $1 \leq k < n$. $F^{\text{ADD/K}}$ is built upon a $(n, 2n)$ -block cipher E as follows:

$$\begin{aligned} G_i &= E(G_{i-1}, H_{i-1} | M_i) \oplus G_{i-1} \\ H_i &= E(G_{i-1} \boxplus 2^{n-k}, H_{i-1} | M_i) \oplus (G_{i-1} \boxplus 2^{n-k}), \end{aligned}$$

where $|$ represents concatenation. The symbol ' \boxplus ' denotes an addition modulo 2^n .

Lemma 4. The compression function $F^{\text{ADD/K}}$ is in CYCLIC-DM and has a cycle length of 2^k .

Proof. To map with Definition 5 we let $\Omega = \{0, 1\}^n$, $b = n$, $\pi^T = \pi^B = \text{ID}$, $\rho = \text{ID}$ and $\sigma : \{0, 1\}^n \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^n \times \{0, 1\}^{2n}$ is chosen as $\sigma(M, K) = (M \boxplus 2^{n-k}, K)$. The claim follows since

$$\underbrace{(\sigma \circ \dots \circ \sigma)}_{2^k \text{ times}}(M, K) = (M \boxplus 2^k \cdot 2^{n-k}, K) = (M, K).$$

■

Therefore we can apply Theorem 3 for $k = 1$ or Theorem 4 if $k \geq 2$.

Corollary 2. No adversary asking less than 2^{n-k-1} queries can have more than a chance of 0.5 in finding a collision for the compression function $F := F^{\text{ADD/K}}$ for any $1 < k < n$.

Proof. This result can be obtained by using a simple calculation. As the cycle length c is equal to 2^k (Lemma 4), it follows using Theorem 4

¹ The bounds given in [18] are slightly weaker (but numerically comparable) than we would have received by our Theorem 4. Also, for this construction, we can easily use Theorem 4 and result in the stated bound of 2^{125} .

$$\mathbf{Adv}_F^{\text{COLL}}(q) = \frac{2^{2k}}{2} \left(\frac{q}{2^{n-1}} \right)^2.$$

By applying $\mathbf{Adv}_F^{\text{COLL}}(q) = 0.5$ and solving after q one obtains $q(k) = \sqrt{2^{2n-2k-2}} = 2^{n-k-1}$. \blacksquare

Using $n = 128$, as for AES-256, we can derive without effort that no adversary asking less than 2^{122} queries can have more than a chance of 0.5 in finding a collision for the compression function $F^{\text{ADD}/5}$. The compression function $F^{\text{ADD}/5}$ has a cycle length of $2^5 = 32$.

5.2 Cube-DM (Cycle Length = 3)

The 'most optimal' result in terms of security – at least in the class CYCLIC-DM – can be achieved by using a compression function that has a cycle length 3. The approach is slightly different compared to ADD/ κ -DM as neither additions modulo 2^n nor XOR can be used to create a permutation σ with $|\sigma| = 3$. The guiding idea to use a message space Ω such that $|\Omega|$ is evenly divisible by three. This construction is visualized in Figure 6 and given in Definition 11.

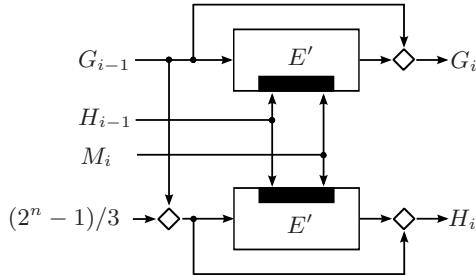


Fig. 6. CUBE-DM, a compression function with cycle length $|\sigma| = 3$, the symbol ' \diamond ' denotes an addition modulo $2^n - 1$

Definition 10. Let $E : \{0, 1\}^n \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ be a block cipher with n -bit plain-/ciphertext and $2n$ -bit key. Let $\Omega = \{0, 1\}^n - \{1^n\}$, i.e. $|\Omega| = 2^n - 1$. The block cipher $E' : \Omega \times (\Omega \times \{0, 1\}^n) \rightarrow \Omega$, where $\Omega \times \{0, 1\}^n$ is the key space is defined as

$$E'_K(X) = \begin{cases} E_K(X), & \text{if } E_K(X) \neq 1^n, \\ E_K(E_K(X)), & \text{else.} \end{cases}$$

This definition of the block cipher E' ensures that $E'_K(X) \in \Omega$ for any value of $X \in \Omega$: since E is a permutation, it follows that E' is a permutation. It is easy to see that, for n even, $|\Omega|$ is divisible by three since

$$|\Omega| \bmod 3 = 2^n - 1 \bmod 3 = (2 \cdot 2)^{n'} - 1 \bmod 3 = 0 \bmod 3. \quad (2)$$

Definition 11. Let $\Omega = \{0, 1\}^n - \{1^n\}$, $N = |\Omega| = 2^n - 1$. Let $F^{\text{CUBE}} : \Omega^2 \times \{0, 1\}^n \rightarrow \Omega^2$ be a compression function such that $(G_i, H_i) = F^{\text{CUBE}}(G_{i-1}, H_{i-1}, M_i)$ where $G_{i-1}, H_{i-1}, G_i, H_i \in \Omega$ and $M_i \in \{0, 1\}^b$. Furthermore, let $\text{const} = (2^n - 1)/3$ and ' \diamond ' be the addition modulo $2^n - 1$. Now F^{CUBE} is built upon a block cipher E' as in Definition 10:

$$\begin{aligned} G_i &= E_{H_{i-1}|M_i}(G_{i-1}) \diamond G_{i-1} \\ H_i &= E_{H_{i-1}|M_i}(G_{i-1} \diamond \text{const}) \diamond (G_{i-1} \diamond \text{const}), \end{aligned}$$

where ' $|$ ' represents concatenation.

Lemma 5. The compression function F^{CUBE} is in CYCLIC-DM and has a cycle length of 3.

Proof. To map with Definition 5, we choose $\rho = \text{ID}$, $\pi^T = \pi^B = \text{ID}$, $b = n$ and $\sigma : \Omega^2 \times \{0, 1\}^n \rightarrow \Omega^2 \times \{0, 1\}^n$ is chosen to be $\sigma(M, K) = (M \diamond (2^n - 1)/3, K)$. The claim follows using (2) and

$$(\sigma \circ \sigma \circ \sigma)(M, K) = (M \diamond 3 \cdot \frac{2^n - 1}{3} \bmod 2^n - 1, K) = (M, K). \quad \blacksquare$$

The threshold value of $\alpha = 2^{125.42}$ as given in Table 2 follows with Theorem 4. Note that the operation ' \diamond ' is trivially efficient since a simple 'if' and an 'addition' suffice for implementation. Also, the implementation of E' is not assumed to cost any measurable performance.

6 Discussion and Conclusion

In this paper, we have investigated the security of ABREAST-DM, a DBL compression function based on a $(n, 2n)$ block cipher that was presented at EU-ROCRYPT'92. In the ideal cipher model, we showed that this construction has birthday type collision resistance: any adversary asking less than $2^{124.42}$ queries cannot find a collision with probability greater than $1/2$. The proof technique was generalized to a class of double block length compression functions CYCLIC-DM and rigorous security bounds in terms of collision resistance and preimage resistance were given for this construction. The security of such constructions mainly depends on a parameter, the *cycle length*. Several new double block length compression functions were presented, some of them (CUBE-DM and ADD/4-DM) both have a higher security guarantee in terms of collision resistance than the best known DBL compression functions known in literature today.

Our work not only adds to the understanding of block cipher based compression functions but also introduces generic construction principles for them. For these constructions we also have provided a rigorous security analysis in terms of collision resistance and a (relatively weak) bound for preimage resistance. Somewhat interestingly, one of the implicit results seems to be that, given the right construction, the security does *not* depend on whether the two block ciphers are fed in with different keys – at least in the case for $(n, 2n)$ block ciphers.

It is clear that there still needs to be a lot of research done in the field of block cipher based hash functions. There do not exist completely satisfying constructions and/or security proofs for, *e.g.*, MDC-2/4. More generally, there has to be added a lot to our understanding, especially for constructions that are more efficient, *e.g.* have rate 1, or use other building blocks such as, *e.g.*, (n, n) block ciphers.

Acknowledgements

We would like to thank T. Shrimpton for pointing out the idea of ADD/CUBE-DM. Also, we would like to thank the anonymous referees for helpful comments.

References

1. ANSI. ANSI X9.31:1998: Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA). American National Standards Institute, pub-ANSI:adr (1998)
2. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
3. Bosselaers, A., Preneel, B.: Integrity Primitives for Secure Information Systems. In: Bosselaers, A., Preneel, B. (eds.) RIPE 1992. LNCS, vol. 1007, p. 239. Springer, Heidelberg (1995)
4. Meyer, C., Matyas, S.: Secure program load with manipulation detection code (1988)
5. Coppersmith, D., Pilpel, S., Meyer, C.H., Matyas, S.M., Hyden, M.M., Oseas, J., Bracht, B., Schilling, M.: Data authentication using modification detection codes based on a public one way encryption function. U.S. Patent No. 4,908,861, March 13 (1990)
6. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
7. Dean, R.D.: Formal aspects of mobile code security. PhD thesis, Princeton, NJ, USA, Adviser-Andrew Appel (1999)
8. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD-5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
9. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 210–224. Springer, Heidelberg (1991)
10. Fleischmann, E., Gorski, M., Lucks, S.: On the Security of Tandem-DM. In: Robshaw [33], pp. 84–103
11. Fleischmann, E., Gorski, M., Lucks, S.: Security of cyclic double block length hash functions including abreast-dm. Cryptology ePrint Archive, Report 2009/261 (2009), <http://eprint.iacr.org/>
12. Dobbertin, H.: The status of MD5 after a recent attack (1996)
13. Hattori, M., Hirose, S., Yoshida, S.: Analysis of double block length hash functions. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 290–302. Springer, Heidelberg (2003)

14. 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)
15. Hirose, S.: Some Plausible Constructions of Double-Block-Length Hash Functions. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
16. Hohl, W., Lai, X., Meier, T., Waldvogel, C.: Security of iterated hash functions based on block ciphers. In: Stinson [39], pp. 379–390
17. ISO/IEC. ISO DIS 10118-2: Information technology - Security techniques - Hash-functions, Part 2: Hash-functions using an n-bit block cipher algorithm. First released in 1992 (2000)
18. J. Lee, D. Kwon. The Security of Abreast-DM in the Ideal Cipher Model. Cryptology ePrint Archive, Report 2009/225 (2009), <http://eprint.iacr.org/>
19. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer [6], pp. 474–490
20. Kilian, J., Rogaway, P.: How to protect des against exhaustive key search. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 252–267. Springer, Heidelberg (1996)
21. Knudsen, L.R., Lai, X., Preneel, B.: Attacks on fast double block length hash functions. *J. Cryptology* 11(1), 59–72 (1998)
22. Knudsen, L.R., Muller, F.: Some attacks against a double length hash proposal. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 462–473. Springer, Heidelberg (2005)
23. 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)
24. Rabin, M.: Digitalized Signatures (1978)
25. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press, Boca Raton (1996)
26. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
27. Nandi, M., Lee, W.I., Sakurai, K., Lee, S.-J.: Security analysis of a 2/3-rate double length compression function in the black-box model. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 243–254. Springer, Heidelberg (2005)
28. NIST National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard (April 1995), <http://csrc.nist.gov>
29. NIST National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard (April 1995), <http://csrc.nist.gov>
30. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson [39], pp. 368–378
31. Rivest, R.L.: RFC 1321: The MD5 Message-Digest Algorithm. Internet Activities Board (April 1992)
32. Rivest, R.L.: The md4 message digest algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
33. Robshaw, M.J.B. (ed.): FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
34. Rogaway, P., Steinberger, J.P.: Constructing cryptographic hash functions from fixed-key blockciphers. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 433–450. Springer, Heidelberg (2008)
35. Rogaway, P., Steinberger, J.P.: Security/efficiency tradeoffs for permutation-based hashing. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 220–236. Springer, Heidelberg (2008)

36. Satoh, T., Haga, M., Kurosawa, K.: Towards secure and fast hash functions. TIE-ICE: IEICE Transactions on Communications/Electronics/Information and Systems (1999)
37. Stam, M.: Blockcipher Based Hashing Revisited. In: Robshaw [33]
38. 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)
39. Stinson, D.R. (ed.): CRYPTO 1993. LNCS, vol. 773. Springer, Heidelberg (1994)
40. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions md4 and ripemd. In: Cramer [6], pp. 1–18
41. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full sha-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)

A Hirose's FSE'06 Proposal of a DBL Compression Function

At FSE'06, Hirose [15] proposed the DBL compression function F^{Hirose} (Definition 12 and Figure 7). He proved that when F^{Hirose} is employed in an iterated hash function H , then no adversary asking less than $2^{125.7}$ queries can have more than a chance of 0.5 in finding a collision for $n = 128$.

Definition 12. Let $F^{Hirose} : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ be a compression function such that $(G_i, H_i) = F^{Hirose}(G_{i-1}, H_{i-1}, M_i)$ where $G_i, H_i, M_i \in \{0, 1\}^n$. F^{Hirose} is built upon a $(n, 2n)$ block cipher E as follows:

$$G_i = F_T(G_{i-1}, H_{i-1}, M_i) = E(G_{i-1}, H_{i-1}|M_i) \oplus G_{i-1}$$

$$H_i = F_B(G_{i-1}, H_{i-1}, M_i) = E(G_{i-1} \oplus \text{const}, H_{i-1}|M_i) \oplus G_{i-1} \oplus \text{const},$$

where ' $|$ ' represents concatenation and $\text{const} \in \{0, 1\}^n - \{0^n\}$ is a constant.

A visualization of this compression function is given in Figure 7.

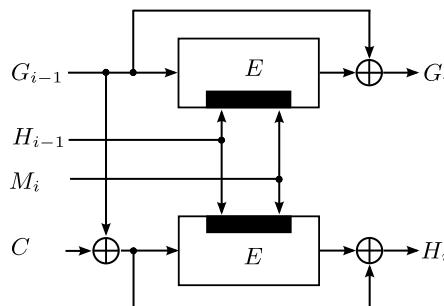


Fig. 7. The compression function F^{Hirose} , E is an $(n, 2n)$ block cipher

B A Non-cyclic Compression Function: Tandem-DM

The TANDEM-DM compression function was proposed by Lai and Massey at EUROCRYPT'92 [23]. It uses two cascaded Davies-Meyer [2] schemes. The compression function is illustrated in Figure 8 and is formally given in Definition 13.

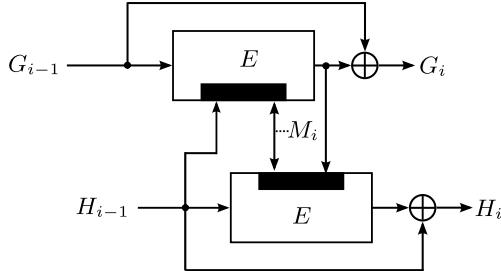


Fig. 8. The compression function TANDEM-DM F^{TDM} where E is an $(n, 2n)$ block cipher, the black rectangle inside the cipher rectangle indicates the key input

Definition 13. Let $F^{TDM} : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ be a compression function such that $(G_i, H_i) = F^{TDM}(G_{i-1}, H_{i-1}, M_i)$ where $G_i, H_i, M_i \in \{0, 1\}^n$. F^{TDM} is built upon an $(n, 2n)$ block cipher E as follows:

$$\begin{aligned} W_i &= E(G_{i-1}, H_{i-1} | M_i) \\ G_i &= F_T(G_{i-1}, H_{i-1}, M_i) = W_i \oplus G_{i-1} \\ H_i &= F_B(G_{i-1}, H_{i-1}, M_i) = E(H_{i-1}, M_i | W_i) \oplus H_{i-1}. \end{aligned}$$

C Collision Resistance – Proof of Theorem 3

Due to the special structure of the compression function in the case of $c = |\sigma| = 2$, the following definition is useful for the proof.

Definition 14. A pair of distinct inputs $(G_{i-1}, H_{i-1}, M_i), (G'_{i-1}, H'_{i-1}, M'_i)$ to F^{Cyc} is called a matching pair if $(G'_{i-1}, H'_{i-1}, M'_i) = (\rho^{-1} \circ \sigma \circ \rho)(G_{i-1}, H_{i-1}, M_i)$. Otherwise they are called a non-matching pair.

Fix numbers n, q and an adversary \mathcal{A} asking q backward and forward queries to its oracle E in total. Note that we will assume throughout this proof that the cycle length $c = |\sigma| = 2$. All queries to the oracle are saved in a query history \mathcal{Q} . Let $COLL^{Cyc-2}$ be the event that the adversary is able to construct a collision of F^{Cyc} in this case. We will examine the queries one at a time as they come in; the latest query made by the adversary, his i -th query, will always be given index $2i$, and is denoted as Q_{2i} . Say the query $Q_{2i} = (X_{2i}, K_{2i}, Y_{2i})$ is a forward or backward query mounted by the adversary and assume that Q_{2i} is used in the

top row. As two queries are required for the computation of F^{CYC} we will give the adversary the bottom row query for free. This query is uniquely determined by its plaintext X_{2i+1} and key K_{2i+1} component as follows:

$$(X_{2i+1}, K_{2i+1}) = \rho(\sigma(\rho^{-1}(X_{2i}, K_{2i})))$$

and the adversary is given the ciphertext $Y_{2i+1} = E_{K_{2i+1}}(X_{2i+1})$. If the adversary uses the query Q_{2i} in the bottom row, we give him the top row query for free:

$$(X_{2i+1}, K_{2i+1}) = \rho(\sigma^{-1}(\rho^{-1}(X_{2i}, K_{2i})))$$

and the adversary is given the ciphertext $Y_{2i+1} = E_{K_{2i+1}}(X_{2i+1})$ in this case. Since $\sigma^2 = \text{ID}$ it follows that $\sigma = \sigma^{-1}$ it follows that in either case, the adversary is given the *same* free query, *i.e.* the input to the other query is always uniquely determined using one and the same computation.

Now assume for the simplicity of the following argument that the query Q_{2i} is used in the top row and Q_{2i+1} in the bottom row. As $G_i = Y_{2i} \oplus \pi^T(X_{2i})$ depends both on the plaintext and the ciphertext of E and one of them is fixed by query and the other is determined randomly by the oracle it follows that G_i is randomly determined by that answer. Using the same argument, $H_i = Y_{2i+1} \oplus \pi^T(X_{2i+1})$ is also randomly determined by the other answer.

For any $2 \leq i \leq q$ let C_i be the event that a colliding pair of non-matching inputs are found for F^{CYC} with the i -th pair of queries. Namely, it is the event that for some $i' < i$

$$F^{\text{CYC}}(\rho^{-1}(X_{2i}, K_{2i})) \in \{F^{\text{CYC}}(\rho^{-1}(X_{2i'}, K_{2i'})), F^{\text{CYC}}(\rho^{-1}(X_{2i'+1}, K_{2i'+1}))\}$$

or

$$F^{\text{CYC}}(\rho^{-1}(X_{2i+1}, K_{2i+1})) \in \{F^{\text{CYC}}(\rho^{-1}(X_{2i'}, K_{2i'})), F^{\text{CYC}}(\rho^{-1}(X_{2i'+1}, K_{2i'+1}))\}$$

This condition is equivalent to

$$(Y_{2i} * \pi^T(X_{2i}), Y_{2i+1} * \pi^B(X_{2i+1})) = (Y_{2i'} * \pi^T(X_{2i'}), Y_{2i+1} * \pi^B(X_{2i'+1})) \quad \text{or} \quad (3)$$

$$(Y_{2i} * \pi^T(X_{2i}), Y_{2i+1} * \pi^B(X_{2i+1})) = (Y_{2i'+1} * \pi^T(X_{2i'+1}), Y_{2i} * \pi^B(X_{2i'})) \quad \text{or} \quad (4)$$

$$(Y_{2i+1} * \pi^T(X_{2i+1}), Y_{2i} * \pi^B(X_{2i})) = (Y_{2i'} * \pi^T(X_{2i'}), Y_{2i+1} * \pi^B(X_{2i'+1})) \quad \text{or} \quad (5)$$

$$(Y_{2i+1} * \pi^T(X_{2i+1}), Y_{2i} * \pi^B(X_{2i})) = (Y_{2i'+1} * \pi^T(X_{2i'+1}), Y_{2i} * \pi^B(X_{2i'})). \quad (6)$$

Note that (3) is equal to (6) and (4) is equal to (5) if $\pi^T = \pi^B$. In this case, it follows that for $2q < N$

$$\Pr[C_i] \leq \frac{2(i-1)}{(N - (2i-2))(N - (2i-1))} \leq \frac{2q}{(N - 2q)^2}. \quad (7)$$

Assuming $\pi^T \neq \pi^B$ we obtain

$$\Pr[C_i] \leq \frac{4(i-1)}{(N - (2i-2))(N - (2i-1))} \leq \frac{4q}{(N - 2q)^2}. \quad (8)$$

For unifying the treatment of these two cases, we set $a = 1$ if $\pi^T = \pi^B$ and $a = 2$ otherwise. Let C be the event that a colliding pair of non-matching inputs are found for F^{Cyc} with q (pairs) of queries. Then,

$$\Pr[C] \leq \sum_{i=2}^q \Pr[C_j] \leq \sum_{i=2}^q \frac{2q \cdot a}{(N - 2q)^2} \leq \frac{2aq^2}{(N - 2q)^2}.$$

Now, let with the i -th query be \hat{C}_i the event such that a colliding pair of matching inputs is found for F^{Cyc} . It follows, that

$$\Pr[\hat{C}_i] \leq \frac{2}{N - 2i}.$$

Let \hat{C} be the event that a colliding pair of matching inputs are found for F^{Cyc} with q (pairs) of queries. Then,

$$\Pr[\hat{C}] \leq \sum_{i=2}^q \Pr[\hat{C}_i] \leq \frac{2q}{N - 2q}.$$

Since $\mathbf{Adv}_F^{\text{COLL}}(q) = \Pr[C \vee \hat{C}] \leq \Pr[C] + \Pr[\hat{C}]$, the claim follows. ■

D Preimage Resistance – Proof of Theorem 5

Although, the main focus is on collision resistance, we are also interested in the difficulty of inverting the compression function of F^{Cyc} . Generally speaking, second-preimage resistance is a stronger security requirement than preimage resistance. A preimage may have some information of another preimage which produces the same output. However, in the ideal cipher model, for the compression function F^{Cyc} , a second-preimage has no information useful to find another preimage. Thus, only preimage resistance is analyzed. Note, that there have been various results that discuss attacks on iterated hash functions in terms of pre- and second-preimage, *e.g.* long-message second-preimage attacks [7,19], in such a way that the preimage-resistance level of a compression function cannot easily be transferred to an iterated hash function built on it.

The adversary's goal is to output a preimage (G, H, M) for a given ζ , where ζ is taken randomly from the output domain, such as $F^{\text{Cyc}}(G, H, M) = \zeta$. We will again dispense the adversary from having to output such a preimage. Instead, we will determine whether the adversary has been successful or not by examining its query history \mathcal{Q} . We say, that $\text{PREIMG}(\mathcal{Q})$ holds if there is such a preimage and \mathcal{Q} contains all the queries necessary to compute it.

Definition 15. (*Inverting random points of a compression function*) Let F^{CYC} be as in Definition 3. Fix an adversary \mathcal{A} that has access to oracles E, E^{-1} . The advantage of \mathcal{A} of inverting $F := F^{\text{CYC}}$ is the real number

$$\begin{aligned}\mathbf{Adv}_F^{\text{INV}}(\mathcal{A}) &= \Pr[E \xleftarrow{R} \text{BC}(n, k); \zeta \xleftarrow{R} \Omega^2; \\ &\quad (G, H, M) \xleftarrow{R} \mathcal{A}^{E, E^{-1}}(\zeta) \mid F^{\text{CYC}}(G, H, M) = \zeta].\end{aligned}$$

Again, for $q \geq 1$, we write

$$\mathbf{Adv}_F^{\text{INV}}(q) = \max_{\mathcal{A}} \{\mathbf{Adv}_F^{\text{INV}}(\mathcal{A})\}$$

where the maximum is taken over all adversaries that ask at most q oracle queries.

Note, that there has been a discussion on formalizations of preimage resistance. For details we refer to [2, Section 2, Appendix B].

The preimage resistance of the compression function F is given in the following Theorem.

Theorem 6. Let $F := F^{\text{CYC}}$ be as in Definition 3. For any $N' = N - q$ and $q > 1$

$$\mathbf{Adv}_F^{\text{INV}}(q) \leq 2q/(N')^2.$$

Proof. Fix $\zeta = (\sigma_1, \sigma_2) \in \Omega^2$ where $\sigma_1, \sigma_2 \in \Omega$ and an adversary \mathcal{A} asking q queries to its oracles. We upper bound the probability that \mathcal{A} finds a preimage for a given ζ by examining the oracle queries as they come in and upper bound the probability that the last query can be used to create a preimage, *i.e.* we upper bound $\Pr[\text{PREIMG}(\mathcal{Q})]$. Let \mathcal{Q}_i denote the first i queries made by the adversary. The term 'last query' means the latest query made by the adversary since we examine again the adversary's queries $(X_i, K_i)_{\text{fwd}}$ or $(K_i, Y_i)_{\text{bwd}}$ one at a time as they come in. The last query is always given index i .

Case 1: The last query (X_i, K_i, Y_i) is used in the top row. Either X_i or Y_i was randomly assigned by the oracle from a set of at least the size $N' := N - q$.

The query is successful *in the top row* if $P_M^T(X_i) \oplus Y_i = \sigma_1$ and thus has a chance of success of $\leq 1/N'$. In \mathcal{Q}_i there is at most one query Q_j , $j \leq i$ that can be used in the bottom row. This 'bottom' query is successful if such a query is in the query history \mathcal{Q} and $P_M^B(X_j) \oplus Y_j = \sigma_2$ and therefore has a chance of success of $\leq 1/N'$. So the total chance of success is $\leq q/(N')^2$ as the adversary mounts at most q queries.

Case 2: The last query (X_i, K_i, Y_i) is used in the bottom row. The analysis is essentially the same as in case 1. The total chance of success is $\leq q/(N')^2$, too. ■

As any query can be either used in the top or the bottom row, the claim follows. ■

Another Glance at Double-Length Hashing

Onur Özen and Martijn Stam*

LACAL, EPFL, Switzerland

{onur.ozen,martijn.stam}@epfl.ch

Abstract. We propose a novel framework for blockcipher-based double-length hash functions by extending the recent generalization presented by Stam at FSE '09 for single-call hash functions. We focus on compression functions compressing $3n$ bits to $2n$ bits that use one or two calls to a $2n$ -bit key, n -bit block blockcipher. In case of a single call, we concentrate on security in the iteration. In case of two calls, we restrict ourselves to two parallel calls (initially to distinct and independent blockciphers). We analyse the kind of pre- and postprocessing functions that are sufficient to obtain close to optimal collision resistance, either in the compression function or in the iteration. Our framework can be used to get a clearer understanding of a large class of double-length hash functions of this type.

1 Introduction

Ever since the initial design of cryptographic hash functions, one of the most popular and best-known methods to create a hash function revolves around blockciphers. First a compression function is created using the blockcipher (e.g. using the Davies-Meyer method) and subsequently a full-blown hash function is created using the Merkle-Damgård transform [5, 20]. Many of the designers of the SHA-3 candidates follow this paradigm by instantiating their designs using blockciphers, in particular with (components of) the Advanced Encryption Standard (AES). This approach is particularly useful for resource constrained environments as one only needs to implement one blockcipher to obtain an encryption scheme and a hash function on a given platform.

Of the many methods, the most common approach is to use a single call to a blockcipher operating on n -bit blocks with k -bit keys, which typically results in a compression function from $n + k$ bits to n bits. This category is often referred to as rate-1 single-length. Here the rate is a measure of efficiency, usually defined as the ratio of the number of message blocks being hashed over the number of blockcipher calls (thus higher rates are considered more efficient). More recently this category has been called classical rate-1 single call, to emphasize that only a single call to the blockcipher is made per compression function (this is not evident from being rate-1).

Unfortunately, compression functions whose output length matches the block-length of the blockcipher historically face one major problem: existing blockciphers seldom have sufficiently large block-lengths. For instance, to meet the basic requirements of

* The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

NIST, one would need a blockcipher operating on more than 256 bits. This rules out most existing blockciphers, including AES, which operates on 128 bits blocks only.

To counter this discrepancy in required security levels, so called double-length compression functions (and corresponding hash functions) were introduced: these are compression functions having $2n$ -bit output while being based on a blockcipher with only n -bit blocks. Thus, one can hope to achieve for instance collision resistance up to roughly 2^n blockcipher evaluations. Double block-length compression functions can also be useful for wide-pipe designs [18].

Double block-length hash functions come in various guises, depending on the number of blockcipher calls per compression function and the (bit)length of the key (to the blockcipher). The three most important variants are: (i) one call to a $2n$ -bit key blockcipher; (ii) two calls to a $2n$ -bit key blockcipher; and (iii) two calls to an n -bit key blockcipher. In this work, we will largely ignore (iii) and concentrate on double-key blockciphers. Our aim is to give a general, unifying framework by providing sufficient conditions to gain optimal collision resistance for these hash functions.

Our starting point will be the recent framework by Stam [27] for single call $m + s$ -to- s bit compression functions consisting of three simple steps:

1. Prepare key and plaintext: $(K, X) \leftarrow C^{\text{PRE}}(M, V)$;
2. Make the call: $Y \leftarrow E(K, X)$;
3. Output the digest: $W \leftarrow C^{\text{POST}}(M, V, Y)$.

Here, E is a blockcipher with key size k and block size n and C^{PRE} and C^{POST} can be arbitrary functions given their respective domain and codomain (where $|K| = k$, $|X| = |Y| = n$, $|M| = m$, $|W| = |V| = s$). For suitable parameters ($k = m = s = n$) and \mathbb{F}_2 -‘block’-linear instances of C^{PRE} and C^{POST} this leads to the PGV-schemes [23]. Yet our interest goes to the supercharged scenario, where $s > n$ (but still $n + k = m + s$) and then in particular the rate-1 double-length scenario with $s = 2n$ and $m = n$.

Stam [27] gave a general theory of collision-resistant supercharged compression functions plus a specific construction for the double-length case with almost optimal collision resistance. However, his framework does not include the earlier work by Lucks [19], who gave a rate-1 double-length hash function that is almost optimally collision resistant, but in the iteration only. We extend the original framework so it includes schemes that only become secure in the iteration.

One of the main reasons to look at security in the iteration—and possibly foregoing security of the compression function itself—is that it potentially leads to more efficient schemes. For example, for collision resistance of a classical, single-length compression function it is hard to avoid a feedforward from the input to the blockcipher to the output of the compression function (as in the Davies-Meyer construction). When considering security in the iteration only, this feedforward is no longer required. It has recently been shown [2] that the feedforward can contribute significantly to the cost of implementing the hash function.

Although efficiency of blockcipher-based compression functions is usually measured by the rate (the number of blockcipher calls required per message block), this metric does not always give accurate efficiency estimates. In particular, both Lucks’s and Stam’s rate-1 construction require two \mathbb{F}_{2^n} finite field multiplications. In practical situations, these field multiplications might actually cost more than a blockcipher call. For

this reason, we will also look at schemes using two calls to the underlying blockcipher. For this, we extend the above framework as follows:

1. Prepare key and plaintext: $(K_1, X_1) \leftarrow C_1^{\text{PRE}}(M, V), (K_2, X_2) \leftarrow C_2^{\text{PRE}}(M, V)$;
2. Make the calls: $Y_1 \leftarrow E_1(K_1, X_1), Y_2 \leftarrow E_2(K_2, X_2)$;
3. Output the digest: $W \leftarrow C^{\text{POST}}(M, V, Y_1, Y_2)$.

Here E_1 and E_2 are both blockciphers with key size k and block size n , and $C_1^{\text{PRE}}, C_2^{\text{PRE}}$, and C^{POST} can be arbitrary functions given their respective domain and codomain. Note that this scheme makes blockcipher calls in parallel. (One could generalize even further by allowing even more calls, and sequentially). We investigate particularly the case where $|K_1| = |K_2| = |V| = 2n$ and $|M| = |X_1| = |X_2| = n$. There are plenty of designs known that adhere to this framework (and even more that do not; see the Appendix for a summary of related work).

We further group the schemes according to two important characteristics. Firstly, we will distinguish between collision resistance in the compression function (Type-I) and collision resistance in the (MD) iteration (Type-II), just as was done in the classical case by Black et al. [1]. Secondly, for secure compression functions only, we will differentiate between schemes where the two blockciphers E_1 and E_2 are distinct (and independent in the ideal cipher model) and schemes where only a single blockcipher is used, so $E_1 = E_2$.

Each type is defined by a set of conditions on pre- and postprocessing functions (cf. [27]). For Type-I schemes with distinct blockciphers, the requirements are a rather straightforward generalization of those by Stam for single-call constructions. For Type-I schemes with only a single blockcipher, we follow Hirose's [12] and Nandi's [22] approach for implicit domain separation based on some extra conditions on $C_2^{\text{PRE}}(C_1^{-\text{PRE}}(\cdot))$ (in particular, this being an involution without fixed points).

We show that Type-I schemes enjoy near optimal collision resistance in the ideal cipher model. We believe preimage resistance to be close to optimal as well, but like other works in this area, we were only able to prove preimage resistance up to the birthday bound. The ramifications of our framework for \mathbb{F}_2 -‘block’-linear instances of C^{PRE} and C^{POST} for Type-I schemes are investigated and compared to earlier work.

For Type-II schemes, the generalization from the classical single-call case is less straightforward. (Type-II schemes are investigated only under the assumption that two independent blockciphers are called.) In fact, we can only prove collision resistance up to roughly $2^n/n$ queries, although we do believe that our conditions suffice for optimal collision resistance. We note that for the classical single-call case no loss occurs.

2 Preliminaries

General notation. For a positive integer n , we write $\{0, 1\}^n$ for the set of all bitstrings of length n . When X and Y are strings we write $X \parallel Y$ to mean their concatenation and $X \oplus Y$ to mean their bitwise exclusive-or (xor).

For positive integers k and n , we let $\text{Block}(k, n)$ denote the set of all blockciphers with k -bit key and operating on n -bit blocks. That is $E \in \text{Block}(k, n)$ is a collection of 2^k permutations on the set $\{0, 1\}^n$. Given that $E(K, \cdot)$ is a permutation for all $K \in \{0, 1\}^k$, we write $D(K, \cdot)$ for its inverse.

Unless otherwise specified, all finite sets are equipped with a uniform distribution for random sampling. For example, $E \xleftarrow{\$} \text{Block}(k, n)$ denotes random sampling from the set $\text{Block}(k, n)$ and assignment to E . We use the convention to write oracles that are provided to an algorithm as superscripts, for example \mathcal{A}^E would be an adversary with black box access to some function E .

We write $B[Q; p]$ for random variable counting the number of successes in Q independent Bernoulli trials, each with success probability p . It is binomially distributed, for integer $0 \leq \kappa \leq Q$:

$$\Pr[B[Q; p] = \kappa] = \binom{Q}{\kappa} p^\kappa (1-p)^{Q-\kappa}.$$

A Chernoff bound can be used to bound the tail probability for any $\kappa > Qp$, namely

$$\Pr[B[Q; p] > \kappa] < \left(\frac{epQ}{\kappa}\right)^\kappa.$$

Compression functions. A *compression function* is a mapping H from $\{0, 1\}^m \times \{0, 1\}^s$ to $\{0, 1\}^s$ for some $m, s > 0$. A *blockcipher-based* compression function is a mapping $H : \{0, 1\}^m \times \{0, 1\}^s \rightarrow \{0, 1\}^s$ given by a program that, given (M, V) , computes $H^{E_1, \dots, E_r}(M, V)$ via access to a finite number of specified oracles E_1, \dots, E_r , where $E_1, \dots, E_r : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ are (ideal) blockciphers with k -bit key and operating on n -bit blocks.¹ A *single-layer* blockcipher-based compression function calls all its encryption oracles in parallel. That is, let $C_i^{\text{PRE}} : \{0, 1\}^m \times \{0, 1\}^s \rightarrow \{0, 1\}^k \times \{0, 1\}^n$, for $i = 1, \dots, r$, and $C^{\text{POST}} : \{0, 1\}^m \times \{0, 1\}^s \times (\{0, 1\}^n)^r \rightarrow \{0, 1\}^s$, be pre- and postprocessing functions respectively. Compression of a message block then proceeds as follows (see Fig. 1). Given an s -bit state V and m -bit message M , compute digest $W = H^{E_1, \dots, E_r}(M, V)$ by

$$(K_i, X_i) \leftarrow C_i^{\text{PRE}}(M, V); \\ Y_i \leftarrow E_i(K_i, X_i)$$

for $i = 1, \dots, r$ and finally output

$$W \leftarrow C^{\text{POST}}(M, V, Y_1, \dots, Y_r).$$

Hash functions. A compression function can be made into a hash function by iterating it. We briefly recall the standard Merkle-Damgård iteration [5, 21], where we assume that there is already some injective padding from $\{0, 1\}^* \rightarrow (\{0, 1\}^m)^* \setminus \emptyset$ in place (note that we disallow the empty message $\mathbf{M} = \emptyset$, corresponding to $\ell = 0$, as output of the injective padding). Given an initial vector $V_0 \in \{0, 1\}^s$ define $\mathcal{H}^H : (\{0, 1\}^m)^* \rightarrow \{0, 1\}^s$ as follows for $\mathbf{M} = (M_1, \dots, M_\ell)$:

1. Set $V_i \leftarrow H^{E_1, \dots, E_r}(M_i, V_{i-1})$ for $i = 1, \dots, \ell$;
2. Output $\mathcal{H}^H(\mathbf{M}) = V_\ell$.

¹ It is customary not to use decryptions for the actual computations, though there are a few exceptions to this rule.

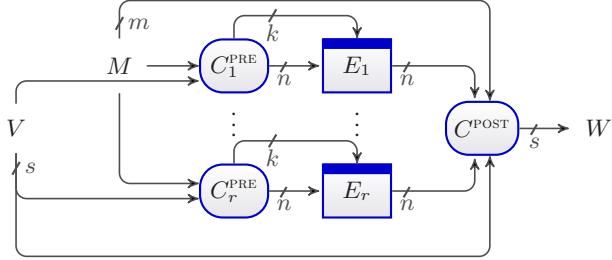


Fig. 1. General form of an $m + s$ -to- s bit single layer blockcipher based compression function based on r calls to underlying blockciphers with k -bit keys and n -bit blocks

(Bearing this iteration in mind, given a compression function $H : \{0, 1\}^m \times \{0, 1\}^s \rightarrow \{0, 1\}^s$ we will refer to the $\{0, 1\}^m$ part of the input as ‘message’ and the $\{0, 1\}^s$ as the ‘state’.)

Security notions. A *collision-finding adversary* is an algorithm with access to one or more oracles, whose goal is to find collisions in some specified compression or hash function. It is standard practice to consider information-theoretic adversaries only: these adversaries are computationally unbounded and their complexity is measured only by the number of queries made to their oracles. Without loss of generality, such adversaries are assumed not to repeat queries to oracles nor to query an oracle outside of its specified domain.

Definition 1. Let $n, k, m, s > 0$ be integer parameters, and fix an integer $r > 0$. Let $H : \{0, 1\}^m \times \{0, 1\}^s \rightarrow \{0, 1\}^s$ be a compression function taking r oracles $E_1, \dots, E_r \in \text{Block}(k, n)$. The collision-finding advantage of adversary \mathcal{A} is defined to be

$$\text{Adv}_H^{\text{coll}}(\mathcal{A}) = \Pr \left[E_1..E_r \stackrel{\$}{\leftarrow} \text{Block}(k, n), ((M, V), (M', V')) \leftarrow \mathcal{A}^{E_1..E_r, D_1..D_r} : (M, V) \neq (M', V') \text{ and } H^{E_1..E_r}(M, V) = H^{E_1..E_r}(M', V') \right].$$

Define $\text{Adv}_H^{\text{coll}}(q)$ as the maximum advantage over all adversaries making at most q queries to each of their oracles.

A *preimage-finding adversary* is an algorithm with access to one or more oracles, whose goal is to find preimages in some specified compression. There exist several definitions depending on the distribution of the element of which a preimage needs to be found. The strongest notion is that preimage resistance should hold with respect to any distribution, which can be formalized as everywhere preimage resistance [24].

Definition 2. Let $n, k, m, s > 0$ be integer parameters, and fix an integer $r > 0$. Let $H : \{0, 1\}^m \times \{0, 1\}^s \rightarrow \{0, 1\}^s$ be a compression function taking r oracles $E_1, \dots, E_r \in \text{Block}(k, n)$. The preimage-finding advantage of adversary \mathcal{A} is defined to be

$$\text{Adv}_H^{\text{epre}}(\mathcal{A}) = \max_{W \in \{0,1\}^s} \Pr \left[E_1..E_r \xleftarrow{\$} \text{Block}(k, n), (M', V') \leftarrow \mathcal{A}^{E_1..E_r, D_1..D_r}(W) : W = H^{E_1..E_r}(M', V') \right] .$$

Define $\text{Adv}_H^{\text{epre}}(q)$ as the maximum advantage over all adversaries making at most q queries to each of their oracles.

The quantities $\text{Adv}_{\mathcal{H}}^{\text{coll}}(q)$ and $\text{Adv}_{\mathcal{H}}^{\text{epre}}(q)$, denoting collision resp. (everywhere) preimage resistance for the iterated hash function \mathcal{H}^H are defined similarly (in this case the advantage of \mathcal{A} is the maximum success probability taken over the choice of possible initial values V_0 , which is input to \mathcal{A}).

Although the standard Merkle-Damgård transform does not preserve collision resistance on its own, it does preserve the combination of collision resistance and everywhere preimage resistance. Formally,

Theorem 1. *Let H be a blockcipher based compression function and let \mathcal{H} be the iterated hash function based on H . Then*

$$\text{Adv}_{\mathcal{H}}^{\text{coll}}(q) \leq \text{Adv}_H^{\text{coll}}(q) + \text{Adv}_H^{\text{epre}}(q), \quad \text{Adv}_{\mathcal{H}}^{\text{epre}}(q) \leq \text{Adv}_H^{\text{epre}}(q) .$$

3 Double-Length Single-Call Compression Functions

In this section, we deal with the possibility of building a rate-1 double-length blockcipher-based hash function. That is, given a blockcipher with $2n$ -bit keys and n -bit blocks, we create a $3n$ -to- $2n$ bit compression function that provides (almost) optimal collision resistance when iterated. Promising results were previously given by Lucks [19], who gave a scheme secure in the iteration, and Stam [27], who gave a general framework for security of the compression function.

Our goal is to extend this framework to schemes secure only in the iteration. Thus, it includes Lucks's construction and gives a better understanding of that scheme, also in the hope to be able to improve upon it. After all, for Lucks's scheme, which is only secure in the iteration, the overhead is dominated by two finite field multiplications. This is a similar overhead to Stam's scheme, which in contrast is already secure as compression function. Curiously, it seems hard to improve the efficiency of Lucks's scheme. However, with our analysis it does become easier to extend the scheme to blockciphers with longer key lengths (taking in more message bits per call at only a slight loss of collision resistance). We also uncover a minor flaw in Lucks's original proof.

Stam's collision-resistant compression functions. We briefly recap the result from Stam [27] for collision-resistant compression functions. In fact, the results are not just for double-length compression functions. Namely, given a single call to a blockcipher with n -bit blocks and k -bit keys, a compression function is considered taking in m -bit message and s -bit state (and outputting s -bit again). The following definition and theorem are given:

Definition 3. [27, Definition 19] A single-call blockcipher-based compression function H^E is called supercharged single-call Type-I with overlap γ iff $s \geq n$, $m + s = n + k$ and the following three hold:

1. The preprocessing C^{PRE} is bijective.
2. For all M, V the postprocessing $C^{\text{POST}}(M, V, \cdot)$ is injective, with effective range $R_{\text{POST},(M,V)}$.
3. For all K, Y the modified postprocessing $C^{\text{AUX}}(K, \cdot, Y)$ is injective, with effective range $R_{\text{AUX},(K,Y)}$.

Where the overlap γ is defined as:

$$\gamma = \max \left\{ |R_Z \cap R_{Z'}| : Z, Z' \in \{\text{POST, AUX}\} \times \{0, 1\}^{k+n}, Z \neq Z' \right\} .$$

Theorem 2. [27, Theorem 20] Let H^E be a supercharged single-call Type-I compression function with overlap γ . Then the advantage of an adversary in finding a collision after $q \leq 2^{n-1}$ queries can be upper bounded by

$$\text{Adv}_H^{\text{coll}}(q) \leq q\kappa/2^{n-1} + 2^{m+s+1} \left(\frac{e\gamma q}{(\kappa - 1)2^{n-1}} \right)^{\kappa-1}$$

for arbitrary positive integer $\kappa > q\gamma/2^{n-1}$.

For double-length compression functions (so $s = k = 2n$ and $m = n$) Stam gives a construction with $\gamma = 3$ based on polynomial evaluation. Concretely, this leads to a compression function with $\text{Adv}_H^{\text{coll}}(q) \leq (n + 1/2)q/2^{n-3}$ and $\text{Adv}_H^{\text{epre}}(q) \leq q/2^{n-1}$.

Security in the iteration. For classical single-length compression functions Black et al. [1] discovered a class of compression functions that lead to collision-resistant hash functions when iterated (using the plain Merkle-Damgård transform), even though the compression function itself might not be collision resistant.

For supercharged compression functions, the security in the iteration has not been studied in its full generality. Lucks [19] gave a fairly wide class, but we provide a comprehensive generalization of his class based on the framework referred to in the previous section. Moreover, as we will see there is a technical flaw in Lucks's proof. Our proof for security in the iteration can therefore be seen as either a refinement of Stam's (and Duo and Li's) approach based on ideas from Lucks, or as a correction of Lucks's approach based on ideas from Duo and Li, and Stam. For instance, in the definition below, the first three requirements are inspired by Stam's supercharged compression function and the classical Type-II schemes, whereas the fourth requirement is strongly inspired by Lucks's work.

Definition 4. A single-call blockcipher-based compression function H^E is called supercharged single-call Type-II iff $s \geq n$, $m + s = n + k$ and the following four hold:

1. The preprocessing C^{PRE} is bijective.
2. For all M, V the postprocessing $C^{\text{POST}}(M, V, \cdot)$ is injective, with effective range $R_{\text{POST},(M,V)}$.

3. For all K the inverse preprocessing $C^{-\text{PRE}}(K, \cdot)$ restricted to V is injective, with effective range $R_{-\text{PRE},(K)}$.
4. For all K, X, X' the functions $C^{\text{POST}}(C^{-\text{PRE}}(K, X), \cdot)$ and $C^{\text{POST}}(C^{-\text{PRE}}(K, X'), \cdot)$ are equal.

Given bijectivity of C^{PRE} , there is a unique correspondence between pairs (M, V) and pairs (K, X) defining a forward query. Hence we can regard $R_{\text{POST},(M,V)}$ as the set of possible outcomes W for a particular forward query, which we could make explicit by writing $R_{\text{POST},(K,X)}$ instead. However, as we will assume that C^{POST} is independent of X we will simply write $R_{\text{POST},(K)}$. Similarly, for an inverse query (K, Y) the set $R_{-\text{PRE},(K)}$ gives the possible chaining variables V .

Let $\mathcal{Z} = \{-\text{PRE}, \text{POST}\} \times \{0, 1\}^k$. For each $Z \in \{\text{POST}\} \times \{0, 1\}^k$ define the overlap function $\gamma_Z : \{0, \dots, 2^n\} \rightarrow \mathbb{N}$ as

$$\gamma_Z(i) = |\{Z' \in \{-\text{PRE}, \text{POST}\} \times \{0, 1\}^k : Z \neq Z' : |R_Z \cap R_{Z'}| = i\}| ,$$

and for each $Z \in \{-\text{PRE}\} \times \{0, 1\}^k$ as

$$\gamma_Z(i) = |\{Z' \in \{\text{POST}\} \times \{0, 1\}^k : |R_Z \cap R_{Z'}| = i\}| .$$

Furthermore, let overlap γ be defined as the smallest integer such that for all $Z \in \{-\text{PRE}, \text{POST}\} \times \{0, 1\}^k$ and all $i > \gamma$ it holds that $\gamma_Z(i) = 0$.

Theorem 3. *Let H^E be a supercharged single-call Type-II compression function with overlap γ as defined above. Then the advantage of an adversary in finding a collision after $q \leq 2^{n-1}$ queries can be upper bounded by*

$$\text{Adv}_{\mathcal{H}}^{\text{coll}}(q) \leq q\kappa/2^{n-1} + 2^{m+s+1} \left(\frac{e\gamma q}{(\kappa - 1)2^{n-1}} \right)^{\kappa-1}$$

for arbitrary positive integer $\kappa > q\gamma/2^{n-1}$.

Proof. We first recall the general framework to prove collision resistance in the iteration [7, 27], adapted to the current setting. Let $V_0 \in \{0, 1\}^s$ be \mathcal{H} 's initial vector. We define a directed graph $G = (V_G, E_G)$ with vertex set $V_G = \{0, 1\}^s$ —corresponding to all 2^s possible chaining values—and initially an empty edge set $E_G = \emptyset$. Edges are dynamically added based on the queries to E and D . In particular, we add an arc (V, W) , labeled by M , if we know a message M such that $W = H^E(M, V)$ and the relevant query to either E or D has been made. Finding a collision would require constructing a ρ -shape containing the initial vector V_0 .

The graph we use is similar to the one used by Duo and Li [7]; the proof of [27, Theorem 9] uses an undirected graph. The proof of [19, Theorem 1] uses a directed graph but the direction relates to E or D being used to generate arc, so typically the adversary has full control over the start edge but not over the endpoint. With this modified direction, Lucks argues that to find a collision in the iterated hash, either a natural collision (two arcs pointing to the same node) or a natural preimage (an arc pointing to the initial vector V_0) have to occur. Indeed this also follows from the ρ -shape: any

way of assigning directions to the edges in the ρ will invariably lead to one of Lucks's events.

In fact, we can refine this analysis even further. In order for a collision to occur, one of the following three events needs to occur (on the i 'th query, for some i):

1. A forward query leads to a digest that previously resulted as digest of a forward query, or equals the initial value.
2. A forward query leads to a digest that previously resulted as chaining variable of an inverse query.
3. An inverse query leads to a chaining variable that previously resulted as digest of a forward query, or equals the initial value.

In particular, unlike previous work, we can ignore the event that two inverse queries both lead to the same chaining variable V .

We will now bound the probability of the i 'th forward query (K, X) leading to one of these bad events. For $Z = (\text{POST}, K)$ let $T_Z(i - 1)$ be the set containing the initial vector, all chaining variables that resulted so far from inverse queries, and all digests that resulted so far from forward queries using keys different from K . Then for a bad event to occur, the resulting digest should be in $T_Z(i - 1)$. Note that we can safely ignore forward queries with the same key. Queries (K, X) and (K, X') (with $X \neq X'$) are guaranteed to lead to different answers Y and Y' since $E(K, \cdot)$ is a permutation. Moreover, both Y and Y' are fed into the same injection (due to properties 2 and 4 of the compression function) and can therefore never produce colliding digests.

We also know that the digest will certainly be in the set R_Z corresponding to the query (K, X) made to E . Moreover, the digest will be uniformly distributed over R_Z , up to the usual correction taking care of previous queries to E or D using the same key. For a bad event to occur on the i 'th forward query, the digest therefore needs to be in $R_Z \cap T_Z(i - 1)$, which happens with probability at most $|R_Z \cap T_Z(i - 1)|/(2^n - i)$.

Next consider the third possible bad event occurring on the i 'th inverse query (K, Y) . For $Z = (-\text{PRE}, K)$ let $T_Z(i - 1)$ be the set containing the initial vector and all digests that resulted so far from forward queries. This time the chaining variable will be uniformly distributed over R_Z (up to the usual correction) and, for the bad event to occur, the chaining variable should be in $T_Z(i - 1)$. Again this happens with probability at most $|R_Z \cap T_Z(i - 1)|/(2^n - i)$.

For any given positive integer κ , let p_κ be the probability that $\max_Z |R_Z \cap T_Z(q)| > \kappa$. Then the total probability of finding a collision can be upper bounded by $p_\kappa + \sum_{i=1}^q \kappa/(2^n - i)$, where as usual we can upper bound the second term by $q\kappa/2^{n-1}$ (or get a vacuous bound).

What remains is bounding p_κ . Here we need to use a slightly more sophisticated approach than originally used [19, 27]. Fix Z , then the probability that some other query $Z' \neq Z$ gives an answer that will lie in $R_Z \cap T_Z$ is upper bounded by $\gamma/(2^n - i) < \gamma/2^{n-1}$, where T_Z is essentially used to screen out queries Z' that are irrelevant (such as inverse queries when Z itself is an inverse query). Since the answer to Z' will certainly lie in $R_{Z'}$, we can upper bound the probability that the answer lies in $R_Z \cap R_{Z'}$. Note that $|R_Z \cap R_{Z'}| \leq \gamma$ by construction (and definition). Thus the probability that any query adds to $|R_Z \cap T_Z(q)|$ is at most $\gamma/2^{n-1}$ (assuming $q < 2^{n-1}$). Assume that $V_0 \in R_Z$ we can then upper bound the probability that $|R_Z \cap T_Z(q)|$ exceeds κ by a

binomial distribution with q repetitions and probability $\gamma/2^{n-1}$. For $\kappa > q\gamma/2^{n-1}$ this is allowed. A Chernoff bound followed by a union bound then allows us to bound by:

$$\begin{aligned} \Pr \left[\max_Z |R_Z \cap T_Z(q)| \geq \kappa \right] &\leq \sum_Z \Pr [|R_Z \cap T_Z(q)| \geq \kappa] \\ &\leq 2^{m+s+1} \Pr [B[q; \gamma/2^{n-1}] \geq \kappa - 1] \\ &\leq 2^{m+s+1} \left(\frac{e\gamma q}{(\kappa - 1)2^n} \right)^{\kappa - 1}. \end{aligned}$$

Collecting the two terms gives the stated upper bound. \square

Comparison with Lucks's construction. It is worthwhile to compare our work with that of Lucks, who develops a theory and proposes some constructions for what he calls rate-1 double-length hash functions (collision resistant in the iteration). Below we reproduce his definition and theorem adapted to our notation.

Definition 5. [19] Let $f : \{0, 1\}^n \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$ be a function satisfying

Invertibility. For all M , the function $f(M, \cdot)$ is invertible.

Uniqueness. For all V, K there is at most one M such that $f(M, V) = K$.

Collision universality. For all distinct pairs (V, V') , the number of pairs (M, M') for which $f(M, V) = f(M', V')$ is at most γ .

Define a Lucks-style compression function with overlap γ by setting $C^{\text{PRE}}(M, V) = (f(M, V), M)$ and $C^{\text{POST}}(M, V, Y) = f(Y, f(M, V))$.

Theorem 4. [19, Theorem 1] Let a Lucks-style compression function with overlap γ be given. Then for any positive integer κ , the collision advantage of an adversary against the iterated hash function can be bounded by

$$\text{Adv}_{\mathcal{H}}^{\text{coll}}(q) \leq q\kappa/(2^n - q) + 2^{2n} \left(\frac{\gamma q}{2^{n-q}} \right)^\kappa / \kappa!$$

Our bound is strikingly similar to Lucks's bound, which raises the question how the two results compare. It turns out that Lucks's construction satisfies the requirements of a supercharged single-call Type-II compression function with $m = n$ and $k = s = 2n$, provided we add one more constraint, namely

Collision universality II. For all distinct pairs (V, V') , the number of pairs (M, M') for which $f^{-1}(M, V) = f(M', V')$ is at most γ .

In fact, without this extra condition the proof given by Lucks appears incomplete (see elaboration below). It is interesting to note that, given our general framework and the morphology of Lucks's compression function, it is possible to derive the conditions Lucks poses on his function f (although in fairness we did steer our framework partly based on Lucks's work).

Lemma 1. Any amended Lucks-style compression function with overlap γ (based on f) is a supercharged single-call Type-II compression function (with overlap γ).

Proof. We will first tick off the three requirements given in Definition 4, before bounding the overlap.

1. Firstly, by definition C^{PRE} is bijective iff $C^{\text{PRE}}(M, V) = (f(M, V), M)$ is bijective. Since the domain and range are finite (and identical) bijectivity is equivalent to injectivity. It is clear that, regardless of f , different messages $M \neq M'$ will never cause a collision under C^{PRE} . For any given M , C^{PRE} will be injective iff $f(M, \cdot)$ is injective. Hence, C^{PRE} is bijective iff $f(M, \cdot)$ is bijective for all M , corresponding to Lucks's *invertibility*.
2. Secondly, $C^{\text{POST}}(M, V, \cdot) = f(\cdot, f(M, V))$ is injective iff $f(\cdot, S)$ injective for $S = f(M, V)$. Since we need $C^{\text{POST}}(M, V, \cdot)$ to be injective for all (M, V) and f is invertible, this is equivalent to $f(\cdot, S)$ injective for all S . By definition (of injectivity) this corresponds to Lucks's *uniqueness*.
3. Thirdly, $C^{-\text{PRE}}(K, \cdot)$ when restricted to V should be injective. Since $C^{-\text{PRE}}(K, X) = (X, f^{-1}(X, K))$, we have that $C^{-\text{PRE}}(K, \cdot)$ restricted to V is $f^{-1}(\cdot, K)$. In order $f^{-1}(\cdot, K)$ to be injective for all K is equivalent to Lucks's *uniqueness*.
4. Finally, for all K, X, X' the function $C^{\text{POST}}(C^{-\text{PRE}}(K, X), \cdot) = f(\cdot, K)$ should be independent of X , which is clearly the case.

What remains to show is that the overlap γ as defined for supercharged single-call Type-II compression functions matches the one as defined for the amended Lucks scheme. In other words, we want to put bounds on γ_Z for all $Z \in \mathcal{Z}$ using the collision universality for f . Let $Z, Z' \in \mathcal{Z}$ with $Z \neq Z'$. We will consider two cases, depending on whether Z and Z' both correspond to a forward query or to distinct types (w.l.o.g., Z a forward query and Z' an inverse query).

Suppose $Z = (\text{POST}, K)$ and $Z' = (\text{POST}, K')$ for $K, K' \in \{0, 1\}^k$ with $K \neq K'$ and consider $R_Z \cap R_{Z'}$. Then $W \in R_Z \cap R_{Z'}$ iff there exist Y and Y' such that $f(Z, Y) = W = f(Z', Y')$. Since by assumption (collision universality) there exist at most γ pairs (Y, Y') such that $f(Z, Y) = f(Z', Y')$ it follows that $|R_Z \cap R_{Z'}| \leq \gamma$.

Finally, suppose that $Z = (\text{POST}, K)$ and $Z' = (-\text{PRE}, K')$. Then $V \in R_Z \cap R_{Z'}$ iff there exist X and Y such that $f(V, X) = K$ and $f(K, Y) = V$. By our amendment (collision universality II) there exist at most γ pairs (X, Y) for which $f(K, Y) = f^{-1}(K, X)$; it follows that $|R_Z \cap R_{Z'}| \leq \gamma$. \square

Note that the last case is missed by Lucks in his proof. Essentially, in his bounding of ‘natural collisions’ he only deals with the forward-forward scenario (so he also misses the inverse-inverse case, which as we showed can be ignored). As Lucks already points out, his compression function cannot be collision resistant (assuming an adversary can invert f , which it always can in the information theoretic setting). Indeed, if we look at the third requirement from Definition 3, we see that for collision resistance of the compression function we require $C^{\text{AUX}}(K, \cdot, Y)$ to be injective. Although this is strictly speaking only (part of) a sufficient condition—not a necessary one—here $C^{\text{AUX}}(K, \cdot, Y) = f(Y, K)$ is a constant function and thus as far removed from injective as possible (showing that Lucks's construction can never be a supercharged single-call Type-Icompression function).

Lucks's instantiation. Lucks also presents an explicit construction based on finite field arithmetic, identifying $\{0, 1\}^n$ with \mathbb{F}_{2^n} , by defining $f : \mathbb{F}_{2^n}^3 \rightarrow \mathbb{F}_{2^n}^2$ as

$$f(M, V_1, V_2) = \begin{cases} (V_1 + M, V_2 \cdot M) & \text{if } M \neq 0, \\ (V_1, V_2) & \text{if } M = 0. \end{cases}$$

Because of our amendment, we need to recheck the computations of the overlap γ . Without the amendment and disallowing $V_2 = 0$ leads to $\gamma = 3$, additionally disallowing $M = 0$ even reduces this to $\gamma = 1$ (note that if the initial vector has $V_2 \neq 0$, then $V_2 = 0$ cannot occur later on in the iteration either). With the amendment in the second case we get $\gamma = 2$ instead.

4 Collision-Resistant Double-Call Compression Functions

Although reasonably collision-resistant single-call compression and hash functions exist, the overhead is quite large. As a consequence, double-call constructions might well be more efficient in practice (either existing ones or new ones). In this section, we will consider collision-resistant compression functions and see how existing double-length proposals fit in. We will first discuss the scenario where the two blockciphers are independently sampled, before discussing the case of one blockcipher called twice. In Section 5, we discuss security in the iteration, an hitherto unstudied problem for the types of constructions we consider.

Common setup. Let us consider the specific case of a double-length compression function that compresses (only) n bits per two parallel calls to $2n$ -bit key n -bit block blockciphers E_1 and E_2 (so $r = 2$, $k = s = 2n$ and $m = n$). The input to the compression function consists of $2n$ bits chaining input V and n bit message input M . The preprocessing function C^{PRE} takes these $3n$ bits as input and outputs $6n$ bits, being $2n$ -bit key K_1 and n -bit plaintext X_1 to E_1 as well as $2n$ -bit key K_2 and n -bit plaintext X_2 to E_2 . For convenience, we will separate C^{PRE} into two, being C_1^{PRE} and C_2^{PRE} , as illustrated in Fig. 2. Following the evaluation of E_1 and E_2 , the postprocessing C^{POST} takes (M, V, Y_1, Y_2) (the latter two are the outputs from E_1 and E_2 respectively) and outputs $2n$ bits output W .

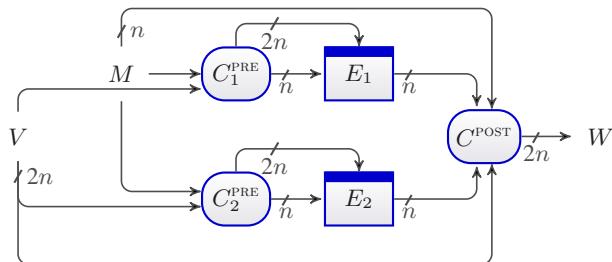


Fig. 2. The overall view of a double-length Type-I compression function

4.1 Distinct and Independent Blockciphers

Definition 6. A double-length blockcipher-based compression function is called double-length Type-I iff the following three hold:

1. C_1^{PRE} (the map that takes M, V to K_1 and X_1) and C_2^{PRE} (the map that takes M, V to K_2 and X_2) are both bijections;
2. $C^{\text{POST}}(M, V, \cdot, \cdot)$ is a bijection (from $2n$ to $2n$ bits, for all (M, V));
3. $C_1^{\text{AUX}} = C^{\text{POST}}(C_1^{\text{PRE}}(K_1, \cdot), Y_1, \cdot)$ is a bijection (for all K_1, Y_1) and $C_2^{\text{AUX}} = C^{\text{POST}}(C_2^{\text{PRE}}(K_2, \cdot), \cdot, Y_2)$ as well (for all K_2, Y_2).

Collision resistance. The following result gives close to optimal collision resistance for double-length Type-I compression functions. Note that we count the total number of queries q the adversary makes to either of its four oracles (E_1, E_2, D_1, D_2). Since a compression function evaluation itself takes two queries, this means we lose a factor of at most four in the bound below compared to the lower bound based on a generic (birthday) collision attack against the compression function.

Theorem 5. Let H^{E_1, E_2} be a double-length Type-I compression function (based on blockciphers with block-length n). Then the advantage of an adversary in finding a collision in H^{E_1, E_2} after q queries can be upper bounded by

$$\text{Adv}_H^{\text{coll}}(q) \leq \frac{1}{2}q(q-1)/(2^n - q)^2.$$

Proof. A collision consists of two pairs (M, V) and (M', V') satisfying $H^{E_1, E_2}(M, V) = H^{E_1, E_2}(M', V')$ yet $(M, V) \neq (M', V')$. We construct a list of triples (M, V, W) such that $W = H^{E_1, E_2}(M, V)$ and the adversary has made the relevant queries to E_1, E_2 and/or D_1, D_2 . We will bound the probability of a collision occurring in this list.

Bijectivity of C_1^{PRE} ensures that any query to E_1 (or D_1) will add at most one triple (M, V, W) to the list of computable inputs (with their output). Moreover, any query to E_1 corresponds uniquely to a forward query (X_2, K_2) to E_2 , by the bijectivity of C_2^{PRE} . (In case of a query to D_1 , the relevant value of (X_2, K_2) is only known after the D_1 -query has been answered.) The case for queries made to E_2 is similar due to symmetry. We will assume that an adversary asks a query to E_1 and the corresponding query to E_2 in conjunction—or rather, we consider a derived adversary obeying this rule—and henceforth refer to these tuples as query pairs. Note that after q queries by the original adversary, there are at most q query pairs for the derived adversary and that the advantage of the derived adversary is at least that of the original one.

As a result, after $i - 1$ query pairs the list of computable values contains exactly $i - 1$ triples (M, V, W) . We claim that the probability of the i 'th query pair causing a collision with any of these triples is at most $(i-1)/(2^n - i + 1)^2$. Using a union bound, the probability of a collision after q queries can then be upper bounded by $\sum_{i=1}^q (i-1)/(2^n - i + 1)^2 \leq \frac{1}{2}q(q-1)/(2^n - q)^2$. What remains is our claim for success on the i 'th query pair. Let us distinguish depending on the first query of the pair.

Consider a forward query (K_1, X_1) to E_1 . By the bijectivity of C_1^{PRE} and C_2^{PRE} , the corresponding values (M, V) and (K_2, X_2) are uniquely determined. Suppose that so

far $t_1 \leq i - 1$ queries to E_1 have been made involving key K_1 and $t_2 \leq i - 1$ to E_2 involving key K_2 . The answer to a fresh query to $E_1(K_1, \cdot)$ will therefore be distributed uniformly over a set of $2^n - t_1$ possible outcomes and, similarly, the answer to a fresh query to $E_2(K_2, \cdot)$ will be distributed uniformly over a set of at least $2^n - t_2$ possible outcomes. Each possible answer (Y_1^*, Y_2^*) will be combined under C^{POST} with the pair (M, V) consistent with the (K_1, X_1) and (K_2, X_2) queries being made, leading to a possible compression function outcome W^* . Because C^{POST} is bijective when the pair (M, V) is fixed, distinct (Y_1^*, Y_2^*) lead to distinct W^* , so there are at least $(2^n - (i-1))^2$ possible outcomes W^* , all equally likely. The probability of hitting a set of size $(i-1)$ is therefore at most $(i-1)/(2^n - i + 1)^2$, as claimed. As the situation is symmetric for a forward query made to E_2 , the number of possible outcomes is the same.

Similarly, consider an inverse query (K_1, Y_1) . This yields a unique X_1 and hence by the bijectivity of C_1^{PRE} , there is a unique pair (M, V) corresponding to this query once answered. Thus, each inverse query will add one triple (M, V, W) to the adversary's list of computable values. Again, the unique pair (M, V) fully determines (K_2, X_2) and obviously Y_2 by the bijectivity of C_2^{PRE} and E_2 . Now, suppose that so far t_1 queries to E_1 (or D_1) have been made involving key K_1 , resulting in t_1 plaintext-ciphertext pairs. The answer to a fresh query to $D_1(K_1, \cdot)$ will therefore be different from the previous plaintexts. Moreover, each of the $2^n - t_1$ answers is equally likely if E_1 is an ideal cipher. Each possible answer X_1^* will be combined under $C_1^{-\text{PRE}}$ and produce a unique (M, V) pair which also determines the forward query to E_2 . Assuming E_2 is an ideal cipher, the output Y_2^* of this query will be distributed uniformly over a set of at least $2^n - (i-1)$ possible outcomes. This time, the bijectivity of C_1^{AUX} implies that the uncertainty in X_1^* and Y_2^* is fully inherited by W , that is W is uniform over a set of size at least $(2^n - (i-1))^2$. Again the probability of hitting any of the $i-1$ previously computed digests is as claimed. The result for the inverse query made to D_2 follows similarly using that C_2^{AUX} is bijective. \square

Preimage resistance. The following result gives an upper bound for $\text{Adv}_H^{\text{epr}}(q)$, which implies a bound by Hirose [11]. However, this expression is far from optimal and vacuous for $q \geq 2^n$. Nevertheless, it states that the success probability of finding a preimage is negligible if $q \leq 2^{n-1}$. As far as we are aware, proving preimage resistance beyond the birthday bound is still an open problem for double-length constructions.

Theorem 6. *Let H^{E_1, E_2} be a double-length Type-I compression function (based block-ciphers with block-length n). Then the advantage of an adversary in finding a preimage in H^{E_1, E_2} after $q \leq 2^{n-1}$ queries can be upper bounded by*

$$\text{Adv}_H^{\text{epr}}(q) \leq q/(2^n - q)^2.$$

Proof. Let \mathcal{A} be an adversary trying to find a preimage for its input σ , asking a total of q queries to (E_1, D_1, E_2, D_2) . As done in the proof of Theorem 5, we will only consider a derived adversary that makes its queries to E_1 and E_2 in pairs, allowing it to construct triples (M, V, W) such that $W = H^{E_1, E_2}(M, V)$. A preimage of σ is found iff σ occurs as W in the list of triples constructed this way, so it suffices to bound that probability instead.

As before, the i 'th query pair will add one triple (M, V, W) to the list of computable values where W is uniform over a set of size at least $(2^n - i + 1)^2$ (by the bijectivity of C^{POST} for forward-forward queries and C_j^{AUX} for backward-forward queries for $j = 1, 2$). Thus, the probability that the i 'th query pair finds a preimage is at most $1/(2^n - i + 1)^2$. Using a union bound, the probability of finding a preimage after q queries can then be upper bounded by $\sum_{i=1}^q 1/(2^n - i + 1)^2 \leq q/(2^n - q)^2$. \square

4.2 Using a Single Blockcipher: Implicit Domain Separation

Previously, we assumed two independently sampled blockciphers. In practice, it is more realistic if only one blockcipher is used, twice in this case per compression function. From a theoretical point of view, this can easily be enforced by using explicit domain separation at only a small cost. In particular, if E is a blockcipher with key length $K+1$, we can define $E_1(K, X) = E(0||K, X)$ and $E_2(K, X) = E(1||K, X)$. It is well known (and easy to verify) that if E is a randomly selected blockcipher, then (and only then) E_1 and E_2 are essentially two independently sampled blockciphers. Although the cost of this explicit separation is only one key bit, it is not an entirely satisfactory situation. For one, many existing constructions do not use this explicit separation and would consequently fall outside any framework that relies upon it.

Recently Nandi [22] and Hirose [12] showed how implicit domain separation can be achieved based on an involution without fixed points. In particular, define $p = C_2^{\text{PRE}}(C_1^{-\text{PRE}}(\cdot))$, then p is an involution without fixed points iff $p(x) \neq x$ yet $p^2(x) = x$ for all $x \in \{0, 1\}^{k+2n}$. (Note that Nandi and Hirose always assume C_1^{PRE} to be the identity function.) We incorporate this approach into our general framework.

In the definition below, we first repeat the first three requirements from Definition 6 (where the domain separation was still explicit). New requirement 4 captures the notion of $p = C_2^{\text{PRE}}(C_1^{-\text{PRE}}(\cdot))$ being an involution without fixed points. The advantage of using involutions is that, as in the case for explicit domain separation, relevant queries to E still come in pairs. That is, for $(K, X) \in \{0, 1\}^{k+2n}$ we call $(K', X') = p(K, X)$ its *conjugate* (and note that $(K, X) = p(K', X')$). Queries $E(K, X)$ and $E(K', X')$ are similarly called conjugate queries. An inverse query $D(K, Y)$, once answered, has a uniquely defined conjugate (forward) query as well. Finally $(M, V) \in \{0, 1\}^{m+2n}$ has conjugate $(M', V') = C_2^{-\text{PRE}}(C_1^{\text{PRE}}(M, V))$.

Fixed points in p are prohibited. They would correspond to the situation where $C_1^{\text{PRE}}(M, V) = C_2^{\text{PRE}}(M, V)$, so a single query would suffice to evaluate the compression function (for that particular input (M, V)). If p would consist of only fixed points (i.e. equal the identity function), then the construction is essentially a supercharged single call compression function in disguise. Since it is known that even in that case almost optimal collision resistance can be achieved, in principle a smaller number of fixed points in p could theoretically also be dealt with.

Similarly, the new requirement 5, stating that two conjugate message-state pairs never collide with each other, could be relaxed to a situation where conjugate message-state pairs only cause a collision (with each other) with a certain probability. The theorem statement below can be amended by adding the sum of the q largest such ‘conjugate-internal’ collision probabilities to the upper bound on the advantage. A natural example of such a situation might be the occurrence of a conjugate-internal collision

iff the conjugate queries give identical answers (in the absence of fixed points this is impossible if conjugate queries have identical keys).

Definition 7. A double-length blockcipher-based compression function is called IDS double-length Type-I iff the following five hold:

1. C_1^{PRE} (the map that takes M, V to K_1 and X_1) and C_2^{PRE} (the map that takes M, V to K_2 and X_2) are both bijections;
2. $C^{\text{POST}}(M, V, \cdot, \cdot)$ is a bijection (from $2n$ to $2n$ bits, for all (M, V));
3. $C_1^{\text{AUX}} = C^{\text{POST}}(C_1^{\text{PRE}}(K_1, \cdot), Y_1, \cdot)$ is a bijection (for all K_1, Y_1) and $C_2^{\text{AUX}} = C^{\text{POST}}(C_2^{\text{PRE}}(K_2, \cdot), \cdot, Y_2)$ as well (for all K_2, Y_2).
4. For all M, V, M', V' , If $C_1^{\text{PRE}}(M, V) = C_2^{\text{PRE}}(M', V')$ then $(M, V) \neq (M', V')$ and $C_1^{\text{PRE}}(M', V') = C_2^{\text{PRE}}(M, V)$.
5. For all $(M, V) \neq (M', V')$ with $C_1^{\text{PRE}}(M, V) = C_2^{\text{PRE}}(M', V')$ and all Y_1, Y_2 , it holds that $C^{\text{POST}}(M, V, Y_1, Y_2) \neq C^{\text{POST}}(M', V', Y_2, Y_1)$.

Collision resistance. As in the proof of Theorem 5 (and of course the original proofs [12, 22]) we will only consider a derived adversary that always asks conjugate queries in conjunction. However, in contrast to the scenario with two distinct blockciphers, in this case such a query pair will yield two compression function evaluations (for some (M, V) and its conjugate). Consequently, the bound here is less tight (roughly by a factor of two) compared to that of Theorem 5.

Finally, looking at involutions only for p is not fully general. In particular, it does not include the well-known Abreast-DM scheme. In that case, one can generalize by regarding longer cycles in p as well. This has recently been worked out (independently) by Lee and Kwon [17] and Fleischmann et al. [9]. Roughly speaking, they prove that if $p^c = 1$ for some (smallest) $c > 2$ and given certain natural restrictions on the postprocessing, then the collision finding advantage can be upper bounded by $\frac{1}{2}(cq/(2^n - cq))^2$. (For Abreast-DM it holds that $c = 6$.)

Theorem 7. Let H^E be an IDS double-length Type-I compression function (based on a blockcipher with block-length n). Then the advantage of an adversary in finding a collision in H^E after q queries can be upper bounded by

$$\text{Adv}_H^{\text{coll}}(q) \leq 2q(q-1)/(2^n - 2q)^2 .$$

where $q < 2^{n-1}$.

Proof. Let \mathcal{A} be a collision-finding adversary asking its oracles E and D a total of q queries. We will consider the derived adversary \mathcal{A}' that asks conjugate queries in pairs: i.e. if \mathcal{A} queries (K, X) to E , then \mathcal{A}' queries both (K, X) and its conjugate $(K', X') = p(K, X)$. We will bound the probability of the i 'th query pair causing a collision.

Our claim is that the probability that the i 'th query pair causing a collision is upper bounded by $4(i-1)/(2^n - (2i-2))(2^n - (2i-1))$. First note that the new query pair will add two tuples (M, V, W) and (M', V', W') to the list of computable digests, where (M, V) and (M', V') are conjugate. Moreover (by requirement 5) we are guaranteed

that $W \neq W'$, so we only need to worry about either W or W' being in the list of digests already known.

Since each query pair adds two tuples to the list, there are $2(i - 1)$ possible digest values to hit, either by (M, V, W) or by (M', V', W') . We proceed similar to the proof of Theorem 5. Although the two distributions are strongly dependent, individually each of W and W' will be distributed over a set of size at least $(2^n - (2i - 2))(2^n - (2i - 1))$ (see below). Using a union bound, the probability of a collision on the i 'th query can therefore be upper bounded by $4(i - 1)/(2^n - 2i + 1)^2$ and using a further union bound, the probability of a collision after q queries can be upper bounded by $2q(q - 1)/(2^n - 2q)^2$. We still need to prove our claim on the distribution of W and W' .

For a forward-forward pair, Y_1 is distributed uniformly random over a set of size at least $(2^n - (2i - 2))$ and Y_2 over a set of size at least $2^n - (2i - 1)$ (assuming Y_1 is queried first, otherwise the roles change). Because C^{POST} is bijective the claim follows.

For a mixed inverse-forward pair first X_1 is returned by a query (K_1, Y_1) to D . It is distributed uniformly over a set of size at least $(2^n - (2i - 2))$. Subsequently Y_2 is returned by the conjugate query (K_2, X_2) to E ; it is distributed uniformly over a set of size at least $(2^n - (2i - 1))$. This query pair leads to $W = C^{\text{POST}}(C_1^{-\text{PRE}}(K_1, X_1), Y_1, Y_2)$ and $W' = C^{\text{POST}}(C_2^{-\text{PRE}}(K_1, X_1), Y_2, Y_1)$. Bijectivity of C_1^{AUX} and C_2^{AUX} does the rest. \square

Preimage resistance. The following result gives an upper bound for $\text{Adv}_H^{\text{epr}}(q)$ for IDS-double-length Type-I compression functions. Similar to the one given in Theorem 6, it is vacuous for $q \geq 2^n$ and only concludes that the success probability of finding a preimage is negligible for $q \leq 2^{n-1}$.

Theorem 8. *Let H^E be a double-length compression function of IDS-double-length Type-I (based on a blockcipher with block-length n). Then the advantage of an adversary in finding a preimage in H^E after q queries can be upper bounded by*

$$\text{Adv}_H^{\text{epr}}(q) \leq 2q/(2^n - 2q)^2.$$

Proof. Let \mathcal{A} be an adversary trying to find a preimage for its input σ . We will assume that \mathcal{A} asks each of its oracles E and D a total of q queries. Similar to the proof of Theorem 7, we will consider the derived adversary \mathcal{A}' that asks conjugate queries in pairs. We want to bound the probability that the i 'th query pair leads to a preimage for σ .

After $i - 1$ queries, we know that the list of computable values contains exactly $2(i - 1)$ triples (M, V, W) . The i 'th query will add the pair (M, V, W) and (M', V', W') to the list of computable digests, where (M, V) and (M', V') are conjugates. Our claim is that the probability that the i 'th query pair leads to a preimage for σ is upper bounded by $2/(2^n - (2i - 2))(2^n - (2i - 1))$: By the same reasoning from the proof of Theorem 7, W and W' are distributed over a set of size at least $(2^n - (2i - 2))(2^n - (2i - 1))$. As we have only one value to hit, the probability of finding a preimage for each conjugate queries is at most $1/(2^n - (2i - 2))(2^n - (2i - 1))$; hence twice this probability for the i 'th pair. Union bound gives the desired result. \square

5 Collision Resistance in the Iteration

We have seen how to construct collision-resistant compression functions using two blockcipher calls and how to achieve collision resistance in the iteration using just one blockcipher call (per compression function). In this section, we consider security in the iteration for two blockcipher calls. The hope is that this will lead to new schemes with less overhead than existing options.

In the definition below, we explicitly disallow a feedforward from the input of the compression function to the postprocessing (thus making the Type-II schemes disjunct from the Type-I schemes). From a practical point of view, not having a feedforward is certainly good news. From a theoretical point of view, it allows us to obtain a tighter bound. If we would allow the usual postprocessing, we can only prove $\text{Adv}_{\mathcal{H}}^{\text{coll}}(q) \leq q^3/(2^n - q)^2$, which only gives a collision resistance guarantee up to roughly $2^{2n/3}$ queries.

The bound we give on the collision resistance is still somewhat removed from optimal: it only provides collision resistance up to roughly $2^n/n$ queries. Moreover, for smaller number of queries the advantage is roughly the square root of what it should be (i.e. too big). We do not believe that this is a problem inherent with the scheme at hand, but rather a shortcoming of the current proof and testament to the complications that arise when looking at security in the iteration.

Definition 8. A 2-call blockcipher-based compression function is called double-length Type-II iff $s = k = 2n$, $m = n$, and the following three hold:

1. C_1^{PRE} and C_2^{PRE} are both bijections;
2. $C^{\text{POST}}(Y_1, Y_2)$ is a bijection (from $2n$ to $2n$ bits; for all (Y_1, Y_2)) and independent of V and M ;
3. $C_1^{-\text{PRE}}(K_1, \cdot)$ and $C_2^{-\text{PRE}}(K_2, \cdot)$ are injections when restricted to V with effective ranges $R_{-\text{PRE},(K_1)}$ respectively $R_{-\text{PRE},(K_2)}$. Assume that there are at most 2^n different effective ranges (when considering different keys).

Theorem 9. Let H^{E_1, E_2} be a double-length Type-II compression function. Then the advantage of an adversary in finding a collision in the iterated hash function \mathcal{H}^H after q queries is upper bounded by

$$\text{Adv}_{\mathcal{H}}^{\text{coll}}(q) \leq 2\kappa q/(2^n - q) + q^2/(2^n - q)^2 + 2^{n+1} \left(\frac{e2^n q}{\kappa(2^n - q)^2} \right)^\kappa$$

for any positive integer κ .

Proof. We follow the idea of the proof of Theorem 3 to prove the desired result and as in the proof of Theorem 5 we will consider a derived adversary only that makes corresponding queries to E_1 and E_2 in conjunction (since C_1^{PRE} and C_2^{PRE} are bijections as usual). Thus, the derived adversary either makes a query pair consisting of two forward queries, or a mixed pair consisting of an inverse query followed by a forward query.

We define a directed graph $G = (V_G, E_G)$ with vertex set $V_G = \{0, 1\}^s$, corresponding to all 2^s possible chaining values, and initially an empty edge set $E_G = \emptyset$.

We will dynamically add edges based on the queries to E_1, E_2 and/or D_1, D_2 . In particular, we add an arc (V, W) , labeled by M , if we know a message M such that $W = H^{E_1, E_2}(M, V)$ and the relevant queries to E_1, E_2 and/or D_1, D_2 have been made. If an arc is added as a consequence of a forward query pair, then the node corresponding to W becomes a forward node. If an arc is added as a consequence of a mixed query pair, both nodes become mixed nodes. Initially none of the nodes has a status except for the initial vector, which is treated as a forward node.

Our claim is that in order to find a collision, one of the following three (bad) events needs to occur (on the i 'th query pair, for some i):

1. A forward query pair leads to a digest that is already a forward node (i.e. previously resulted as digest of a forward query pair, or equals the initial value).
2. A forward query pair leads to a digest that is already a mixed node (i.e. previously resulted as digest or chaining variable of a mixed query pair).
3. A mixed query pair leads to a digest or chaining variable that is already a forward node (i.e. previously resulted as digest of a forward query pair, or equals the initial value).

In particular, we can ignore the event that a mixed query pair leads to a digest or chaining variable that is already a mixed node. To see this, first observe that these are all the four possibilities to construct longer chains within the graph. Secondly, by allowing an arbitrary number of occurrences of *only* the fourth (discarded) event a collision can never be found for the simple reason that the initial vector will not be part of the chain (since hitting the initial vector with a mixed query would have triggered the third event).

We will now bound the probability of any of the bad events occurring on the i 'th query pair.

1. Consider a forward query pair to E_1 and E_2 , corresponding to a unique pair (M, V) . Suppose that so far $t_1 \leq i$ queries to E_1 have been made involving key K_1 . The answer Y_1 to a fresh query to $E_1(K_1, \cdot)$ will therefore be distributed uniformly over a set of $2^n - t_1$ possible outcomes. Similarly, suppose that $t_2 \leq i$ queries to E_2 have been made involving key K_2 , so that the answer Y_2 to $E_2(K_2, \cdot)$ is uniform over a set of $2^n - t_2$. Moreover, Y_1 and Y_2 are independent, so that the probability that (Y_1, Y_2) hits any particular value in $\{0, 1\}^{2n}$ is at most $1/(2^n - i)^2$. Since C^{POST} is bijective when the pair (M, V) is fixed, the probability that any particular digest W is hit is at most $1/(2^n - i)^2$. There are at most i possible W 's labeled as a forward node, therefore, the probability of i 'th forward pair prompting this bad event at most $i/(2^n - i)^2$.
2. Again, consider a forward query pair to E_1 and E_2 , corresponding to a unique pair (M, V) . Since there are at most $2(i - 1)$ nodes labeled as mixed node at this point, the probability of the i 'th forward pair prompting this bad event is at most $2(i - 1)/(2^n - i)^2$.
3. Finally consider a mixed query pair. We will assume it consists of a D_1 -query followed by an E_2 -query, the other case is analogous. This time we can argue that X_1 and Y_2 are both distributed uniformly and independently over sets of cardinality at least $2^n - i$. Consequently, the same holds for chaining variable V and digest W . Since there are at most i forward nodes that can be hit, this gives (with a union

bound) a probability of success upper bounded by $2i/(2^n - i)$. Unfortunately, this upper bound is too lose to be useful.

Although there are up to i forward nodes, not all of them are relevant. Indeed, the query to D_1 essentially fixes K_1 and Y_1 . Since C^{POST} is bijective, this leaves only 2^n possible digests. Similarly, since $C_1^{-\text{PRE}}$ is injective, it leaves only 2^n possible chaining variables. Thus, we only need to consider the number of forward nodes in the effective range (of C^{POST} and $C_1^{-\text{PRE}}$ resp.) after fixing (K_1, Y_1) . Let this number be κ (maximum for C^{POST} or $C_1^{-\text{PRE}}$), then the probability of the bad event occurring (on the i 'th mixed query) is at most $2\kappa/(2^n - i)$.

The question is whether good bounds on the number κ of relevant forward nodes can be found. Let us fix some K_1 and consider the probability a forward query lands in $R_{-\text{PRE},(K_1)}$. We have already seen that a forward query leads to an almost uniform distribution of the resulting digest and we know that $R_{-\text{PRE},(K_1)}$ has cardinality 2^n . Hence if the total number of forward query pairs is q , then each query pair has a probability at most $(2^n)/(2^n - q)^2$ to hit our set (effective range), so using a Chernoff bound we have that for any given set the probability it is hit more than κ times is at most

$$\left(\frac{e2^n q}{\kappa(2^n - q)^2} \right)^\kappa.$$

Since there are at most 2^{n+1} different sets of size 2^n to consider we can apply a union bound, resulting in an extra factor 2^{n+1} .

Picking up the various probabilities proves the theorem statement. \square

6 Implications for Linear Schemes

Here we investigate the implications of our results for rate-1/2 double-length compression functions with \mathbb{F}_2 -‘block’-linear pre- and postprocessing functions. (The conditions for the supercharged, rate-1 construction cannot be met by a purely \mathbb{F}_2 -‘block’-linear construction.)

Compression function security with two independent blockciphers. Hirose [11] suggested a method to construct double-length Type-I compression functions that are optimally collision resistant with \mathbb{F}_2 -‘block’-linear instances of C^{PRE} and C^{POST} and based on two different blockciphers with $2n$ -bit keys and n -bit blocks. Our constructions are a generalization of his work and indeed, when we restrict ourselves to \mathbb{F}_2 -‘block’-linear constructions without output mixing ($T_1 = (10)$ and $T_2 = (01)$ in the notation below), we get exactly the same set of schemes as Hirose did before us.

Let us first set up some notation. We will treat V and W as $V = (V_1||V_2)$ and $W = (W_1||W_2)$ and, as is customary [13] for schemes with linear processing C^{PRE} and C^{POST} , we will represent the schemes using matrices.

We will use $\overset{3 \times 3}{2}$ to express the way (K_i, X_i) are functions of M and (V_1, V_2) for $i \in \{1, 2\}$. Thus C_1^{PRE} is represented by matrix $(\overset{K_1}{X_1})$ with $K_1 \in \overset{2 \times 3}{2}$ and $X_1 \in \overset{1 \times 3}{2}$. The vector $X_1 \in \overset{3}{2}$ corresponds to $X_1 = X_1 \cdot (M, V_1, V_2)^T$, making a distinction between the linear map $X_1 \in \overset{2}{2}$ and the value $X_1 \in \{0, 1\}^n$. (Similarly C_2^{PRE} is

represented by \mathbf{K}_2 and \mathbf{X}_2 .) Postprocessing C^{POST} is represented by matrices $\begin{pmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{pmatrix} \in \mathbb{F}_2^{2 \times 2}$ and $\begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{pmatrix} \in \mathbb{F}_2^{2 \times 3}$, where

$$\begin{pmatrix} W_1 \\ W_2 \end{pmatrix} = \begin{pmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{pmatrix} \cdot \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} + \begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{pmatrix} \cdot \begin{pmatrix} V_1 \\ V_2 \\ M \end{pmatrix}.$$

We are now ready to see what the requirements from Definition 6 mean in terms of \mathbf{K}_i , \mathbf{X}_i , \mathbf{U}_i , and \mathbf{T}_i and hence for the classification and security of existing schemes.

Lemma 2. *A \mathbb{F}_2 -‘block’-linear double-length blockcipher-based compression function is double-length Type-I iff*

1. $\begin{pmatrix} \mathbf{K}_1 \\ \mathbf{X}_1 \end{pmatrix}$ and $\begin{pmatrix} \mathbf{K}_2 \\ \mathbf{X}_2 \end{pmatrix}$ are both invertible matrices.
2. $\begin{pmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{pmatrix}$ is an invertible matrix.
3. $\begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{K}_1 \\ \mathbf{X}_1 \end{pmatrix}^{-1} \begin{pmatrix} K_1 \\ X_1 \end{pmatrix} + \begin{pmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{pmatrix} \cdot \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}$ is bijective as a function of X_1 and Y_2 for all K_1 and Y_1 .
4. $\begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{K}_2 \\ \mathbf{X}_2 \end{pmatrix}^{-1} \begin{pmatrix} K_2 \\ X_2 \end{pmatrix} + \begin{pmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{pmatrix} \cdot \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}$ is bijective as a function of X_2 and Y_1 for all K_2 and Y_2 .

Proof. We will check the three conditions in Definition 6. Firstly, C_1^{PRE} and C_2^{PRE} are bijections. This is equivalent to $\begin{pmatrix} \mathbf{K}_1 \\ \mathbf{X}_1 \end{pmatrix}$ and $\begin{pmatrix} \mathbf{K}_2 \\ \mathbf{X}_2 \end{pmatrix}$ being invertible. Second condition is $C^{\text{POST}}(M, V, \cdot, \cdot)$ ’s being a bijection (from $2n$ to $2n$ bits, for all (M, V)) which is the case iff $\begin{pmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{pmatrix}$ is invertible. Finally, we should have C_1^{AUX} and C_2^{AUX} are bijections for all K_1, Y_1 and K_2, Y_2 respectively, which is simply rephrased to the linear case in the third and fourth requirements from the lemma.

For the single-length PGV schemes it is possible to give a much cleaner description of what corresponds to the third and fourth requirement, stating it as a condition on the binary vectors \mathbf{X} , \mathbf{K} , and \mathbf{U} alone. Here it would be possible to derive such statements, for instance for the third requirement by symbolically inverting the 3-by-3 matrix $\begin{pmatrix} \mathbf{K}_1 \\ \mathbf{X}_1 \end{pmatrix}$, isolating its third column and multiplying by $\begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{pmatrix}$ to obtain a vector in \mathbb{F}_2^2 , which should be linearly independent from the second column of $\begin{pmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{pmatrix}$ (this column is typically equal to (01) , simplifying the statement further). A derivation for the fourth requirement is similar. We also note that enumerating all possibilities (as PGV did for the single-length case) is becoming rather laborious: there are a total of 2^{24} schemes (compared to 64) of which a large number satisfy the requirements just laid out.

To make the discussion above more concrete, we will consider a tweaked version of Abreast-DM, dubbed *Abreast-DM-t*, that can be proven collision resistant using our framework (it was also considered by Hirose [11]). Abreast-DM itself, designed by Lai and Massey [16], is one of the early double-length blockcipher-based compression function proposals. Lai and Massey conjectured it to be optimally collision and preimage resistant as they could not mount a successful attack on the design. Abreast-DM is defined as follows

$$\begin{aligned} W_1 &= V_1 \oplus E(V_2 \parallel M, V_1) \\ W_2 &= V_2 \oplus E(M \parallel V_1, V_2 \oplus 1^n). \end{aligned}$$

As it stands, Abreast-DM does not fit the framework above since it uses the same blockcipher twice and the bitwise complement $V_2 \oplus 1^n$ is not covered by linear transformations (it would require a generalization to affine transformations). Let us therefore define the tweaked version Abreast-DM-t as follows:

$$\begin{aligned} W_1 &= V_1 \oplus E_1(V_2 \parallel M, V_1) \\ W_2 &= V_2 \oplus E_2(M \parallel V_1, V_2). \end{aligned}$$

According to the notation introduced above, we have

$$\begin{aligned} \mathbf{K}_1 &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{X}_1 = (1 \ 0 \ 0), \mathbf{U}_1 = (1 \ 0 \ 0), \mathbf{T}_1 = (1 \ 0), \\ \mathbf{K}_2 &= \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \mathbf{X}_2 = (0 \ 1 \ 0), \mathbf{U}_2 = (0 \ 1 \ 0), \mathbf{T}_2 = (0 \ 1). \end{aligned}$$

So, $(\frac{\mathbf{K}_1}{\mathbf{X}_1})$, $(\frac{\mathbf{K}_2}{\mathbf{X}_2})$, and $(\frac{\mathbf{T}_1}{\mathbf{T}_2})$ are all invertible matrices. Since the latter is the identity matrix, we can slightly simplify our check for the third and fourth requirement. For the third condition, we then need that

$$\mathbf{U}_1 \cdot \left(\frac{\mathbf{K}_1}{\mathbf{X}_1} \right)^{-1} \begin{pmatrix} K_1 \\ X_1 \end{pmatrix} = (1 \ 0 \ 0) \cdot \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} K_1^1 \\ K_1^2 \\ X_1 \end{pmatrix} = X_1,$$

depends on X_1 (which it clearly does). Similarly, the fourth condition verifies since

$$\mathbf{U}_2 \cdot \left(\frac{\mathbf{K}_2}{\mathbf{X}_2} \right)^{-1} \begin{pmatrix} K_2 \\ X_2 \end{pmatrix} = (0 \ 1 \ 0) \cdot \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} K_2^1 \\ K_2^2 \\ X_2 \end{pmatrix} = X_2.$$

depends on X_2 . Therefore, Abreast-DM-t satisfies the requirements of Lemma 2 and the bounds from Theorems 5 and 6 apply.

It should be noted that recently ‘untweaked’ Abreast-DM was shown optimally collision resistant up to a small constant factor as well [9, 17]: the collision finding advantage can be upper bounded by $\frac{1}{2}(6q/(2^n - 6q))^2$ for $6q < 2^n$.

Compression function security with a single blockcipher. Hirose’s [12] blockcipher-based double-length compression function is one of the existing schemes having optimal collision resistance which is covered by Definition 7 (and a suitable adaptation of Lemma 2). Recall the scheme:

$$\begin{aligned} W_1 &= V_1 \oplus E(V_2 \parallel M, V_1) \\ W_2 &= V_1 \oplus c \oplus E(V_2 \parallel M, V_1 \oplus c) \text{ where } c \text{ is a nonzero constant.} \end{aligned}$$

Let us check the requirements given in Definition 7.

$$\begin{aligned} p(V_2 \parallel M, V_1) &= (V_2 \parallel M, V_1 \oplus c) \text{ where } c \text{ is a nonzero constant,} \\ \mathbf{K}_1 &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{X}_1 = (1 \ 0 \ 0), \mathbf{U}_1 = (1 \ 0 \ 0), \end{aligned}$$

Clearly, p is an involution without fixed points and $\binom{K_1}{X_1}$ is invertible. For C_2^{PRE} , we have $C_2^{\text{PRE}}(M, V_1 \parallel V_2) = (M, V_1 \oplus c \parallel V_2)$ which is indeed a bijection. What is left to check or the bijectivity requirements for C_1^{AUX} and C_2^{AUX} . The requirement for C_1^{AUX} follows from

$$Y_1 \oplus \mathbf{U}_1 \cdot \begin{pmatrix} \mathbf{K}_1 \\ \mathbf{X}_1 \end{pmatrix}^{-1} \begin{pmatrix} K_1 \\ X_1 \end{pmatrix} = Y_1 \oplus \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} K_1^1 \\ K_1^2 \\ X_1 \end{pmatrix} = Y_1 \oplus X_1,$$

being a bijection as a function of X_1 for all K_1 and Y_1 . Finally, C_2^{AUX} is defined as $C_2^{\text{AUX}} = C^{\text{POST}}(C_2^{\text{-PRE}}(K_2, \cdot), Y_2) = Y_2 \oplus X_2$ which is also a bijection as a function of X_2 for all Y_2 . Therefore, Hirose's scheme satisfies the requirements of Theorem 7.

7 Conclusion

We have presented a general framework for double-length blockcipher-based hash functions by extending the recent generalization of Stam [27] for single-call blockcipher-based hash functions. We analysed sufficient conditions for the pre- and postprocessing functions to obtain close to optimal collision resistance, either in the compression function or in the iteration.

We mainly targeted compression functions compressing $3n$ -bits to $2n$ -bits that use one or two calls to a $2n$ -bit key, n -bit block blockcipher. In case of a single call, we restricted ourselves to security in the iteration, updating prior work by Lucks [19]. For two calls, we restricted ourselves to parallel calls, either to two distinct and independent blockciphers or to a single blockcipher. Optimal collision resistance (up to a small multiplicative factor in the advantage) was shown for compression functions. For the iterated case, we gave conditions that we believe are sufficient for a similar bound, but we only showed collision resistance up to about $2^n/n$ queries, leaving open the problem of finding a tighter bound.

References

- Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
- 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)
- Bracht, B., Coppersmith, D., Hyden, M., Matyas Jr., S., Meyer, C., Oseas, J., Pilpel, S., Schilling, M.: Data authentication using modification detection codes based on a public one-way encryption function. U.S. Patent No 4,908,861 (March 1990)
- Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
- Damgård, I.: A design principle for hash functions. In: Brassard (ed.) [4], pp. 416–427.
- Dunkelman, O. (ed.): FSE 2009. LNCS, vol. 5665. Springer, Heidelberg (2009)
- Duo, L., Li, C.: Improved collision and preimage resistance bounds on PGV schemes. Technical Report 462, IACR's ePrint Archive (2006)

8. Fleischmann, E., Gorski, M., Lucks, S.: On the security of Tandem-DM. In: Dunkelman (ed.) [6] (to appear)
9. Fleischmann, E., Gorski, M., Lucks, S.: Security of cyclic double block length hash functions including Abreast-DM. Technical Report 261, IACR's ePrint Archive (2009)
10. Hattori, M., Hirose, S., Yoshida, S.: Analysis of double block length hash functions. In: Patterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 290–302. Springer, Heidelberg (2003)
11. Hirose, S.: Provably secure double-block-length hash functions in a black-box model. In: Park, C., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 330–342. Springer, Heidelberg (2005)
12. Hirose, S.: Some plausible constructions of double-length hash functions. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
13. Hohl, W., Lai, X., Meier, T., Waldvogel, C.: Security of iterated hash functions based on block ciphers. In: Stinson (ed.) [29], pp. 379–390
14. Knudsen, L., Lai, X., Preneel, B.: Attacks on fast double block length hash functions. *Journal of Cryptology* 11(1), 59–72 (1998)
15. 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)
16. Lai, X., Massey, J.L.: Hash function based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
17. Lee, J., Kwon, D.: The security of abreast-dm in the ideal cipher model. Technical Report 225, IACR's ePrint Archive (2009)
18. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
19. Lucks, S.: A collision-resistant rate-1 double-block-length hash function. In: Biham, E., Handschuh, H., Lucks, S., Rijmen, V. (eds.) Symmetric Cryptography. Dagstuhl Seminar Proceedings, Dagstuhl, Germany, vol. 07021, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl (2007)
20. Merkle, R.C.: A certified digital signature. In: Brassard (ed.) [4], pp. 218–238
21. Merkle, R.C.: One way hash functions and DES. In: Brassard [4], pp. 428–446
22. Nandi, M.: Towards optimal double-length hash functions. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 77–89. Springer, Heidelberg (2005)
23. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson (ed.) [29], pp. 368–378
24. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
25. Rogaway, P., Steinberger, J.: Constructing cryptographic hash functions from fixed-key blockciphers. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 433–450. Springer, Heidelberg (2008)
26. Satoh, T., Haga, M., Kurosawa, K.: Towards secure and fast hash functions. *IEICE Transactions, Special Section on Cryptography and Information Security* E82-A(1) (1999)
27. Stam, M.: Block cipher based hashing revisited. In: Dunkelman (ed.) [6], pp. 67–83
28. Steinberger, J.: 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)
29. Stinson, D.R. (ed.): CRYPTO 1993. LNCS, vol. 773. Springer, Heidelberg (1994)

A Related Work

Classical single-call hash functions. Preneel et al. [23] studied the security of the general blockcipher-based compression function $H(M, V) = E(K, X) \oplus U$ where $U \in \{0, M, V, M \oplus V\}$ (plus affine offsets thereof). The resulting 64 schemes are called the PGV-schemes (and include Davies-Meyer, Matyas-Meyer-Oseas, and Preneel-Miyaguchi). Preneel et al.’s approach was attack oriented: they successfully mounted attacks on 52 of the compression functions and concluded that the remaining, unbroken 12 were collision resistant. This work was later validated by Black et al. [1] who proved collision resistance of these 12 schemes up to the birthday bound (in the ideal cipher model). Additionally, they showed that another 8 PGV-schemes are secure when properly iterated, even though collisions in the respective compression functions can easily be found. Duo and Li [7] later gave slightly tighter bounds by providing an alternative proof technique.

A deeper understanding of the main result provided by Black et al. has recently been given by Stam [27], who generalizes to any pre- and postprocessing functions. This provides a clearer understanding of which properties of the pre- and postprocessing are sufficient for optimal security (in particular the 12 + 8 secure PGV-schemes can be derived from this framework) and it eases the study of compression functions with deviating input and output lengths.

Double-call hash functions from double-key blockciphers. The two classical hash functions of this type are Tandem-DM and Abreast-DM [16]. Both schemes employ a single blockcipher called twice, where for Tandem-DM the calls are made in sequence and for Abreast-DM they are made in parallel. Although both are widely believed to be (close to) optimally collision resistant, for a long time no security proof of this was known for either construction. Only recently Fleischmann et al. [8] gave a proof of collision resistance up to (almost) the birthday bound for Tandem-DM; a proof for the collision resistance up to (almost) the birthday bound for Abreast-DM is even more recent [9, 17]. For both schemes proving preimage resistance beyond the birthday bound is still an open problem.

There are some other double-length hash functions based on two calls to a double-key blockcipher that enjoy collision resistance up to the birthday bound. Hirose [11] proposed several conditions to achieve optimal security for the linear instances of pre- and postprocessing functions for rate-1/2 in this category. In a way, these schemes can be regarded for this class what the PGV schemes are for the classical, single-length setting. In his constructions, the only drawback is that two independent blockciphers are called which is sometimes considered a disadvantage. Later, taking inspiration from earlier work by Nandi [22], Hirose [12] presented a rate-1/2 scheme with optimal collision resistance using only one blockcipher.

Satoh et al. [26] mounted several collision and preimage attacks on the compression functions of double-call hash functions from double-key blockciphers where only one blockcipher is used. They also provided a set of conditions for the case where the message length is equal to digest length which may be optimally collision resistant and left its investigation as a future work. This work has been further analysed by Hattori et al. [10]. They first investigated a case uncovered by the analysis of Satoh et al., then

showed that the compression functions which were not attacked by Satoh et al. are at most as secure as those of single-length hash functions.

Other double-length hash functions. There are two classical double-length compression functions based on a blockcipher with key size equaling the block size n , namely MDC-2 and MDC-4 [3], where MDC-2 makes two calls and MDC-4 four calls to compress a single message block.² The compression function of MDC-2 is known to be collision resistant only up to $2^{n/2}$ (thus offering no improvement over single-length constructions), but the iterated hash was widely believed to provide near optimal collision resistance. Steinberger [28] gave the first nontrivial result in the ideal cipher model for the collision resistance of MDC-2 by showing that an adversary asking fewer than $2^{3n/5}$ queries has only a negligible chance of finding a collision in the iteration. Knudsen et al. [15] recently gave a collision attack of complexity approximately $2^n/n$, nibbling a logarithmic factor of the optimal birthday bound (they also gave a preimage attack on MDC-2). Thus, there remains a large gap with respect to the security of MDC-2. For a more complicated (yet less efficient) scheme like MDC-4 even less is known.

Knudsen et al. [14] studied the security of double-length hash functions that compress two blocks of message for two calls to an underlying blockcipher with key size corresponding to the block size. They show that for all \mathbb{F}_2 -‘block’-linear schemes collisions can be found (with high probability) in time $3 \times 2^{3n/4}$ and preimages in time 4×2^n .

Rogaway and Steinberger [25] investigated the collision and preimage resistance of compression functions built from fixed-key blockciphers. They demonstrated compression functions compressing $3n$ bits to $2n$ bits using five, respectively six permutation calls with collision resistance up to at least $2^{0.55n}$, respectively $2^{0.63n}$ queries and preimage resistance up to at least $2^{0.8n}$ (for both constructions).

² MDC-2 and MDC-4 were originally designed for use with DES (Data Encryption Standard) which supports 56 bits key and 64 bits block; the generalization allowing key size equal to the block size is immediate.

Geometric Ideas for Cryptographic Equation Solving in Even Characteristic

Sean Murphy and Maura B. Paterson*

Information Security Group, Dept. of Mathematics,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, U.K.

Abstract. The GeometricXL algorithm is a geometrically invariant version of the XL algorithm that uses polynomials of a much smaller degree than either a standard Groebner basis algorithm or an XL algorithm for certain multivariate equation systems. However, the GeometricXL algorithm as originally described is not well-suited to fields of even characteristic. This paper discusses adaptations of the GeometricXL algorithm to even characteristic, in which the solution to a multivariate system is found by finding a matrix of low rank in the linear span of a collection of matrices. These adaptations of the GeometricXL algorithm, termed the EGHAM process, also use polynomials of a much smaller degree than a Groebner basis or an XL algorithm for certain equation systems. Furthermore, the paper gives a criterion which generally makes a Groebner basis or standard XL algorithm more efficient in many cryptographic situations.

Keywords: XL Algorithm, GeometricXL Algorithm, EGHAM process.

1 Introduction

The solution of a system of multivariate equations over a field is a problem that has recently attracted much attention in cryptology. The classical method for analysing such an equation system is to calculate its Gröbner basis by using Buchberger's algorithm or a related method [2,6,7]. Furthermore, other techniques have been proposed for solving a multivariate equation system in a cryptographic context, such as the XL algorithm [5]. However, a Gröbner basis algorithm with the lexicographic ordering and an XL algorithm are closely related [1].

The geometric properties of the XL algorithm are discussed in [12]. In particular, the XL algorithm is not a geometrically invariant algorithm, that is a simple change of co-ordinate system can vastly increase or decrease the running time of the XL algorithm. The GeometricXL algorithm, proposed in [12], is a geometrically invariant algorithm which can solve certain multivariate equation systems using polynomials of a much smaller degree than a Gröbner basis algorithm or

* M.B. Paterson was supported by EPSRC grants GR/S42637 and EP/D053285.

an XL algorithm. However, the **GeometricXL** algorithm as described in [12] cannot easily be used in a field of even characteristic, which is of course a situation of great cryptographic importance. The main contribution of this paper is to give an adaptation of the **GeometricXL** algorithm that is specifically tailored for use in a field of even characteristic. This adaptation of the **GeometricXL** algorithm, like the original **GeometricXL** algorithm, is one that attempts to find a linear combination of a collection of matrices that has low rank, a problem sometimes termed **MinRank**. We note that some related issues concerning the **MinRank** problem in cryptology are considered in [8]. We term the adaptation of the **GeometricXL** algorithm given in this paper the **EGHAM** process, and we note that the **EGHAM** process can solve certain multivariate equation systems in even characteristic using polynomials of a much smaller degree than a Gröbner basis algorithm or an XL algorithm. Furthermore, a criterion (the \mathcal{LS} -criterion) used by the **EGHAM** process generally greatly reduces the number of equations under consideration. This reduction criterion can also be applied directly to a standard Gröbner basis or XL algorithm in many cryptographic situations, so directly making these algorithms far more efficient.

2 The XL Algorithm

We consider the polynomial ring $\mathbb{F}[x_0, \dots, x_n]$ of polynomials in $n + 1$ variables over a field \mathbb{F} . An XL algorithm transforms a homogeneous (without loss of generality) equation system into a homogeneous equation system $f_1 = \dots = f_m = 0$ of degree D by multiplying the original polynomials by selected monomials [5]. We generally suppose that this equation system has a unique (projective) solution, a common situation in cryptology, though most of our comments are more generally applicable. The aim of an XL algorithm is to solve this new system by linearisation [5], that is by regarding each monomial of degree D as an independent variable and then applying basic linear algebra. The **GeometricXL** algorithm [12] exploits the geometrical properties of the new equation system to give a solution method that is most applicable when the field characteristic is either zero or exceeds D . We discuss the **GeometricXL** algorithm in this case in this section and give an alternative description of the **GeometricXL** algorithm.

2.1 The GeometricXL Algorithm

The critical step of the XL algorithm [12] is an attempt to solve a system of homogeneous equations $f_1 = \dots = f_m = 0$ of degree D in a field of characteristic p by finding a bivariate polynomial in $\langle f_1, \dots, f_m \rangle$. However, the set of bivariate polynomials is not invariant under collineation (change of co-ordinates), so the XL algorithm is not geometrically invariant. The **GeometricXL** algorithm is a geometric invariant generalisation of the XL algorithm. The **GeometricXL** algorithm focusses on *rank-2 product polynomials* (Definition 1), an invariant generalisation of the bivariate polynomial, and the **GeometricXL** algorithm is motivated by Lemma 1, the geometric invariance result of [12].

Definition 1. A *rank-2 product polynomial* is a homogeneous polynomial of degree D in the polynomial ring $\mathbb{F}[x_0, \dots, x_n]$ of the form

$$\prod_{i=1}^D (\theta_i L - \theta'_i L'),$$

where L and L' are homogeneous linear polynomials and θ_i, θ'_i are constants in some extension field $\bar{\mathbb{F}}$ of \mathbb{F} . We let $\mathcal{R}_{\mathbb{F},n}^D$ denote the set of all such rank-2 product polynomials of degree D in $\mathbb{F}[x_0, \dots, x_n]$.

Lemma 1. The set $\mathcal{R}_{\mathbb{F},n}^D$ of all rank-2 product polynomials of degree D in the polynomial ring $\mathbb{F}[x_0, \dots, x_n]$ is invariant under collineation.

The **GeometricXL** algorithm requires us to find a linear combination g of f_1, \dots, f_m such that $g = \sum_{l=1}^m \lambda_l f_l$ is a rank-2 product polynomial, that is $g \in \mathcal{R}_{\mathbb{F},n}^D$. If such a rank-2 product polynomial can be found, then we know that any solution to the original system $f_1 = \dots = f_m = 0$ satisfies $\theta_i L - \theta'_i L' = 0$ for some value of i . This gives us linear expressions in x_0, x_1, \dots, x_n , which provides information about the solution and potentially allows us to eliminate one variable from the equation system, giving us a smaller equation system and so on.

For any homogeneous polynomial h of degree d and any monomial $\mathbf{x} = x_0^{e_0} \dots x_n^{e_n}$ of degree $k \leq d$ (so $e_0 + \dots + e_n = k$), we denote the k^{th} order partial derivative of h with respect to \mathbf{x} by $\mathbf{D}_{\mathbf{x}}^k h$, so $\mathbf{D}_{\mathbf{x}}^k h = \frac{\partial^k h}{\partial \mathbf{x}^k}$. We can now represent all the possible k^{th} order partial derivatives of h as a matrix in which each row is the polynomial $\mathbf{D}_{\mathbf{x}}^k h$ of degree $d - k$ represented as a vector of its coefficients. There are $\binom{n+k}{k}$ monomials of degree k in $n+1$ variables, so we can define an $\binom{n+k}{k} \times \binom{n+d-k}{d-k}$ partial derivatives matrix

$$C_h^{(k)} = (\mathbf{D}_{\mathbf{x}}^k h).$$

The **GeometricXL** algorithm works as a rank-2 product polynomial can be identified using Lemma 2 (given by results of [12]) from its partial derivatives of order $D - 1$ in fields of certain characteristics.

Lemma 2. Suppose g is a homogeneous polynomial of degree D in $\mathbb{F}[x_0, \dots, x_n]$, where the field \mathbb{F} has characteristic zero or characteristic exceeding D . The polynomial g is a rank-2 product polynomial if and only if the partial derivatives matrix $C_g^{(D-1)}$ satisfies

$$C_g^{(D-1)} = (\mathbf{D}_{\mathbf{x}}^{D-1} g) = (a_{\mathbf{x}} L + a'_{\mathbf{x}} L')$$

for some homogeneous linear polynomials L and L' and constants $a_{\mathbf{x}}$ and $a'_{\mathbf{x}}$. An equivalent condition is that for a field \mathbb{F} with characteristic zero or characteristic exceeding D , then $g \in \mathcal{R}_{\mathbb{F},n}^D$ if and only if $C_g^{(D-1)}$ has rank at most 2. Moreover, if $g \in \mathcal{R}_{\mathbb{F},n}^D$ then $C_g^{(D-1)}$ has rank at most 2 in a field \mathbb{F} of any characteristic.

The **GeometricXL** algorithm attempts to find some linear combination of f_1, \dots, f_m such that $g = \sum_{l=1}^m \lambda_l f_l \in \mathcal{R}_{\mathbb{F},n}^D$. The same linear combination of partial derivatives matrices of f_1, \dots, f_m satisfies

$$C_g^{(D-1)} = (\mathbf{D}_x^{D-1} g) = \sum_{l=1}^m \lambda_l (\mathbf{D}_x^{D-1} f_l) = \sum_{l=1}^m \lambda_l C_{f_l}^{(D-1)}.$$

This matrix $C_g^{(D-1)}$ has rank 2 by Lemma 2, so all of the 3-minors (3×3 sub-determinants) of $C_g^{(D-1)} = \sum_{l=1}^m \lambda_l C_{f_l}^{(D-1)}$ vanish. This gives a system of cubic equations in $\lambda_1, \dots, \lambda_m$. For some equation systems and choices of m and n , this cubic system is easily soluble, for example by linearisation. This process is in essence the **GeometricXL** algorithm, and it uses polynomials of a much smaller degree for certain equation systems than either a Gröbner basis or **XL** algorithm [12].

It is a consequence of Lemma 2 that the performance of the **GeometricXL** algorithm as originally described in [12] depends greatly on the field characteristic. When the characteristic p of the field \mathbb{F} satisfies $p = 0$ or $p > D$, then identifying a partial derivatives matrix of rank at most 2 gives a rank-2 product polynomial (Lemma 2) and so gives information about the solution to the original system. However, when the characteristic p satisfies $0 < p \leq D$, then a partial derivatives matrix of rank at most 2 may or may not give a rank-2 product polynomial. In particular, the **GeometricXL** algorithm of [12] is not well-suited for fields of even characteristic.

2.2 An Alternative Description of the GeometricXL Algorithm

The **GeometricXL** algorithm as described in [12] requires the use of $(D - 1)^{th}$ -order partial derivatives matrices in order to solve a homogeneous multivariate polynomial system of degree D . However, we now give an equivalent description of the **GeometricXL** algorithm in terms of first order partial derivatives matrices when the field characteristic is either zero or it exceeds the degree D .

Definition 2. Let h be a homogeneous polynomial of degree D in the polynomial ring $\mathbb{F}[x_0, \dots, x_n]$, where the field has characteristic p and either $p > D$ or $p = 0$. Furthermore let $\mathbf{x} = x_0^{e_0} \dots x_n^{e_n}$ denote a monomial of degree k ($0 \leq k \leq D$), so $e_0 + \dots + e_n = k$. The k^{th} -order *catalecticant matrix* [11] of h is

$$\tilde{C}_h^{(k)} = \left(\frac{1}{e_0! \dots e_n!} \cdot \frac{\partial^k h}{\partial x_0^{e_0} \dots \partial x_n^{e_n}} \right) = \left(\frac{1}{e_0! \dots e_n!} \mathbf{D}_x h \right).$$

Lemma 3. The catalecticant matrix $\tilde{C}_h^{(k)}$ satisfies $(\tilde{C}_g^{(k)})^T = \tilde{C}_g^{(d-k)}$ [11], and the catalecticant and partial derivatives matrices of order k , $\tilde{C}_h^{(k)}$ and $C_h^{(k)}$, share the same row space and have the same rank, with in particular $\tilde{C}_h^{(1)} = C_h^{(1)}$.

The **GeometricXL** algorithm tries to find a rank-2 product polynomial g . Lemma 3 shows that the column space of first order partial derivatives

matrix $C_g^{(1)}$ is identical to the row space of the $(D-1)^{th}$ order partial derivatives matrix $C_g^{(D-1)}$, so giving Lemma 4.

Lemma 4. If the characteristic of \mathbb{F} is either zero or exceeds D , then a multivariate polynomial g over \mathbb{F} of degree D is a rank-2 product polynomial, that is $g \in \mathcal{R}_{\mathbb{F},n}^D$, if and only if its first partial derivatives matrix $C_g^{(1)}$ has rank 2.

Lemma 4 means that the **GeometricXL** algorithm could be carried out by using the cubic system derived from the first order partial derivatives matrix rather than the $(D-1)^{th}$ order partial derivatives matrix. Furthermore, a basis for the column space of $C_g^{(1)}$ gives a pair of homogeneous linear polynomials L and L' of use in the product form of g . We give an example of an application of this alternative **GeometricXL** algorithm in Appendix A.

3 A Rank-2 Product Polynomial in Even Characteristic

We now suppose in this and subsequent sections that the field \mathbb{F} is of even characteristic, and we wish to find solutions to $f_1 = \dots = f_m = 0$, where f_1, \dots, f_m are homogeneous polynomials of degree D in $\mathbb{F}[x_1, \dots, x_m]$. For a **GeometricXL** algorithm in even characteristic, we wish to find a linear combination of f_1, \dots, f_m that is a rank-2 product polynomial, that is we wish to find g such that

$$g = \sum_{l=1}^m \lambda_l f_l = \prod_{i=1}^D (\theta_i L + \theta'_i L') = \sum_{i=0}^D \alpha_i^2 L^i (L')^{D-i},$$

for some $\alpha_i \in \mathbb{F}$ (as \mathbb{F} has even characteristic), $\theta_i, \theta'_i \in \overline{\mathbb{F}}$ (some extension field of \mathbb{F}) and homogeneous linear polynomials $L, L' \in \mathbb{F}[x_0, \dots, x_n]$. These homogeneous linear polynomials L and L' can be written as

$$L = \sum_{j=0}^n a_j x_j \text{ and } L' = \sum_{j=0}^n a'_j x_j.$$

The alternative description of the **GeometricXL** algorithm given in Section 2.2 finds a rank-2 product polynomial by finding a first order partial derivatives matrix of rank 2 in the case that the field characteristic p satisfies $p = 0$ or $p > D$. However, this property still holds in even characteristic, though the converse is not true, as the irreducible polynomial $x_0^2 + x_1 x_2$ over GF(2) with a partial derivatives matrix of rank 2 demonstrates.

We now discuss the properties of the partial derivatives matrix of a rank-2 product polynomial in even characteristic. We let W_D denote the vector space of homogeneous polynomials over \mathbb{F} of degree D in $n+1$ variables, so W_D has dimension $\binom{n+D}{D}$ [9]. In the terminology of [12], W_D is $\mathbb{S}^D(V^*)$, the D^{th} symmetric power of the dual space V^* of the standard vector space V of dimension $n+1$ over \mathbb{F} . We now define certain subspaces of W_D that we consider in the development of a **GeometricXL** algorithm for even characteristic.

Definition 3. The $\mathcal{L}^2\mathcal{S}$ -subspace of W_D for even degree $D = 2s + 2$ is the subspace $U_s = \langle x_i x_j \mathbf{x}^2 | \mathbf{x} \in W_s \rangle < W_{2s+2}$.

Definition 4. The $\mathcal{L}^1\mathcal{S}$ -subspace of W_D for odd degree $D = 2s + 1$ is the subspace $U'_s = \langle x_i \mathbf{x}^2 | \mathbf{x} \in W_s \rangle < W_{2s+1}$.

Definition 5. The $\mathcal{L}^0\mathcal{S}$ -subspace of W_D for even degree $D = 2s$ is the subspace $U''_s = \langle \mathbf{x}^2 | \mathbf{x} \in W_s \rangle < W_{2s}$.

Lemma 5. Any $\mathcal{L}^2\mathcal{S}$ -subspace U_s , $\mathcal{L}^1\mathcal{S}$ -subspace U'_s and $\mathcal{L}^0\mathcal{S}$ -subspace U''_s is invariant under collineation. The dimensions of these subspaces are given by:

$$\begin{aligned} \text{Dim}(U_s) &= \binom{n+1}{2} \binom{n+s}{s} + \binom{n+s+1}{s+1}, \\ \text{Dim}(U'_s) &= \binom{n+1}{1} \binom{n+s}{s}, \\ \text{Dim}(U''_s) &= \binom{n+s}{s}. \end{aligned}$$

The partial derivative mapping is a linear mapping, so any series of partial derivatives defines linear transformations $W_{2s+2} \rightarrow W_{2s+1} \rightarrow W_{2s} \rightarrow W_{2s-1}$. Thus any series of repeated partial differentiation in even characteristic gives rise to a series of linear transformations $U_s \rightarrow U'_s \rightarrow U''_s \rightarrow \{0\}$ or informally $\mathcal{L}^2\mathcal{S} \rightarrow \mathcal{L}^1\mathcal{S} \rightarrow \mathcal{L}^0\mathcal{S} \rightarrow 0$. Our analysis of rank-2 product polynomials in even characteristic now proceeds by considering even and odd degree polynomials as separate cases.

3.1 A Rank-2 Product Polynomial of Even Degree

We can write the even degree D as $D = 2s + 2$, so a rank-2 product polynomial $g \in \mathcal{R}_{\mathbb{F},n}^{2s+2}$ can be expressed as

$$g = \sum_{i=0}^{2s+2} \alpha_i^2 L^i (L')^{2s+2-i} = \sum_{i=0}^{s+1} \alpha_{2i}^2 L^{2i} (L')^{2(s+1-i)} + \sum_{i=0}^s \alpha_{2i+1}^2 L^{2i+1} (L')^{2s+1-2i}.$$

Thus we can express a rank-2 product polynomial g of even degree as

$$g = \left(\sum_{i=0}^{s+1} \alpha_{2i} L^i (L')^{(s+1-i)} \right)^2 + LL' \left(\sum_{i=0}^s \alpha_{2i+1} L^i (L')^{s-i} \right)^2.$$

We denote the first square of degree $2(s+1)$ by S^* and the second square of degree $2s$ by S , so g is given by

$$g = LL'S + S^*.$$

As \mathbb{F} has even characteristic, any partial derivative of S or S^* vanishes, so a partial derivative of $g \in \mathcal{R}_{\mathbb{F},n}^{2s+2}$ is given by

$$\frac{\partial g}{\partial x_l} = \frac{\partial (LL')}{\partial x_l} S = \frac{\partial L}{\partial x_l} L'S + \frac{\partial L'}{\partial x_l} LS = a_l(L'S) + a'_l(LS).$$

We have therefore shown that the partial derivative of a rank-2 product polynomial of even degree with respect to any variable is a linear combination of $L'S$ and LS . The above comments are summarised in Lemma 6.

Lemma 6. Let $g \in \mathbb{F}[x_0, \dots, x_n]$ be a rank-2 product polynomial of even degree $2s + 2$. If the field \mathbb{F} has even characteristic, then $g \in \mathcal{R}_{\mathbb{F},n}^{2s+2}$ has the following properties:

1. $g \in U_s$, the $\mathcal{L}^2\mathcal{S}$ -subspace;
2. $\frac{\partial g}{\partial x_l} \in U'_s$, the $\mathcal{L}^1\mathcal{S}$ -subspace;
3. the partial derivatives matrix $C_g^{(1)}$ has rank at most 2.

3.2 A Rank-2 Product Polynomial of Odd Degree

We can write the odd degree D as $D = 2s + 1$, so a rank-2 product polynomial $g \in \mathcal{R}_{\mathbb{F},n}^{2s+1}$ can be expressed as

$$g = \sum_{i=0}^{2s+1} \alpha_i^2 L^i (L')^{2s+1-i} = \sum_{i=0}^s \alpha_{2i}^2 L^{2i} (L')^{2s+1-2i} + \sum_{i=0}^s \alpha_{2i+1}^2 L^{2i+1} (L')^{2s-2i}.$$

We can thus express a rank-2 product polynomial g of odd degree as

$$g = L' \left(\sum_{i=0}^s \alpha_{2i} L^i L'^{s-i} \right)^2 + L \left(\sum_{i=0}^s \alpha_{2i+1} L^i L'^{s-i} \right)^2$$

The above expression for g consists of two squares of degree $2s$, which we denote by S' and S respectively. We can thus express g as

$$g = LS + L'S'.$$

As S and S' are square polynomials over a field of even characteristic, any partial derivative of S or S' is zero. Thus a partial derivative of $g \in \mathcal{R}_{\mathbb{F},n}^{2s+1}$ is given by

$$\frac{\partial g}{\partial x_l} = \frac{\partial L}{\partial x_l} S + \frac{\partial L'}{\partial x_l} S' = a_l S + a'_l S'.$$

We have therefore shown that the partial derivative of a rank-2 product polynomial of odd degree with respect to any variable is a linear combination of S and S' . The above comments are summarised by Lemma 7.

Lemma 7. Let $g \in \mathbb{F}[x_0, \dots, x_n]$ be a rank-2 product polynomial of odd degree $2s + 1$. If the field \mathbb{F} has even characteristic, then $g \in \mathcal{R}_{\mathbb{F},n}^{2s+1}$ has the following properties:

1. $g \in U'_s$, the $\mathcal{L}^1\mathcal{S}$ -subspace,
2. $\frac{\partial g}{\partial x_l} \in U''_s$, that is any partial derivative of g is a square;
3. the partial derivatives matrix $C_g^{(1)}$ has rank at most 2.

3.3 A Necessary Criterion for a Rank-2 Product Polynomial

Definition 6. The \mathcal{LS} -criterion for a homogeneous multivariate polynomial g is that g is an element either of the $\mathcal{L}^2\mathcal{S}$ -subspace (even degree) or the $\mathcal{L}^1\mathcal{S}$ -subspace (odd degree).

Lemmas 6 and 7 show that for a polynomial $g = \sum_{l=1}^m \lambda_l f_l$ to be a rank-2 product polynomial, g has to satisfy the \mathcal{LS} -criterion. For equation systems of cubic or higher degree, Lemma 5 shows that the dimension of the $\mathcal{L}^2\mathcal{S}$ -subspace or the $\mathcal{L}^1\mathcal{S}$ -subspace is generally far smaller than the dimension of W_D . For such equation systems, we can therefore obtain many linear constraints on $\lambda_1, \dots, \lambda_m$ for $g = \sum_{l=1}^m \lambda_l f_l$ to be a rank-2 product polynomial. These linear constraints can be processed very efficiently using basic linear algebra. Thus this criterion alone can easily greatly reduce the size of or even solve the equation system.

We give an example of solving a cubic system over a field of even characteristic by considering membership of the $\mathcal{L}^1\mathcal{S}$ -subspace U'_1 in Appendix B. However, both the $\mathcal{L}^1\mathcal{S}$ -subspace and $\mathcal{L}^2\mathcal{S}$ -subspace contain many polynomials that are not rank-2 product polynomials, so there may be a requirement for further processing after this preliminary linear filtering. Furthermore, this criterion cannot be applied to quadratic systems as $U_0 = W_2$. We discuss further techniques to identify rank-2 product polynomials in Section 4.

4 Identification of a Rank-2 Product Polynomial

The basic idea of the **GeometricXL** algorithm to solve the homogeneous system $f_1 = \dots = f_m = 0$ of degree D in $n + 1$ variables is to find a linear combination such that $g = \sum_{l=1}^m \lambda_l f_l$ is a rank-2 product polynomial, that is $g \in \mathcal{R}_{\mathbb{F},n}^D$. However, in even characteristic we can use the properties of rank-2 product polynomials in even characteristic given by Lemma 6 and Lemma 7 to help identify such polynomials. This should enable us subsequently to develop a method for fields of even characteristic based on the **GeometricXL** algorithm. However, we note that such an algorithm still has the potential problem discussed in Section 7.5 of [12], namely the possibility of nested multiple linear factors only one of which corresponds to a solution.

4.1 Multivariate Quadratic Systems

We consider a field \mathbb{F} of even characteristic and a homogeneous quadratic equation system $f_1 = \dots = f_m = 0$ over \mathbb{F} . We need to find a linear combination $g = \sum_{l=1}^m \lambda_l f_l$ such that g is a rank-2 product polynomial of degree 2. However, any such $g \in \mathcal{R}_{\mathbb{F},n}^2$ can be regarded as the product of the two homogeneous linear polynomials L and L' (Section 3.1), that is g is of the form

$$(a_0 x_0 + \dots + a_0 x_0)(a'_0 x_0 + \dots + a'_n x_n) = \sum_{i=0}^n a_i a'_i x_i^2 + \sum_{i=1}^n \sum_{j=0}^{i-1} (a_i a'_j + a'_i a_j) x_i x_j.$$

We can write $\Delta_{ij} = a_i a'_j + a_j a'_i$, so the product of two homogeneous linear polynomials can be expressed as

$$g = LL' = (a_0 x_0 + \dots + a_n x_n)(a'_0 x_0 + \dots + a'_n x_n) = \sum_{i=0}^n a_i a'_i x_i^2 + \sum_{i=1}^n \sum_{j=0}^{i-1} \Delta_{ij} x_i x_j.$$

We can write the homogeneous quadratic polynomial f_l ($1 \leq l \leq m$) as

$$f_l = \sum_{i=0}^n d_{ii}^{(l)} x_i^2 + \sum_{i=1}^n \sum_{j=0}^{i-1} d_{ij}^{(l)} x_i x_j$$

for coefficients $d_{ij}^{(l)}$, so a rank-2 product polynomial $g = \sum_{l=1}^m \lambda_l f_l$ satisfies

$$\begin{aligned} g &= \sum_{i=0}^n \left(\sum_{l=1}^m \lambda_l d_{ii}^{(l)} \right) x_i^2 + \sum_{i=1}^n \sum_{j=0}^{i-1} \left(\sum_{l=1}^m \lambda_l d_{ij}^{(l)} \right) x_i x_j \\ &= \sum_{i=0}^n a_i a'_i x_i^2 + \sum_{i=1}^n \sum_{j=0}^{i-1} \Delta_{ij} x_i x_j. \end{aligned}$$

Thus for g to be a rank-2 product polynomial, we can see by equating coefficients of $x_i x_j$ ($j < i$) that we require $\lambda_1, \dots, \lambda_m$ such that $\Delta_{ij} = \sum_{l=1}^m d_{ij}^{(l)} \lambda_l$.

Let A be the $2 \times (n+1)$ matrix with rows given by the (unknown) coefficients of the linear polynomials L and L' , so

$$A = \begin{pmatrix} a_0 & \dots & a_i & \dots & a_j & \dots & a_n \\ a'_0 & \dots & a'_i & \dots & a'_j & \dots & a'_n \end{pmatrix},$$

then the Δ_{ij} are the 2-minors (2×2 sub-determinants) of A . Now, there are $\binom{n+1}{4}$ 4-minors of the $4 \times (n+1)$ matrix $\bar{A} = \begin{pmatrix} A \\ A \end{pmatrix}$, and these are given by

$$\mathcal{A}_{i_1, i_2, i_3, i_4} = \Delta_{i_1, i_2} \Delta_{i_3, i_4} + \Delta_{i_1, i_3} \Delta_{i_2, i_4} + \Delta_{i_1, i_4} \Delta_{i_2, i_3}.$$

However, the matrix \bar{A} clearly has rank at most 2, so every 4-minor of \bar{A} is identically 0. Thus we obtain $\binom{n+1}{4}$ identities $\mathcal{A}_{i_1, i_2, i_3, i_4} = 0$. As each 4-minor $\mathcal{A}_{i_1, i_2, i_3, i_4}$ of \bar{A} gives rise to a homogeneous quadratic expression in Δ_{ij} , so each 4-minor identity $\mathcal{A}_{i_1, i_2, i_3, i_4} = 0$ gives a homogeneous quadratic identity in Δ_{ij} .

We saw above that for $g = \sum_{l=1}^m \lambda_l f_l$ to be a rank-2 product polynomial, we require that $\Delta_{ij} = \sum_{l=1}^m d_{ij}^{(l)} \lambda_l$. Thus the $\binom{n+1}{4}$ identities $\mathcal{A}_{i_1, i_2, i_3, i_4} = 0$ give rise to a homogeneous quadratic system with $\binom{n+1}{4} \sim \frac{1}{24} n^4$ equations satisfied by $\lambda_1, \dots, \lambda_m$. We can potentially solve this system using linearisation or some other simple technique. If there is a unique solution to this quadratic system in $\lambda_1, \dots, \lambda_m$, then this can directly determine the solution of the original equation system. More generally, an analysis of this quadratic system gives much information about the original quadratic system, which could then be used with other techniques to provide a solution to the original system.

We make some comments about geometric aspects of this process in Section 5.2, and demonstrate the use of this process in finding a solution to a quadratic system in Appendix C.

4.2 Multivariate Systems of Quartic or Higher Even Degree

We consider a field \mathbb{F} of even characteristic and a homogeneous equation system $f_1 = \dots = f_{m'} = 0$ of degree $D = 2s + 2$ ($s > 0$) over \mathbb{F} , where without loss of generality this equation system may have been obtained by applying the $\mathcal{L}^2\mathcal{S}$ criterion of Section 3.3 to a larger system $f_1 = \dots = f_m = 0$ (so $m' \leq m$). We need to find a linear combination $g = \sum_{l=1}^{m'} \lambda_l f_l$ such that g is a rank-2 product polynomial, that is $g \in \mathcal{R}_{\mathbb{F},n}^{2s+2}$. In the quadratic case ($s = 0$), we consider the coefficients of the non-square monomials $x_i x_j$ (so $i \neq j$) to give conditions for a rank-2 product polynomial (Section 4.1). In the case of quartic and higher even degree, we first consider the coefficients Γ_{ij} of $x_i^{2s+1} x_j$ (for $i \neq j$) in g , a natural generalisation of this idea.

If g is a rank-2 product polynomial of degree $2s + 2$, so $g \in \mathcal{R}_{\mathbb{F},n}^{2s+2}$, then $g = LL'S + S^*$, where L and L' are the homogeneous linear polynomials of Section 3 and S and S^* are homogeneous square polynomials of degree $2s$ and $2s + 2$ respectively (Section 3.1). Thus if we let s_i denote the coefficient of x_i^{2s} in S we have

$$\begin{aligned} g &= (a_i x_i + a_j x_j + \dots)(a'_i x_i + a'_j x_j + \dots)(s_i x_i^{2s} + \dots) + S^* \\ &= \Gamma_{ij} x_i^{2s+1} x_j + \dots = s_i \Delta_{ij} x_i^{2s+1} x_j + \dots = (b_i a'_j + b'_i a_j) x_i^{2s+1} x_j + \dots, \end{aligned}$$

where $\Delta_{ij} = a_i a'_j + a'_i a_j$ is a 2-minor of the matrix A of Section 4.1 and $b_i = s_i a_i$ and $b'_i = s_i a'_i$. Thus we have $\Gamma_{ij} = s_i \Delta_{ij} = b_i a'_j + b'_i a_j$ for $i \neq j$. For completeness, we set $\Gamma_{ii} = b_i a'_i + b'_i a_i = s_i a_i a'_i + s_i a_i a'_i = 0$. The matrix $\Gamma = (\Gamma_{ij})$ can be expressed as $\Gamma = B + B'$, where $B = \mathbf{b} \mathbf{a}'^T$ and $B' = \mathbf{b}' \mathbf{a}^T$ for appropriate column vectors of coefficients \mathbf{b} , \mathbf{a} , \mathbf{b}' and \mathbf{a}' , so B and B' are both matrices of rank 1. Thus the matrix Γ of coefficients of $x_i^{2s+1} x_j$ (for $i \neq j$ with $\Gamma_{ii} = 0$) of a rank-2 product polynomial has rank at most 2 as it is the sum $\Gamma = B + B'$ of two matrices of rank 1. There are $\binom{n+1}{3}^2$ 3-minors of Γ , and they are given by

$$\begin{aligned} \mathcal{B}_{i_1, i_2, i_3, j_1, j_2, j_3} &= \Gamma_{i_1, j_1} \Gamma_{i_2, j_2} \Gamma_{i_3, j_3} + \Gamma_{i_1, j_1} \Gamma_{i_2, j_3} \Gamma_{i_3, j_2} + \Gamma_{i_2, j_1} \Gamma_{i_1, j_2} \Gamma_{i_3, j_3} \\ &\quad + \Gamma_{i_2, j_1} \Gamma_{i_1, j_3} \Gamma_{i_3, j_2} + \Gamma_{i_3, j_1} \Gamma_{i_1, j_2} \Gamma_{i_2, j_3} + \Gamma_{i_3, j_1} \Gamma_{i_1, j_3} \Gamma_{i_2, j_2}. \end{aligned}$$

However, every 3-minor of the matrix Γ of rank 2 vanishes, so we obtain $\binom{n+1}{3}^2$ identities $\mathcal{B}_{i_1, i_2, i_3, j_1, j_2, j_3} = 0$.

For g to be a rank-2 product polynomial, we can see by equating coefficients of $x_i^{2s+1} x_j$ ($i \neq j$) that we require $\lambda_1, \dots, \lambda_{m'}$ such that $\Delta_{ij} = \sum_{l=1}^{m'} d_{ij}^{(l)} \lambda_l$, where $d_{ij}^{(l)}$ denote the coefficient of $x_i^{2s+1} x_j$ in f_l . Thus the $\binom{n+1}{3}^2$ identities $\mathcal{B}_{i_1, i_2, i_3, j_1, j_2, j_3} = 0$ give rise to a homogeneous cubic system with $\binom{n+1}{3}^2 \sim \frac{1}{36} n^6$ equations satisfied by $\lambda_1, \dots, \lambda_{m'}$. It may now be possible to solve this resulting cubic system in $\lambda_1, \dots, \lambda_{m'}$ by linearisation or some other method, so providing a solution to the original equation system. We provide an example of this process to solve a homogeneous quartic system in Appendix D.

We also note that it is very easy to produce further homogeneous cubic equations in $\lambda_1, \dots, \lambda_{m'}$ if required, as Γ is far from being the only matrix of coefficients having rank 2. For example, a similar argument to that given above for Γ shows that the matrix of coefficients of $x_0^2 x_i^{2s-1} x_j$ also has rank 2 and so on.

4.3 Multivariate Systems of Odd Degree

We consider a field \mathbb{F} of even characteristic and a homogeneous equation system $f_1 = \dots = f_{m'} = 0$ of odd degree $D = 2s + 1$ over \mathbb{F} , where without loss of generality this equation system may have been obtained by applying the $\mathcal{L}^1\mathcal{S}$ criterion of Section 3.3 to a larger system $f_1 = \dots = f_m = 0$ (so $m' \leq m$). We need to find a linear combination $g = \sum_{l=1}^{m'} \lambda_l f_l$ such that g is a rank-2 product polynomial, that is $g \in \mathcal{R}_{\mathbb{F},n}^{2s+1}$. In a similar way to the case for even degree, we consider the coefficients Λ_{ij} of $x_i^{2s}x_j$ in g , including the coefficients Λ_{ii} of x_i^{2s+1} .

If g is a rank-2 product polynomial of degree $2s+1$, so $g \in \mathcal{R}_{\mathbb{F},n}^{2s+1}$, then we can express g as $g = LS + L'S'$, where L and L' are homogeneous linear polynomials and S and S' are homogeneous square polynomials of degree $2s$ (Section 3.2). We can thus express a summand of g as

$$LS = c_{ij}x_i^{2s}x_j + \dots = (s_i x_i^{2s} + \dots)(a_j x_j + \dots) = s_i a_j x_i^{2s}x_j + \dots,$$

so the matrix $C = (c_{ij})$ has rank 1 as $C = \mathbf{s}\mathbf{a}^T$ for appropriate column vectors of coefficients \mathbf{s} and \mathbf{a} . This means we can express g as

$$g = \Lambda_{ij}x_i^{2s}x_j + \dots = LS + L'S' = (c_{ij} + c'_{ij})x_i^{2s}x_j + \dots.$$

Thus the matrix $\Lambda = (\Lambda_{ij})$ of coefficients of $x_i^{2s}x_j$ of a rank-2 product polynomial has rank at most 2 as it is the sum $\Lambda = C + C'$ of two matrices of rank 1. There are $\binom{n+1}{3}^2$ 3-minors of Λ , and they are given by

$$\begin{aligned} \mathcal{C}_{i_1, i_2, i_3, j_1, j_2, j_3} &= \Lambda_{i_1, j_1}\Lambda_{i_2, j_2}\Lambda_{i_3, j_3} + \Lambda_{i_1, j_1}\Lambda_{i_2, j_3}\Lambda_{i_3, j_2} + \Lambda_{i_2, j_1}\Lambda_{i_1, j_2}\Lambda_{i_3, j_3} \\ &\quad + \Lambda_{i_2, j_1}\Lambda_{i_1, j_3}\Lambda_{i_3, j_2} + \Lambda_{i_3, j_1}\Lambda_{i_1, j_2}\Lambda_{i_2, j_3} + \Lambda_{i_3, j_1}\Lambda_{i_1, j_3}\Lambda_{i_2, j_2}. \end{aligned}$$

However, every 3-minor of Λ vanishes as Λ has rank 2, so we obtain $\binom{n+1}{3}^2$ identities $\mathcal{C}_{i_1, i_2, i_3, j_1, j_2, j_3} = 0$.

For g to be a rank-2 product polynomial, we can see by equating coefficients of $x_i^{2s}x_j$ ($i \neq j$) that we require $\lambda_1, \dots, \lambda_{m'}$ such that $\Delta_{ij} = \sum_{l=1}^{m'} d_{ij}^{(l)} \lambda_l$, where $d_{ij}^{(l)}$ denote the coefficient of $x_i^{2s}x_j$ in f_l . Thus the $\binom{n+1}{3}^2$ identities $\mathcal{C}_{i_1, i_2, i_3, j_1, j_2, j_3} = 0$ give rise to a homogeneous cubic system with $\binom{n+1}{3}^2 \sim \frac{1}{36}n^6$ equations satisfied by $\lambda_1, \dots, \lambda_{m'}$. It may now be possible to solve this resulting cubic system in $\lambda_1, \dots, \lambda_{m'}$ by linearisation or some other method, so providing a solution to the original equation system. We provide an example of this process to solve a quintic system in Appendix E. Furthermore, as in Section 4.2, we note that is very easy to produce further homogeneous cubic equations in $\lambda_1, \dots, \lambda_{m'}$ if required as there are many other coefficient matrices having rank 2, for example the matrix of coefficients of $x_0^2 x_i^{2s-2} x_j$.

5 A Geometrical Interpretation

The geometric techniques of Section 4 can be interpreted using the Grassmannian variety [3,9,10]. We give some basic properties of the Grassmannian variety in Section 5.1 and discuss their application in Section 5.2.

5.1 The Grassmannian Variety and Exterior Algebra

We recall that W_1 denotes the vector space of homogeneous linear polynomials over \mathbb{F} of degree D in $n+1$ variables (Section 3). Thus the two linear polynomials L and L' at the heart of Definition 1 are elements of W_1 , so the pair (L, L') defines a (projective) line in the projective space $\mathcal{P}(W_1)$. The *Grassmannian* $Gr_2(W_1)$ is the set of all 2-dimensional subspaces of W_1 or equivalently the set of all projective lines in this projective space $\mathcal{P}(W_1)$ [9,10].

The tensor product $W_1 \otimes W_1$ is the $(n+1)^2$ -dimensional vector space with basis vectors the set of formal symbols $x_i \otimes x_j$ and a bilinear inclusion map from $W_1 \times W_1$ to $W_1 \otimes W_1$ [4,12]. The symmetric square $\mathbb{S}^2(W_1)$ is the subspace of all symmetric tensors $(\{t_{ij} \in W_1 \otimes W_1 \mid t_{ij} = t_{ji}\})$, a subspace of dimension $\frac{1}{2}(n+1)(n+2)$ [9,12]. In even characteristic, the symmetric square can be decomposed as

$$\mathbb{S}^2(W_1) \cong \langle (x_i \otimes x_i)_i \rangle \oplus \frac{\mathbb{S}^2(W_1)}{\langle (x_i \otimes x_i)_i \rangle},$$

where the second (quotient) summand is the degree-2 part of the *exterior algebra*, a space of dimension $\frac{1}{2}n(n+1)$, which we denote by $\mathbb{E}^2(W_1)$. Thus we have $\mathbb{S}^2(W_1) \cong \langle (x_i \otimes x_i)_i \rangle \oplus \mathbb{E}^2(W_1)$. The Grassmannian or *Plücker embedding* of the Grassmannian $Gr_2(W_1)$ in the degree-2 part of the exterior algebra is an injective mapping $\psi: Gr_2(W_1) \rightarrow \mathcal{P}(\mathbb{E}^2(W_1))$ defined by $(L, L') \mapsto L \wedge L'$ or equivalently

$$(L, L') = \left(\sum_{i=0}^n a_i x_i, \sum_{i=0}^n a'_i x_i \right) \mapsto \sum_{i=0}^n \sum_{j=0}^{i-1} (a_i a'_j + a'_i a_j) (x_i \wedge x_j).$$

This vector of co-ordinates $(a_i a_j + a'_i a'_j)$ is known as the Grassmannian or *Plücker co-ordinates* of the (projective) line defined by L and L' . Thus the Grassmannian co-ordinates of the line defined by L and L' are given by the 2-minors of the matrix A of Section 4.1, with rows given by L and L' .

The Grassmannian embedding allows the representation of (projective) lines in $\mathcal{P}(W_1)$ as distinct points in the projective space $\mathcal{P}(\mathbb{E}^2(W_1))$, which is isomorphic to $PG(\frac{1}{2}n(n+1)-1, \mathbb{F})$. The *Grassmannian variety* \mathcal{G} of $\mathcal{P}(\mathbb{E}^2(W_1))$ is the set of all such embedded lines [3,9]. Thus the *Grassmannian variety* \mathcal{G} is simply $\psi(Gr_2(W_1))$, the image of the Grassmannian $Gr_2(W_1)$, so \mathcal{G} is defined by

$$\mathcal{G} = \left\{ \left\langle (\Delta_{21}, \Delta_{31}, \dots, \Delta_{n+1,n-1}, \Delta_{n+1,n})^T \right\rangle \in \mathcal{P}(\mathbb{E}^2(W_1)) \mid \mathcal{A}_{i_1, i_2, i_3, i_4} = 0 \right\},$$

where $\mathcal{A}_{i_1, i_2, i_3, i_4}$ is the quadratic identity of Section 4.1. Thus the Grassmannian variety \mathcal{G} can be defined by the intersection of quadrics in $\mathcal{P}(\mathbb{E}^2(W_1))$.

5.2 The Grassmannian Variety and the GeometricXL Algorithm

We consider first a **GeometricXL** algorithm for homogeneous quadratic systems in even characteristic, discussed in Section 4.1. We need to find $g = \sum_{i=1}^m \lambda_i f_i$ such that g is the product of two homogeneous linear polynomials L and L' . We

can think of g in the natural and obvious way as an element of $\mathbb{S}^2(W_1)$. Thus we can define the canonical projection $\pi(g)$ of g onto the exterior algebra $\mathbb{E}^2(W_1)$, so $\pi(g)$ is the square-free part of g . However, g is a product of linear polynomials if and only if $\pi(g) \in \mathcal{G}$, the Grassmannian variety. Thus the geometrical interpretation of the process of Section 4.1 is that it is a process that first attempts to find polynomials with no square terms in the variety

$$\mathcal{G} \bigcap \langle \pi(f_1), \dots, \pi(f_m) \rangle,$$

and then analyses these polynomials to find rank-2 product polynomials.

There are some further obvious geometric comments that can be made about higher degree equations systems. For example, for odd degree systems any solution lies on the secant variety of the variety of all polynomials which are the product of a linear polynomial and a square polynomial, and such secant varieties are the basis of the **GeometricXL** algorithm when the characteristic exceeds D or is zero [12]. However, a full geometric interpretation is still needed for cubic and higher degree equation systems, and could provide interesting ideas for solution methods. Such an interpretation is very likely to depend on the Grassmannian variety \mathcal{G} as any rank-2 product polynomial depends fundamentally on the linear polynomials L and L' , or equivalently such an interpretation depends fundamentally on a point of the Grassmannian variety \mathcal{G} .

6 The EGHAM Process

In the common cryptographic situation of attempting to find a unique solution to a homogeneous equation system $f_1 = \dots = f_m = 0$ over a field of even characteristic, the goal of the XL algorithm is to find a bivariate polynomial $g = \sum_{i=1}^m \lambda_i f_i$, and the goal of the Gröbner basis algorithm (under lexicographic ordering) is to find such a bivariate polynomial g in the reduced Gröbner basis. However, we have shown that any such linear combination of homogeneous polynomials generating such a bivariate polynomial g must satisfy the \mathcal{LS} -criterion. Thus the \mathcal{LS} -criterion (Definition 6) can be applied as part of a standard Gröbner basis or XL algorithm in this situation. Applying the \mathcal{LS} -criterion generally greatly reduces the number of polynomials under consideration by an XL algorithm or Gröbner basis algorithm. Furthermore, the reduced set of equations obtained by applying the \mathcal{LS} -criterion can be considered for any order of the variables, as the \mathcal{LS} -criterion is independent of the order of the variables. This means that the XL algorithm or Gröbner basis algorithm with the \mathcal{LS} -criterion is far more efficient at finding an appropriate bivariate polynomial or concluding that no such bivariate polynomial exists for a given degree (under any variable order).

We can therefore include the \mathcal{LS} -criterion as part of an adaptation, using the ideas of Section 4, of the **GeometricXL** algorithm [12] for use in fields of even characteristic. This gives us a process for implementing an even (characteristic) **GeometricXL** heuristic algorithmic method, which we term the **EGHAM** process and which we describe in Figure 1.

- Generate a homogeneous system $f_1 = \dots = f_m = 0$ of degree D from an original equation system in even characteristic.
- Apply the \mathcal{LS} -criterion (Definition 6), that is consider only linear combinations of f_1, \dots, f_m in the $\mathcal{L}^2\mathcal{S}$ -subspace or $\mathcal{L}^1\mathcal{S}$ -subspace for systems of cubic or higher degrees, so generally reducing the size of the problem from m equations to m' equations.
- Construct a quadratic or cubic equation system in the coefficients $\lambda_1, \dots, \lambda_{m'}$ for $\sum_{i=1}^{m'} \lambda_i f_i$ to be a rank-2 product polynomial, for example by considering the matrix of coefficients of $x_i^{D-1} x_j$ (Section 4).
- Analyse this quadratic or cubic system, perhaps by linearisation, to find a rank-2 product polynomial in $\langle f_1, \dots, f_{m'} \rangle$, which can then be factored to provide information about the solution of (or even solve) $f_1 = \dots = f_{m'} = 0$.
- Apply this solution information directly to the original equation system.

Fig. 1. Basic Description of the EGHAM Process

The EGHAM process is an adaptation of the **GeometricXL** algorithm to fields of even characteristic, and so is a geometric generalisation of an **XL** algorithm for such fields. Thus the EGHAM process generates equations of at worst the same degree as either a Gröbner basis (with lexicographic ordering) or an **XL** algorithm, though usually by processing a far smaller equation system through the application of the \mathcal{LS} -criterion. However, as is clearly demonstrated by the examples in the Appendices, the EGHAM process can use polynomials of a much smaller degree for many equation systems in even characteristic than a Gröbner basis algorithm or an **XL** algorithm.

7 Conclusions

A major contribution of this paper is the development of the \mathcal{LS} -criterion (Definition 6) in the solution of homogeneous cryptographic equation systems in fields of even characteristic. We have used the \mathcal{LS} -criterion to develop the EGHAM process, an adaptation of the **GeometricXL** algorithm [12] which is suitable for use in fields of even characteristic. The EGHAM process can find the solution of a cryptographic equation system in even characteristic much more efficiently than a standard Gröbner basis or **XL** algorithm in many cases.

Acknowledgments

We wish to thank Martin Albrecht and our referees for their comments.

References

1. Ars, G., Faugère, J.-C., Imai, H., Kawazoe, M., Sugita, M.: Comparison between **XL** and Gröbner Basis Algorithms. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 338–353. Springer, Heidelberg (2004)

2. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD thesis, Universität Innsbruck (1965)
3. Burau, W.: Mehrdimensionale Projektive und Höhere Geometrie, Berlin (1961)
4. Cohn, P.: Classical Algebra. John Wiley, Chichester (2000)
5. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
6. Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139, 61–88 (1999)
7. Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). In: Mora, T. (ed.) International Symposium on Symbolic and Algebraic Computation – ISSAC 2002, pp. 75–83 (2002)
8. Faugère, J.-C., Levy-dit-Vehel, F., Perret, L.: Cryptanalysis of MinRank. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 280–296. Springer, Heidelberg (2008)
9. Harris, J.: Algebraic Geometry: A First Course. Graduate Text in Mathematics, vol. 133. Springer, Heidelberg (1992)
10. Hirschfeld, J.W.P., Thas, J.A.: General Galois Geometries. Oxford University Press, Oxford (1991)
11. Iarrobino, A., Kanev, V.: Power Sums, Gorenstein Algebras and Determinantal Loci. Lecture Notes in Mathematics, vol. 1725. Springer, Heidelberg (1999)
12. Murphy, S., Paterson, M.B.: A Geometric View of Cryptographic Equation Solving. Journal of Mathematical Cryptology 2, 63–107 (2008)

A The Alternative GeometricXL Algorithm

We demonstrate the alternative **GeometricXL** algorithm (Section 2.2) on a homogeneous cubic system $f_1 = f_2 = f_3 = 0$ of three equations in three variables over GF(37). We give the coefficients of f_1, f_2, f_3 below with respect to the lexicographic monomial ordering $x_0^3, x_0^2x_1, \dots, x_2^3$.

$$\begin{array}{ccccccccc} 23 & 27 & 15 & 25 & 11 & 24 & 26 & 7 & 21 & 36 \\ 21 & 35 & 2 & 18 & 4 & 1 & 29 & 5 & 32 & 33 \\ 32 & 13 & 28 & 10 & 8 & 13 & 24 & 10 & 19 & 15 \end{array}$$

We need to find a linear combination $\lambda_1 C_{f_1}^{(1)} + \lambda_2 C_{f_2}^{(1)} + \lambda_3 C_{f_3}^{(1)}$ of these first order partial derivatives matrices $C_{f_1}^{(1)}, C_{f_2}^{(1)}$ and $C_{f_3}^{(1)}$ that has rank 2. Thus we need to find $\lambda_1, \lambda_2, \lambda_3$ such that

$$\begin{aligned} & \lambda_1 \begin{pmatrix} 32 & 17 & 30 & 25 & 11 & 24 \\ 27 & 13 & 11 & 4 & 14 & 21 \\ 15 & 11 & 11 & 7 & 5 & 34 \end{pmatrix} \\ & + \lambda_2 \begin{pmatrix} 26 & 33 & 4 & 18 & 4 & 1 \\ 35 & 36 & 4 & 13 & 10 & 32 \\ 2 & 4 & 2 & 5 & 27 & 25 \end{pmatrix} \\ & + \lambda_3 \begin{pmatrix} 22 & 26 & 19 & 10 & 8 & 13 \\ 13 & 20 & 8 & 35 & 20 & 19 \\ 28 & 8 & 26 & 10 & 1 & 8 \end{pmatrix} \end{aligned}$$

has rank 2. We can identify a matrix of rank 2 by considering its 3-minors, and, as an example, the first 3-minor of the above matrix is given by

$$34\lambda_1^3 + 22\lambda_1^2\lambda_2 + 30\lambda_1^2\lambda_3 + 19\lambda_1\lambda_2^2 + 9\lambda_1\lambda_2\lambda_3 + 9\lambda_1\lambda_3^2 + 22\lambda_2^3 + 20\lambda_2^2\lambda_3 + 27\lambda_2\lambda_3^2 + 5\lambda_3^3.$$

There are 15 3-minors of the above 3×6 matrix, and we require them all to vanish for this matrix to have rank 2. This gives a system of 15 homogeneous cubic equations in $\lambda_1, \lambda_2, \lambda_3$. However, there are only 10 cubic monomials in 3 variables, so we can solve this system by direct linearisation to obtain $\lambda_1 = \lambda_3$ and $\lambda_2 = 26\lambda_3$ as the unique solution. We thus consider the polynomial $g = f_1 + 26f_2 + f_3$ which has a vector of coefficients given by $(9, 25, 21, 22, 12, 26, 27, 36, 21, 21)$ (with respect to lexicographic ordering) and partial derivatives matrix of rank 2

$$C_g^{(1)} = C_{f_1}^{(1)} + 26C_{f_2}^{(1)} + C_{f_3}^{(1)} = \begin{pmatrix} 27 & 13 & 5 & 22 & 12 & 26 \\ 25 & 7 & 12 & 7 & 35 & 21 \\ 21 & 12 & 15 & 36 & 5 & 26 \end{pmatrix}.$$

We can now either factor g directly or by noting that any factor of g is a linear combination (possibly over an extension field) of $27x_0 + 25x_1 + 21x_2$ and $13x_0 + 7x_1 + 12x_2$, which are given by a basis for the column space of $C_g^{(1)}$. Thus we obtain a factorisation over GF(37) of a linear combination of f_1, f_2, f_3 as

$$f_1 + 26f_2 + f_3 = 9(x_0 + 32x_1 + 3x_2)(x_0^2 + 16x_0x_1 + 24x_0x_2 + 29x_1^2 + 24x_1x_2 + 9x_2^2)$$

For a solution in GF(37), we obtain $x_0 = -(32x_1 + 3x_2)$, which on substitution into f_1 gives (over GF(37))

$$0 = x_1^3 + x_1^2x_2 + 9x_1x_2^2 + 33x_2^3 = (x_1 + 24x_2)(x_1^2 + 14x_1x_2 + 6x_2^2).$$

Thus over GF(37) we obtain $x_1 = -24x_2 = 13x_2$, so $x_0 = -(32.13+3)x_2 = 25x_2$. This gives $x_1 = 2x_0$ and $x_2 = 3x_0$, so the solution to the homogeneous cubic system $f_1 = f_2 = f_3 = 0$ is given by $(x_1, x_2, x_3) = \mu(1, 2, 3)$ for $\mu \in \text{GF}(37)$. For comparison, calculating this solution using a Gröbner basis or an XL algorithm requires the generation of polynomials of degree 6.

B The \mathcal{LS} -Criterion

We let $\mathbb{F} = \text{GF}(2)(\theta)$, where $\theta^4 + \theta + 1 = 0$, be a field with 2^4 elements, and we represent elements of this field in hexadecimal, so, for example, \mathbf{C} denotes $\theta^3 + \theta^2$. We consider the solution of the homogeneous cubic system $f_1 = f_2 = f_3 = 0$ over \mathbb{F} by applying the \mathcal{LS} -criterion (Definition 6), where f_1, f_2 and f_3 are given below.

$$\begin{aligned} & 6x_0^3 + 7x_0^2x_1 + 9x_0^2x_2 + 2x_0x_1^2 + \mathbf{C}x_0x_1x_2 + 5x_0x_2^2 + \mathbf{F}x_1^3 + 0x_1^2x_2 + \mathbf{D}x_1x_2^2 + 6x_2^3 \\ & \mathbf{B}x_0^3 + \mathbf{B}x_0^2x_1 + \mathbf{D}x_0^2x_2 + 0x_0x_1^2 + 8x_0x_1x_2 + 4x_0x_2^2 + 5x_1^3 + 6x_1^2x_2 + 3x_1x_2^2 + \mathbf{A}x_2^3 \\ & \mathbf{E}x_0^3 + 9x_0^2x_1 + \mathbf{D}x_0^2x_2 + \mathbf{D}x_0x_1^2 + 1x_0x_1x_2 + \mathbf{A}x_0x_2^2 + 1x_1^3 + 3x_1^2x_2 + \mathbf{C}x_1x_2^2 + 5x_2^3 \end{aligned}$$

We wish to find a linear combination $g = \lambda_1f_1 + \lambda_2f_2 + \lambda_3f_3 \in \mathcal{R}_{\mathbb{F},n}^3$. The \mathcal{LS} -criterion for a cubic system is the $\mathcal{L}^1\mathcal{S}$ condition (Lemma 7 Part 1), that is

$g \in U'_1 < W_3$. However, a polynomial in three variables in W_3 is in U'_1 if and only if the coefficient of $x_0x_1x_2$ is zero, so we need to find $\lambda_1, \lambda_2, \lambda_3$ such that $C\lambda_1 + 8\lambda_2 + 1\lambda_3 = 0$. Thus we have $(\lambda_1, \lambda_2, \lambda_3)^T \in \langle (1, 0, C)^T, (0, 1, 8)^T \rangle$. We can now define $f'_1 = f_1 + Cf_3$ and $f'_2 = f_2 + 8f_3$, and find a linear combination of f'_1 and f'_2 which factors either by direct search or by some other technique. We thus obtain

$$f'_1 + 9f'_2 = f_1 + 9f_2 + 8f_3 = 3(x_0 + x_1 + 2x_2)(x_0^2 + 4x_0x_1 + 9x_0x_2 + 4x_1^2 + Ax_1x_2 + Bx_2^2).$$

This gives $x_0 = x_1 + 2x_2$ as the unique solution over \mathbb{F} , and upon substitution into f_1, f_2 and f_3 , we obtain the following two linearly independent equations:

$$\begin{aligned} Cx_1^3 + 6x_1^2x_2 + Ex_1x_2^2 + 7x_2^3 &= C(x_1 + 7x_2)^2(x_1 + 9x_2); \\ 5x_1^3 + 5x_1^2x_2 + 1x_1x_2^2 + 0x_2^3 &= 5x_1(x_1 + 5x_2)(x_1 + 9x_2). \end{aligned}$$

We thus deduce that $x_1 = 9x_2$ and $x_0 = (9 + 2)x_2 = Bx_2$, so the solution to $f_1 = f_2 = f_3 = 0$ is given by $(x_0, x_1, x_2) = \mu(4, 2, 1)$ for $\mu \in \mathbb{F}$. For comparison, calculating this solution using a Gröbner basis or an XL algorithm requires the generation of polynomials of degree 6.

C A Quadratic System in Even Characteristic

We use the field $\mathbb{F} \cong \text{GF}(2^4)$ of Appendix B. We consider the solution of the homogeneous quadratic system $f_1 = \dots = f_7 = 0$ using the method of Section 4.1, where $f_1, \dots, f_7 \in \mathbb{F}[x_0, \dots, x_6]$. The coefficients of f_1, \dots, f_7 are given with respect to the lexicographic ordering $x_0^2, x_0x_1, \dots, x_6^2$ by the array below.

$$\begin{array}{ccccccccccccc} 2 & 3 & B & A & 9 & D & 3 & F & C & F & B & 4 & 5 & 3 & A & 0 & E & 6 & 6 & D & C & 9 & F & 5 & E & D & 2 & E \\ 7 & B & F & E & 2 & 8 & 6 & D & 5 & 7 & 3 & 5 & E & 4 & 3 & 3 & E & 3 & D & 1 & 6 & E & B & 4 & A & 5 & E & C \\ 3 & 2 & D & A & E & 0 & 9 & 4 & C & 4 & F & 5 & B & C & B & 3 & 9 & D & 2 & 0 & 6 & E & 3 & 5 & 1 & 6 & 8 & 6 \\ 3 & 6 & E & F & 2 & B & 2 & B & C & 2 & 2 & D & 9 & 3 & 3 & 5 & 9 & 5 & 6 & 8 & C & 6 & 3 & 8 & 6 & 8 & B & 8 \\ 6 & 1 & 0 & 7 & 0 & 4 & 6 & B & 7 & 4 & E & 7 & D & 3 & 6 & 8 & D & 9 & C & A & F & 9 & 4 & 1 & 3 & 6 & D & E \\ 8 & A & 0 & 4 & 0 & 2 & D & 1 & 6 & 8 & 4 & 0 & 0 & B & 7 & 1 & 2 & 6 & 8 & C & 3 & 9 & F & B & 5 & 8 & 4 & 2 \\ 8 & 0 & 8 & 7 & D & F & 0 & 4 & F & F & E & 9 & 2 & 5 & F & 0 & 2 & D & 4 & 9 & 6 & 6 & B & E & 0 & 5 & F & D \end{array}$$

We wish to find $\lambda_1, \dots, \lambda_7$ such that $g = \lambda_1f_1 + \dots + \lambda_7f_7 \in \mathcal{R}_{\mathbb{F},n}^2$, so g is given by

$$\begin{aligned} g &= (2\lambda_1 + 7\lambda_2 + 3\lambda_3 + 3\lambda_4 + 6\lambda_5 + 8\lambda_6 + 8\lambda_7)x_0^2 \\ &\quad + (3\lambda_1 + B\lambda_2 + 2\lambda_3 + 6\lambda_4 + 1\lambda_5 + A\lambda_6 + 0\lambda_7)x_0x_1 \\ &\quad + (B\lambda_1 + F\lambda_2 + D\lambda_3 + E\lambda_4 + 0\lambda_5 + 0\lambda_6 + 8\lambda_7)x_0x_2 + \dots \\ &= \Delta_{01}x_0x_1 + \Delta_{02}x_0x_2 + \dots \end{aligned}$$

We can now use the 2-minor identity (Section 4.1)

$$\mathcal{A}_{0,1,2,3} = \Delta_{01}\Delta_{23} + \Delta_{02}\Delta_{13} + \Delta_{03}\Delta_{12} = 0$$

to obtain a quadratic expression $Q(\lambda_1, \dots, \lambda_7) = 0$, where the coefficients of Q are given with respect to the lexicographic ordering $\lambda_1^2, \lambda_1\lambda_2, \dots, \lambda_7^2$ by the array

$$F \ 9 \ 4 \ 5 \ B \ A \ E \ 6 \ 3 \ 7 \ 1 \ D \ 9 \ 5 \ 4 \ 7 \ 2 \ 2 \ D \ 0 \ 4 \ A \ 0 \ 0 \ 4 \ 8 \ 8 \ A.$$

There are $\binom{7}{4} = 35$ such 2-minor identities giving rise to 35 quadratic expressions in $\lambda_1, \dots, \lambda_7$ in total. As there are only 28 quadratic monomials in 7 variables, we can express this quadratic system as a linear system in 28 variables using a 35×28 matrix. This matrix has rank 27 and reducing it to echelon form gives the unique (up to multiplication) solution

$$(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7) = (2, C, 7, 2, F, 2, 1).$$

We can therefore obtain a linear combination of f_1, \dots, f_7 which is a rank-2 product polynomial and so which factors to give

$$\begin{aligned} 0 &= 2f_1 + Cf_2 + 7f_3 + 2f_4 + Ff_5 + 2f_6 + f_7 \\ &= 6(x_0 + Ex_1 + 6x_2 + 6x_3 + Bx_4 + Fx_5 + 6x_6)(x_0 + 9x_1 + Dx_2 + Cx_3 + 2x_6). \end{aligned}$$

Thus we know that either $x_0 + Ex_1 + 6x_2 + 6x_3 + Bx_4 + Fx_5 + 6x_6 = 0$ or $x_0 + 9x_1 + Dx_2 + Cx_3 + 2x_6 = 0$. We can make these two substitutions to reduce the original problem in seven variables to one of two problems in six variables and so on. We thus find that the unique (projective) solution is $(1, 2, 4, 8, 3, 6, C)$. For comparison, calculating this solution using a Gröbner basis or an XL algorithm requires the generation of polynomials of degree 7.

D A Quartic System in Even Characteristic

We use the field $\mathbb{F} \cong GF(2^4)$ of Appendix B. We consider the solution of the homogeneous quartic system $f_1 = f_2 = f_3 = f_4 = f_5 = 0$ using the method of Section 4.2, where $f_1, \dots, f_5 \in \mathbb{F}[x_0, x_1, x_2, x_3, x_4]$ satisfy the \mathcal{LS} -Criterion (without loss of generality). The coefficients of f_1, \dots, f_5 are given below with respect to the lexicographic ordering $x_0^4, x_0^3x_1, \dots, x_4^4$.

```
49D32B5BD3D4AD69932C00A0A47A008C971A54E967A6950BB086D2FF0D95D8583B6103
2D68B5EC0F085974058900407984F0A59C80E7924EBA6B03A069D8B9E4DC2A2E7634EB
6E03B1E544DF3352E19D00C082A850CFED40169539B0259520FD730402F6C18FCBDC6D
475365A26C3E7F0CC9EC00E030E8301361A220DCAA9EC573D00D369E441F2A9701149E
D943065F4D8DB2F9629E00303CB9E0EDA6DF00E28151E25960567B58E1AB1441A76B8C
```

A rank-2 product polynomial in even degree is an element of the $\mathcal{L}^2\mathcal{S}$ -subspace (Definition 3), and we note without loss of generality that $f_1, \dots, f_5 \in U_1$, the $\mathcal{L}^2\mathcal{S}$ -subspace of W_4 .

A linear combination $g = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3 + \lambda_4 f_4 + \lambda_5 f_5$ is given by

$$\begin{aligned} g &= (4\lambda_1 + 2\lambda_2 + 6\lambda_3 + 4\lambda_4 + D\lambda_5)x_0^4 \\ &\quad + (9\lambda_1 + D\lambda_2 + E\lambda_3 + 7\lambda_4 + 9\lambda_5)x_0^3x_1 \\ &\quad + (D\lambda_1 + 6\lambda_2 + 0\lambda_3 + 5\lambda_4 + 4\lambda_5)x_0^3x_2 + \dots \\ &= \Gamma_{01}x_0^3x_1 + \Gamma_{02}x_0^3x_2 + \dots \end{aligned}$$

We can now use the cubic identities \mathcal{B} for g to be a rank-2 product polynomial to obtain $\binom{5}{3} \times \binom{5}{3} = 100$ homogeneous cubic equations in $\lambda_1, \dots, \lambda_5$ (Section 4.2). For example, the coefficients with respect to the lexicographic ordering $\lambda_1^3, \lambda_1^2\lambda_2, \dots, \lambda_5^3$ of the cubic expression given by $\mathcal{B}_{0,1,2,0,1,2}$ is given below.

09CF5757921C325CC9D45345AC71F6C7A21

Thus we obtain 100 cubic expressions in the 35 cubic monomials in the variables $\lambda_1, \dots, \lambda_5$, so we can express this cubic system as using a 100×35 matrix. This matrix has rank 34 and reducing it to echelon form gives the unique (projective) solution

$$(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5) = (5, 2, 9, B, 1).$$

Thus we construct the linear polynomial $g = 5f_1 + 2f_2 + 9f_3 + Bf_4 + 1f_5$, with coefficients given below, which is a rank-2 product polynomial.

18DABAA51A9A5379A68200304EB690F755CD63ED76419BCD705A2B95FBB4017D5E6587

We can indeed factor this linear combination of f_1, \dots, f_5 to obtain

$$0 = g = (x_0 + Dx_1 + 9x_2 + 5x_3 + Bx_4)^2 \times \text{Irreducible Quadratic}.$$

A solution over \mathbb{F} therefore satisfies $x_0 + Dx_1 + 9x_2 + 5x_3 + Bx_4 = 0$. We can thus make a substitution to reduce the original problem in five variables to a problem in four variables. We can continue using this techniques on the smaller system to give $(1, E, 4, 6, 7)$ as the unique (projective) solution to the original system. For comparison, calculating this solution using a Gröbner basis or an XL algorithm requires the generation of polynomials of degree 15.

E A Quintic System in Even Characteristic

We use the field $\mathbb{F} \cong \text{GF}(2^4)$ of Appendix B. We consider the solution of the homogeneous quintic system $f_1 = f_2 = f_3 = f_4 = f_5 = 0$ using the method of Section 4.2, where $f_1, \dots, f_5 \in \mathbb{F}[x_0, x_1, x_2, x_3, x_4]$ satisfy the \mathcal{LS} -Criterion (without loss of generality). The coefficients of f_1, \dots, f_5 are given below with respect to the lexicographic ordering $x_0^5, x_0^4x_1, \dots, x_4^5$.

```
63DAD00009001040B6DB008048BE3078F167000C007030000000000200C0300
0070103023E90040612D405CC4ED007070000E0B074B30010421B010BD32D7A
0A2B2D000A0060B25D010010178D50FE570000A00C03000000000F0010D00
006070909CAF0040552CD0EE87A900C03000C060EOF7407869930F0C078736
4BC09900030050E4933F00A0ECE25004F66700010080D000000000B00B0300
0070A0E1FDC100E07D8A2050625300D0D00090C0251130D947EF030FCF0B16
8A0D52000A00505FD51800B0324DC0200F39000B00909000000000700B0B00
0010FOAB1A4C0070DC90D0B8CBFA0090700040C0E00DD07C0E790F03EFEE84
2416BD00050060FCC4FA00D078973004AFF5000400304000000000200D0900
0010D09EAB24008004FB104FBA2E008050000C070664130C76D58080812CDDF
```

A rank-2 product polynomial in odd degree is an element of the $\mathcal{L}^1\mathcal{S}$ -subspace (Definition 4), and we note without loss of generality that $f_1, \dots, f_5 \in U'_2$, the $\mathcal{L}^1\mathcal{S}$ -subspace of W_5 .

A linear combination $g = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3 + \lambda_4 f_4 + \lambda_5 f_5$ is given by

$$\begin{aligned} g &= (6\lambda_1 + 0\lambda_2 + 4\lambda_3 + 8\lambda_4 + 2\lambda_5)x_0^5 \\ &\quad + (3\lambda_1 + A\lambda_2 + B\lambda_3 + A\lambda_4 + 4\lambda_5)x_0^4x_1 \\ &\quad + (D\lambda_1 + 2\lambda_2 + C\lambda_3 + 0\lambda_4 + 1\lambda_5)x_0^4x_2 + \dots \\ &= \Gamma_{00}x_0^5 + \Gamma_{01}x_0^4x_1 + \Gamma_{02}x_0^4x_2 + \dots \end{aligned}$$

We can now use the cubic identities \mathcal{C} for g to be a rank-2 product polynomial to obtain $\binom{5}{3} \times \binom{5}{3} = 100$ homogeneous cubic equations in $\lambda_1, \dots, \lambda_5$ (Section 4.2). For example, the coefficients with respect to the lexicographic ordering $\lambda_1^3, \lambda_1^2\lambda_2, \dots, \lambda_5^3$ of the cubic expression given by $\mathcal{C}_{0,1,2,0,1,2}$ is given below.

16785F34B6C6BAC6AF0B48FA8D2CD5BCADA

Thus we obtain 100 cubic expressions in the 35 cubic monomials in the variables $\lambda_1, \dots, \lambda_5$, so we can express this cubic system as using a 100×35 matrix. This matrix has rank 30 and reducing it to echelon form gives the following relations (amongst others):

$$0 = \lambda_1^2\lambda_5 + A\lambda_5^3 = \lambda_2^2\lambda_5 + 1\lambda_5^3 = \lambda_3^2\lambda_5 + E\lambda_5^3 = \lambda_4^2\lambda_5 + 7\lambda_5^3.$$

Thus analysing this equation system quickly shows that the unique (projective) solution is given by

$$(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5) = (F, 1, D, 6, 1).$$

Thus we construct the linear polynomial $g = Ff_1 + 1f_2 + Df_3 + 6f_4 + 1f_5$, with coefficients given below, which is a rank-2 product polynomial.

2379C3000C00E0BD4AA0090F13E608A2B98000400508000000000700D0D00
00F0A01CF25600E0CE1B40FCD25D00202000010F08351B0BE8E8C080A74B9D6

This linear combination of f_1, \dots, f_5 factors to give

$$0 = g = (x_0 + 8x_1 + Ax_2 + Dx_3 + 6x_4) \times (\text{Irreducible Quadratic})^2.$$

A solution over \mathbb{F} therefore satisfies $x_0 + 8x_1 + Ax_2 + Dx_3 + 6x_4 = 0$. We can thus make a substitution to reduce the original problem in five variables to a problem in four variables. We can continue using this techniques on the smaller system to give $(1, 4, 4, A, E)$ as the unique (projective) solution to the original system. For comparison, calculating this solution using a Gröbner basis or an XL algorithm requires the generation of polynomials of degree 20.

Provably Secure Code-Based Threshold Ring Signatures

Léonard Dallot¹ and Damien Vergnaud²

¹ GREYC – UMR 6072, Université de Caen
Campus Côte de Nacre, Boulevard du Maréchal Juin,
14050 Caen Cedex – France
`leonard.dallot@info.unican.fr`

² Ecole Normale Supérieure – C.N.R.S. – I.N.R.I.A.
45, Rue d’Ulm – 75230 Paris CEDEX 05 – France

Abstract. A threshold ring signature scheme enables a set of users to sign a message such that a finite set of possible signers (the ring) is identified, without revealing which subset of ring members actually produced the signature. A recent proposal of Aguillar *et al.* introduced the first code-based threshold ring signature scheme which large signatures (about 20KBytes per member of the ring for 80-bit security).

We propose a new code-based threshold ring signature scheme that achieves small signature size of $675N - 228\ell$ bits, where N is the number of members in the ring and ℓ is the number of signers, for a security level of 80 bits. We give a security proof of our scheme whose security relies — in both random oracle and ideal cipher models — on two coding theory problems, making our scheme the first provably secure code-based threshold ring signature scheme. Unfortunately, as often in code-based cryptography, the presented scheme leads to very large public keys.

1 Introduction

A ring signature enables a user to sign a message so that a finite set of possible signers (of which the user is a member) is identified, without revealing which member of that ring actually produced the signature. We propose a new code-based threshold ring signature scheme (*i.e.* where at least ℓ parties are required for creating a signature) that achieves small signature size. We analyse our scheme in the random oracle model and the ideal cipher model to prove that its security relies on two classical coding theory problems.

1.1 Ring Signatures

The concept of *group signatures* was first introduced by Chaum and van Heyst in 1991 [CvH92]; it is a method for allowing a member of a group to anonymously sign a message on behalf of the group. Essential to a group signature scheme is a *group manager*, who is in charge of adding group members and has the ability to reveal the original signer in the event of disputes. *Ring signatures* — first formalised in 2001 by Rivest, Shamir, and Tauman [RST01] — are

similar to group signatures but differ in two key ways: the anonymity of an individual signature cannot be removed, and any set of users can be used as a group without additional setup (*i.e.* the signer has to be a member of the ring but the other users do not need to cooperate and may be unaware that they are included in a ring signature), rings may be formed completely “on the fly”. Ring signature schemes have been studied extensively since 2001 (*e.g.* [BKM06, RST01, SW07, CGS07]) and a variety of applications have been suggested (*e.g.* [RST01, Nao02, DKNS04, AHR05]).

The original application was anonymous leaking of secrets. For example, a high-ranking official in the government wishes to leak some important information to a journalist. Ring signatures give a way to achieve this task, wherein the journalist can verify that some government official signed the message but cannot ascertain which member actually leaked the secret: the journalist — who knows the public keys of all the ministers — can be sure that one of the ministers signed it, without knowing who is the mole in the cabinet. Another popular application is designated verifier signatures [JSI96]: by signing a message with respect to a ring that contains only the sender and the receiver, the sender can ensure the receiver of the origin of the message, but the later is unable to transmit his conviction to anyone else.

In 2002, Bresson, Stern and Szydlo extended the concept of ring signatures to *threshold* ring signatures [BSS02]. In this context, a group of ℓ users wants to cooperate to produce a signature without revealing their identities among the ring. Simply produce ℓ ring signature clearly not reaches this goal since a receiver cannot ensure that the ℓ signers are different.

1.2 Code-Based Cryptography

Most popular public key cryptographic schemes rely either on the integer factorization or on the discrete logarithm problem. As they are used in most nowadays’s applications of public key cryptography, the later are thus vulnerable to a single breakthrough in algorithms or in hardware (for example, a quantum computer can break all those schemes). Diversity is a way to dilute that risk, and it is the duty of the cryptographic research community to prepare and propose alternatives to the number theory based systems. The most serious tracks today are euclidean lattices, multivariate cryptography and *code-based cryptography*.

Code-based cryptography was introduced by McEliece [McE78], two years after the introduction of public key cryptography by Diffie and Hellman [DH76] in 1976. In 1989, Stern proposed a code-based zero knowledge identification scheme [Ste90, Ste96] but the first native practical code-based signature scheme was proposed in 2001 by Courtois, Finiasz and Sendrier [CFS01]. Recent years saw a renewed interest in code-based cryptography with the proposal of several cryptographic primitives based on coding theory (*e.g.* a NIST hash function proposal [AFG⁺08], a provably secure encryption scheme [KI01], special signatures [AMCG08]).

In [ZLC07], Zheng, Li and Chen proposed the first code-based ring signature scheme and achieved a small signature size ($144 + 126N$ bits where N is the

size of the ring for a security level of now about $2^{63.3}$ [FS09]). A recent proposal of Aguillar, Cayrel and Gaborit introduced the first threshold ring signature scheme [AMCG08]. This scheme uses public keys of small size (347 bits for a security level of 2^{80}) but produces large signatures (about 20KBytes per member of the ring).

1.3 Contributions of the Paper

Short Code-based Threshold Ring Signature. We propose a new code-based threshold ring signature scheme that combines both techniques used by Bresson *et al.* in [BSS02] and Courtois *et al.* in [CFS01]. It achieves small signature size of $414N - 144\ell$ and $579N - 198\ell$ bits for respective security levels of $2^{63.3}$ and $2^{81.7}$ where N is the number of members in the ring and ℓ is the number of signers but requires keys of large size (the public keys respectively need 1.2Mbytes and 99Mbytes to be stored).

Reductionist Security. In cryptography, a system has *reductionist security* (or provable security) — as opposed to *heuristic security* — if its security requirements can be stated formally in an adversarial model with clear assumptions that the adversary has access to the system as well as computational resources. Provable security is an important research area in cryptography: cryptographic primitives or protocols without a rigorous proof cannot be regarded as secure in practice. There are many schemes that were originally thought as secure being successfully cryptanalyzed, which clearly indicates the need of formal security assurance (*e.g.* [COV07, OTD08] for examples in code-based cryptography). With reductionist security, we are confident in using cryptographic applications to replace the traditional way in physical world. In this approach, the security of a cryptographic scheme is based on algorithmic problems that are supposed to be hard to solve. The scheme is secure as long as the underlying algorithmic problems are difficult.

Code-based cryptography typically uses two difficult problems from coding theory: the *Bounded Distance Decoding problem* (BDD) — which is \mathcal{NP} -hard [BMvT78] — and the *Goppa Code Distinguishing* — considered as difficult [Sen02]. Confidence into asymptotic difficulty of the Bounded Distance Decoding problem is strengthened by the recently proven [LM09] equivalence between a lattice version of BDD and the two central problems used in lattice based public key cryptography, namely the unique Shortest Vector Problem (uSVP) and the decisional version GAP-SVP of the Shortest Vector Problem.

However, to obtain reductionist security of a scheme, it is sometimes necessary to work in an idealized model of computing — as opposed to the *standard model*. The *random oracle model* (ROM) formalized by Bellare and Rogaway [BR93] is one of those; in this model, a hash function is considered to be given by the public access to a random oracle, which output for any new queried message a random value uniformly distributed over the output space. Another model of computation, which has been proven to be equivalent to the ROM by Coron, Patarin and Seurin [CPS08], is the *ideal cipher model* (ICM): a publicly accessible random function is replaced by a public access in encryption and decryption

to a random block cipher (the ideal cipher). A proof in one of those model does not offer full satisfaction since such a proof does not imply that the scheme will remain secure when idealized functions will be replaced by concrete functions (e.g. [CGH04, LN09]). Despite of this, the ROM and ICM still are useful tools for proving the security of cryptosystem and, for some functionalities, these models provides the only known constructions.

In this paper, we present the first explicit reductionist security proof (in both the ideal cipher and random oracle models) of a code-based ring signature scheme, relating unforgeability of our scheme to the difficulty of the two problems above; while anonymity of the signers is unconditionally preserved.

2 Threshold Ring Signatures

2.1 Definition

We present a formal definition of an ℓ -out-of- N threshold ring signature scheme. We refer to a set of users $\mathcal{N} = \{1, \dots, N\}$ where each user u has a public and private keys pair (PK_u, SK_u) as a *ring*. For any set $\mathcal{L} \subset \mathcal{N}$, we denote by $\mathcal{SK}_{\mathcal{L}}$ the set $\{SK_{\ell_i} \mid i \in \mathcal{L}\}$ of the private keys of users in \mathcal{L} .

Definition 1 (ℓ -out-of- N threshold Ring Signature Scheme). An ℓ -out-of- N threshold ring signature scheme is defined by four (polynomial time) algorithms *Setup*, *Gen*, *Sign* and *Verify* that respectively sets up parameters, generates keys for a user, signs a message and verifies a signature:

- *Setup*(1^κ), where κ is a security parameter, outputs a set \mathcal{P} of parameters;
- *Gen*(\mathcal{P}), where \mathcal{P} is a set of parameters, outputs a pair of public and secret keys (PK, SK) ;
- *Sign*($\mathcal{P}, M, \mathcal{N}, \mathcal{SK}_{\mathcal{L}}$), where \mathcal{P} is a set of parameters and $\mathcal{L} \subset \mathcal{N}$ are two sets of users such that $|\mathcal{L}| = \ell$ and $|\mathcal{N}| = N$, signs a message M with respect to the public keys of the members of the ring \mathcal{N} ;
- *Verify*($\mathcal{P}, M, \sigma, \mathcal{N}$), where \mathcal{P} is a set of parameters, outputs valid if the proposed signature σ of a message M is valid with respect to the public keys of the members of the ring \mathcal{N} and outputs invalid otherwise;

such that for all security parameters $\kappa > 0$, for all parameters $\mathcal{P} \leftarrow \text{Setup}(1^\kappa)$, for all sets \mathcal{N} and \mathcal{L} such that $\mathcal{L} \subset \mathcal{N}$, $|\mathcal{L}| = \ell$, $|\mathcal{N}| = N$ and for all member $i \in \mathcal{N}$ in the ring $(PK_i, SK_i) \leftarrow \text{Gen}(\mathcal{P})$, we have:

$$\forall M \in \{0, 1\}^*, \forall \sigma \leftarrow \text{Sign}(\mathcal{P}, M, \mathcal{N}, \mathcal{SK}_{\mathcal{L}}), \text{Verify}(\mathcal{P}, M, \sigma, \mathcal{N}) = \text{valid}.$$

Remark 1. The later assertion, deemed *consistency*, formalizes that a properly formed signature has to be accepted as valid.

2.2 Security Model

Anonymity. The anonymity condition requires, informally, that an adversary should not be able to tell which members of a ring took part in the generation of a given signature, even if some of the private keys are corrupted.

More formally, we consider a simulation (*i.e.* a probabilist polynomial time Turing machine) $\text{Simulate}(\mathcal{P}, M, \mathcal{N}, \mathcal{L}, \mathcal{SK}_{\mathcal{N} \setminus \{i\}})$ where \mathcal{P} is a set of parameters, $i \in \mathcal{L}$, that outputs a bit-string which simulates a signature of a message M with respect to the public keys of the members of the ring \mathcal{N} . An ℓ -out-of- N threshold ring signature scheme is *anonymous* if for all security parameters $\kappa > 0$, for all parameters $\mathcal{P} \leftarrow \text{Setup}(1^\kappa)$, any ring \mathcal{N} such that for all member $i \in \mathcal{N}$ in the ring $(PK_i, SK_i) \leftarrow \text{Gen}(\mathcal{P})$, any subset $\mathcal{L} \subset \mathcal{N}$ of users, any message $M \in \{0, 1\}^*$ and any user $u \in \mathcal{L}$, there exists a random sequence such that

$$\text{Simulate}(\mathcal{P}, M, \mathcal{N}, \mathcal{L}, \mathcal{SK}_{\mathcal{N} \setminus \{u\}}) = \text{Sign}(\mathcal{P}, M, \mathcal{N}, \mathcal{SK}_{\mathcal{L}}).$$

Unforgeability. Unforgeability of a threshold signature scheme is defined by the following attack model: an ℓ -adversary \mathcal{A} , who obviously knows public keys of the members of a ring \mathcal{N} of size N , tries to output an ℓ -out-of- N valid ring signature σ^* for a message M^* of its choice, after having corrupted at most $\ell - 1$ user's private keys. He accesses a signature oracle Σ (q_Σ queries), a corruption oracle Υ and possibly a random oracle \mathcal{H} ($q_{\mathcal{H}}$ queries) and/or an ideal cipher oracle \mathcal{E} ($q_{\mathcal{E}}$ queries). \mathcal{A} can adaptively query the corruption oracle Υ to obtain private keys of some of the members of the ring. He also can query Σ which provides threshold ring signatures for message and signers chosen by \mathcal{A} ; obviously, \mathcal{A} is not allowed to output as signature σ^* an answer to one of the queries he made to Σ . We call success of an ℓ -adversary \mathcal{A} , denoted by $\text{Succ}_{\tau, q_\Sigma(q_{\mathcal{H}}, q_{\mathcal{E}})}^{CMA}(\mathcal{A})$ the probability that he outputs a valid forgery. An ℓ -out-of- N ring signature scheme is said to be t -CMA-secure if for any adversary \mathcal{A} , $\text{Succ}_{\tau, q_\Sigma(q_{\mathcal{H}}, q_{\mathcal{E}})}^{CMA}(\mathcal{A})$ is negligible.

2.3 Bresson, Stern and Szydlo's Generic Construction

In [BSS02] Bresson, Stern and Szydlo proposed a generic construction in the ideal cypher model for ℓ -out-of- N threshold ring signature scheme from a trapdoor one-way function. This scheme uses Lagrange's polynomial interpolation formula (Theorem 1), as proposed by Shamir in [Sha79] for secret sharing.

Theorem 1 (Lagrange Interpolation). *Let \mathbb{F} be a finite field and N pairs $(x_1, y_1), \dots, (x_N, y_N)$ in \mathbb{F}^2 such that all x_i are distinct. There exists a unique polynomial p of degree $N - 1$ in $\mathbb{F}[X]$ such that for all index i , $p(x_i) = y_i$. This polynomial is:*

$$p(x) = \sum_{i=1}^N y_i \prod_{j=1, j \neq i}^N \frac{x - x_j}{x_i - x_j}.$$

The ℓ signers of a message can cooperate to build an interpolation on $N - \ell$ values obtained by a one-way function and compute the ℓ missing values by using their secret keys. Let \mathcal{N} be the set of the N users in the ring, $f_{PK/SK} : \{0, 1\}^p \rightarrow \{0, 1\}^q$ be a one-way function, $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^q$ be a hash function and $(E_{k,i})$ be a family of random permutations that cipher messages of q bits with keys of length q_0 . For each user $i \in \mathcal{N}$, the generation algorithm generates a key

pair (SK_i, PK_i) for f according to the security parameter κ . In the following, $(\cdot \parallel \cdot)$ denotes the concatenation of two bit-strings and $\xleftarrow{R} \mathcal{S}$ denotes the random selection of a uniformly distributed value in the set \mathcal{S} . A set \mathcal{L} of ℓ users signs a message M as follows:

1. Set a key $k \leftarrow \mathcal{H}(M)$ and an initial value $y_0 \leftarrow \mathcal{H}(M \parallel \mathcal{N})$
2. For all user $i \in \mathcal{N} \setminus \mathcal{L}$ do
 - (a) Randomly pick $x_i \xleftarrow{R} \{0, 1\}^p$
 - (b) Compute $y_i \leftarrow f_{PK_i}(x_i)$
3. Compute an interpolation polynomial

$$P \in \mathbb{F}_{2^q}[X] \text{ such that } \begin{cases} d(P) = N - \ell \\ P(0) = y_0 \\ P(i) = E_{k,i}(y_i) \text{ for all } i \in \mathcal{N} \setminus \mathcal{L} \end{cases}$$

4. For all user $i \in \mathcal{L}$ do
 $x_i \leftarrow f_{SK_i}^{-1}(E_{k,i}^{-1}(P(i)))$
5. Output $(\mathcal{N}, x_1, \dots, x_N, P)$ as a ring signature of the message M

The signature verification consists in verifying that $P(0) = \mathcal{H}(M \parallel \mathcal{N})$ and the equality $P(i) = E_{\mathcal{H}(M), i}(f_{PK_i}(x_i))$ for all user $i \in \mathcal{N}$.

3 Courtois, Finiasz and Sendrier's Code-Based Signature Scheme

In 2001, Courtois, Finiasz and Sendrier proposed the first practical signature scheme based on coding theory [CFS01]. We present here a slightly modified version presented in [Dal08] that admits a reductionist security proof.

Recall that an $[n, k]$ -binary linear code \mathcal{C} is a linear subspace of dimension k of \mathbb{F}_2^n , where \mathbb{F}_2 is the field with 2 elements. Elements of \mathbb{F}_2^n are called *words* while elements of \mathcal{C} are called *codewords*. An $[n, k]$ -binary linear code is entirely defined by a $(n - k) \times n$ binary matrix H : codewords are words $x \in \mathbb{F}_2^n$ that satisfy $Hx^T = 0$. We call *syndrome* of a word $x \in \mathbb{F}_2^n$ the quantity $s = Hx^T$. The *Hamming weight* of a word x denoted by $hw(x)$ is the number of its non-zero positions. The *Hamming distance* between two words x and y , denoted by $d_H(x, y)$ is the number of positions where they differ. The minimal distance of a code \mathcal{C} is $d = \min_{x, y \in \mathcal{C}} d_H(x, y)$; by linearity $d = \min_{x \in \mathcal{C}} hw(x)$.

If $t \leq \lfloor \frac{d-1}{2} \rfloor$, for any syndrome $s \in \mathbb{F}_2^{n-k}$, there is *at most* one word $x \in \mathbb{F}_2^n$ such that $hw(x) \leq t$ and $Hx^T = s$. A vector s of \mathbb{F}_2^{n-k} is said to be t -decodable (or simply decodable) in the code defined by H if there is such a word x , called t -decoding of s . Knowing if a random syndrome $s \in \mathbb{F}_2^{n-k}$ is t -decodable in the code defined by a random matrix H has been proven to be \mathcal{NP} -complete [BMvT78] and the cost of the best decoding algorithm [BLP08] is exponential in the length and the rate of the code. But for more structured code, the decoding can be easily performed.

Goppa codes are subfield subcodes of particular alternant codes. For given integers m and t , binary Goppa codes are of length $n = 2^m$, of dimension $k = n - mt$. Let us denote $\mathcal{G}_{m,t}$ the family of such Goppa codes; we have $|\mathcal{G}_{m,t}| = 2^{tm}/t!$. Their algebraic structure, which can be efficiently hidden [Sen02], provides a good t -decoding algorithm. Thus, they are good candidates for a cryptographic use. Moreover, as discussed in [CFS01], a random vector randomly distributed over \mathbb{F}_2^{mt} has probability $1/t!$ to be t -decodable in a Goppa code of parameters m and t .

Courtois, Finiasz and Sendrier signature scheme maps a message to a t -decodable syndrome in a public code by random sampling, using properties of hash functions in the random oracle model. The knowledge of the algebraic structure of the code, which has been previously hidden, is used to decode this syndrome, providing the signature. The modified version of Courtois, Finiasz and Sendrier signature scheme mCFS is given by the following algorithms:

- mCFS.Setup(1^κ) : Select parameters m and t such that the t -decoding in a Goppa code of length 2^m , of dimension $2^m - mt$ has complexity at least 2^κ and a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{mt}$. Output the set of parameters $\mathcal{P} = (m, t, \mathcal{H})$.
- mCFS.Gen(\mathcal{P}) : Pick a random binary Goppa code \mathcal{C}_0 in $\mathcal{G}_{m,t}$. Let H_0 be a parity check matrix of \mathcal{C}_0 and Δ_{H_0} be a t -decoding algorithm in \mathcal{C}_0 . Pick a random non singular $mt \times mt$ binary matrix U and a random permutation matrix P of size $2^m \times 2^m$. The public key is $PK = H = UH_0P$ and the secret key is $SK = (\Delta_{H_0}, U, P)$.
- mCFS.Sign($\mathcal{P}, M, SK = (\Delta_{H_0}, U, P)$) : Pick a random value r in $\{1, \dots, 2^{mt}\}$ and compute $s = \mathcal{H}(M \| r)$. Test if $U^{-1}s$ is t -decodable in \mathcal{C}_0 . If not, repeat previous operations. Then, compute $x' = Px^T = \Delta_{H_0}(U^{-1}s)$, the t -decoding of $U^{-1}s$ in \mathcal{C}_0 . Since permutation does not alter the weight of a word, $x = x'P^{-1}$ is a t -decoding of s in the code defined by the parity check matrix H . Output $\sigma = (x, r)$ as a signature of m relatively to the public key H .
- mCFS.Verify($\mathcal{P}, M, PK = H, \sigma = (x, r)$) : Test if $\mathcal{H}(M \| r) = Hx^T$ and $hw(x) \leq t$.

4 Our Code-Based Threshold Ring Signature Scheme

We present a new code-based threshold ring signature scheme based on coding theory. This scheme combines both techniques used by Bresson *et al.* presented Sec. 2.3 and Courtois *et al.* presented Sec 3.

4.1 Description of the Scheme

Our scheme is described by the four following algorithms:

- Setup and Gen that are almost identical to those of mCFS signature scheme but Setup fixes two new parameters: an indexed family $(E_{k,i})$ ($i \in \{1, \dots, N\}$) of

random permutations that encrypt messages of mt bits with keys k of length κ and an additional hash function $\mathcal{H}_\kappa : \{0,1\}^* \rightarrow \mathbb{F}_2^\kappa$. In the following, each user i of a ring \mathcal{N} is associated to a key pair (PK_i, SK_i) where $SK_i = (\Delta_{H_0^{(i)}}, U^{(i)}, P^{(i)})$ and $PK_i = H^{(i)} = U^{(i)}H_0^{(i)}P^{(i)}$.

- $\text{Sign}(\mathcal{P}, M, \mathcal{N}, \mathcal{SK}_{\mathcal{L}})$ that proceeds as follows:

1. Initialization:
 - (a) Compute a key $k \leftarrow \mathcal{H}_\kappa(M)$.
 - (b) Compute an initial value $y_0 \leftarrow \mathcal{H}(M\|\mathcal{N})$
2. For all user $i \in \mathcal{N} \setminus \mathcal{L}$ do
 - (a) randomly pick $x_i \xleftarrow{R} \{x \in \mathbb{F}_2^{2^m} | hw(x) \leq t\}$
 - (b) randomly pick $r_i \xleftarrow{R} \{1, \dots, 2^{mt}\}$
 - (c) compute $y_i \leftarrow H^{(i)}x_i^T + \mathcal{H}(M\|r_i)$
3. Compute an interpolation polynomial

$$P \in \mathbb{F}_{2^{mt}}[X] \text{ such that } \begin{cases} d(P) = |\mathcal{N}| - |\mathcal{L}| \\ P(0) = y_0 \\ P(i) = E_{k,i}(y_i) \text{ for all } i \in \mathcal{N} \setminus \mathcal{L} \end{cases}$$

4. For all users $i \in \mathcal{L}$ do
 - (a) $x_i \leftarrow \emptyset$
 - (b) While $x_i = \emptyset$ do
 - i. $r_i \xleftarrow{R} \{1, \dots, 2^{mt}\}$
 - ii. $x_i^0 \leftarrow \Delta_{H_0^{(i)}}(U_i^{-1} \cdot (E_{k,i}^{-1}(P(i)) + \mathcal{H}(M\|r_i)))$
 - iii. If $x_i^0 \neq \emptyset$ then $x_i \leftarrow x_i^0 P_i^{-1}$
5. Output $\sigma = (N, x_1, \dots, x_n, r_1, \dots, r_n, P)$ as a ring signature.

Note that Step 4b proceeds to $\ell(t!)$ decodings in average.

- $\text{Verify}(\mathcal{P}, M, \sigma, \mathcal{N})$ where $\sigma = (N, x_1, \dots, x_n, r_1, \dots, r_n, P)$ that outputs valid if $P(0) = \mathcal{H}(M\|\mathcal{N})$ and for all user $i \in \mathcal{N}$, $hw(x_i) \leq t$ and $P(i) = E_{\mathcal{H}_\kappa(M),i}(H^{(i)}x_i^T + \mathcal{H}(M\|r_i))$ or outputs invalid otherwise.

4.2 Security Analysis

Consistency is straight forward and *anonymity* is easy to verify. Every polynomial of degree $N - \ell$ can be obtained by interpolation of $N - \ell + 1$ values. The threshold ring signature scheme interpolates some values y_i given by an ideal cipher $E_{k,i}$ and then uniformly distributed. Thus, the signers set is perfectly hidden by the algorithm: for given $\kappa > 0$, $\mathcal{N}, \mathcal{L} \in \mathcal{N}$, $M \in \{0,1\}^*$, where \mathcal{N} is a ring such that for all member $i \in \mathcal{N}$ in the ring $(PK_i, SK_i) \leftarrow \text{Gen}(\mathcal{P})$, the signature algorithm produces a signature $\sigma = (\mathcal{N}, x_1, \dots, x_n, r_1, \dots, r_n, P)$. For any user $u \in \mathcal{L}$, a simulation $\text{Simulate}(\mathcal{P}, M, \mathcal{N}, \mathcal{L}, \mathcal{SK}_{\mathcal{N} \setminus \{u\}})$ first picks a random user $j \in \mathcal{N}$, and “swap” the role of u and j : it performs the same initialization step than Sign and computes for any member $i \in (\mathcal{N} \cup \{u\}) \setminus (\mathcal{L} \cup \{j\})$ a syndrome y'_i in the same way than Sign . It is easy to see that there is a random sequence such that for all $i \in (\mathcal{N} \cup \{u\}) \setminus (\mathcal{L} \cup \{j\})$, $(x'_i, r'_i) = (x_i, r_i)$. Then, the simulation computes an interpolation polynomial $P' \in \mathbb{F}_{2^{mt}}$ from the $N - \ell + 1$

values y'_i ; since we consider that there is a random sequence such that $y'_i = y_i$ for all i , then $P' = P$. For any signer $i \in \mathcal{L} \setminus \{u\}$, the simulation computes couples (x'_i, r'_i) of words of $\mathbb{F}_2^{2^m}$ of weight at most t and random values by using its private key, in the same way **Sign** does; thus, there is a random sequence such that, for all $i \in \mathcal{L} \setminus \{u\}$, $(x'_i, r'_i) = (x_i, r_i)$. Finally, the simulation uses the private key of j to compute a couple (x_j, r_j) such that $r_j \xleftarrow{R} \{1, \dots, 2^{mt}\}$, $hw(x_u) \leq t$ and $P(j) = E_{\mathcal{H}, j}(H^{(j)}x_j^T + \mathcal{H}(m \| r_j))$. Thus there is a random sequence such that $\text{Simulate}(\mathcal{P}, m, \mathcal{N}, \mathcal{L}, \mathcal{SK}_{\mathcal{N} \setminus \{u\}}) = \text{Sign}(\mathcal{P}, m, \mathcal{N}, \mathcal{SK}_{\mathcal{L}})$.

Unforgeability requires a little more attention. We rely its difficulty to those of two problems issued from coding theory. First, we shall consider the following problem:

Definition 2 (Goppa Parameterized Bounded Decoding problem)

Input: A $(n - k) \times n$ binary matrix H and a syndrome $s \in \mathbb{F}_2^{n-k}$

Output: A word $e \in \mathbb{F}_2^n$ such that $hw(e) \leq \frac{n-k}{\log_2 n}$ and $He^T = s$

Note that GPBD problem is stated for random codes but our scheme uses Goppa codes. Thus we shall also consider the *Goppa Code Distinguishing* problem (GD).

Definition 3 (Goppa Code Distinguisher). A distinguisher \mathcal{D} for a permuted Goppa Code is an algorithm which takes as input a parity check matrix H and outputs a bit. \mathcal{D} outputs 1 with probability $\Pr[H \xleftarrow{R} \mathcal{G}(n, k) : \mathcal{D}(H) = 1]$ if H is a random binary parity check matrix of a Goppa code $\mathcal{G}(n, k)$ and outputs 1 with probability $\Pr[H \xleftarrow{R} \mathcal{B}(n, k) : \mathcal{D}(H) = 1]$ if H is a random binary matrix. We call the advantage of a distinguisher \mathcal{D} , denoted by $Adv_{n, k}^{GD}(\mathcal{D})$, the following quantity:

$$\left| \Pr[H \xleftarrow{R} \mathcal{G}(n, k) : \mathcal{D}(H) = 1] - \Pr[H \xleftarrow{R} \mathcal{B}(n, k) : \mathcal{D}(H) = 1] \right|.$$

Denote by $Succ_{n, k}^{GPBD}(\tau)$ the probability of success of the best algorithm that solves the Goppa Parameterized Bounded Decoding problem in time τ and denote by $Adv_{n, k}^{GD}(\tau)$ the advantage of the best distinguisher for a permuted Goppa code in time τ .

Theorem 2 (Unforgeability). Let \mathcal{A} be an ℓ -adversary against the scheme presented Section 4.1 that outputs a forgery in time τ with probability ϵ . \mathcal{A} makes $q_{\mathcal{H}}$ queries to a hash oracle \mathcal{H} , $q_{\mathcal{E}}$ queries to a cipher oracle \mathcal{E} and q_{Σ} queries to a signing oracle Σ . \mathcal{A} also can corrupt the private keys of $\ell - 1$ users. We have:

$$\begin{aligned} \epsilon &\leq q_{\mathcal{E}} q_{\mathcal{H}} (N - t + 1) \binom{N}{t} Succ_{n, k}^{GPBD}(\tau') + Adv_{n, k}^{GD}(\tau') \\ &\quad - \frac{q_{\mathcal{E}}}{2^{mt}} \binom{N}{\ell} \left(\frac{q_{\mathcal{E}}}{N - \ell} \right)^{N - \ell} \end{aligned}$$

where $n = 2^m$, $k = 2^m - mt$ and $\tau' = Nmt^2 q_{\Sigma}$.

Proof. Let ϵ be the probability of success in time τ of \mathcal{A} . We describe how to use it to build an algorithm \mathcal{D} that inverses GPBD problem. \mathcal{D} receives as inputs a random $mt \times 2^m$ binary matrix H^* and a random vector s^* of \mathbb{F}_2^{mt} . Its goal is to output x^* such that $hw(x^*) \leq t$ and $H^*(x^*)^T = s^*$.

\mathcal{D} randomly picks a user i_0 in the set of users in the ring \mathcal{N} and a subset $I_0 \subset \mathcal{N}$ of $\ell - 1$ elements such that $i_0 \notin I_0$. \mathcal{D} hopes that all corrupted users are in I_0 . It sets i_0 's public key PK_{i_0} to H^* and for all users $i \in I_0$, sets key pair (PK_i, SK_i) by running the generation algorithm. For all other users, \mathcal{D} uses a parity check matrix of random permuted Goppa as public key: their private keys will never be used. \mathcal{D} also randomly picks two indexes $q_{\mathcal{E},0}$ and $q_{\mathcal{H},0}$ respectively in $[1, \dots, q_{\mathcal{E}}]$ and $[1, \dots, q_{\mathcal{H}}]$ where $q_{\mathcal{H}}$ is the total number of queries to the hash oracle \mathcal{H} and $q_{\mathcal{E}}$ is the total number of queries (E or E^{-1}) to the cipher oracle. Finally, it picks a random vector \tilde{s} in \mathbb{F}_2^{mt} . Then, \mathcal{A} is initialised with the set of public keys $\{PK_i\}_{i \in \mathcal{N}}$.

\mathcal{D} classically simulates the oracle \mathcal{H} by answering a random value in \mathbb{F}_2^{mt} for each new query and maintaining a list of queries. It also simulates the oracle \mathcal{E} according to a permutation by answering a random value for each new query. But to the $q_{\mathcal{H},0}$ -th query to the hash oracle, it answers by \tilde{s} and to the $q_{\mathcal{E},0}$ -th query E^{-1} to the cipher oracle, it answers by $s^* + \tilde{s}$. Note that \mathcal{D} must maintain a list of the queries such that it remembers whether x is the answer to a query $E^{-1}(y)$ or y is the answer to a query $E(x)$. \mathcal{A} may corrupt up to $\ell - 1$ users by querying its private key. \mathcal{D} trivially answers the queries but fails if \mathcal{A} queries the key of user i_0 .

\mathcal{D} simulates the signing oracle Σ for any set of signers by picking at random elements of the signature and adapting the answers of the oracle \mathcal{E} . More precisely, for a query (M, L) , \mathcal{D} simulates the queries $\mathcal{H}_k(M)$ and $\mathcal{H}(M \parallel \mathcal{N})$ that fixes k and y_0 , picks a random polynomial P of degree $N - \ell$ such that $P(0) = y_0$ and picks N random vectors x_1, \dots, x_n of \mathbb{F}_2^n of weight less than t and N random values r_1, \dots, r_n . Then, \mathcal{D} simulates \mathcal{H} , fixes the values for $E_{k,i}(H^{(i)}x_i^T + \mathcal{H}(M \parallel r_i))$ to $P(i)$ and answers $(m, x_1, \dots, x_n, r_1, \dots, r_n, P)$ as signature. With probability ϵ , \mathcal{A} outputs a forgery $\sigma^* = (M^*, x_1, \dots, x_n, r_1, \dots, r_n, P^*)$. By definition, $P^*(0) = \mathcal{H}(M^* \parallel \mathcal{N})$ and for all $i \in \mathcal{N}$, $P^*(i) = E_{\mathcal{H}(M), i}(H^{(i)}x_i^T)$. If $H^*x_{i_0}^T = s^*$ and $hw(x_{i_0}) \leq t$, then \mathcal{D} outputs x_{i_0} .

Bresson and al. showed that, with probability at least $\epsilon + \frac{q_{\mathcal{E}}}{2^{mt}} \binom{N}{\ell} \left(\frac{q_{\mathcal{E}}}{N-\ell}\right)^{N-\ell}$, \mathcal{A} produces a forgery such that there are at most $N - \ell$ cipher queries to \mathcal{E} for $P^*(i)$. Thus there are at least ℓ indexes i such that \mathcal{A} made a decrypt query to \mathcal{E} for $P^*(i)$; let I^* be the set of those indexes. Since \mathcal{A} can corrupt up to ℓ users, there are at least an index i^* such that \mathcal{A} did not corrupt its private key. Let q^* be the index of the query to \mathcal{E} for $P^*(i^*)$. With probability at least $1/q_{\mathcal{E}}$, $q^* = q_{\mathcal{E},0}$; with probability at least $1/(n - \ell + 1)$, $i^* = i_0$ and with probability at least $1/q_{\mathcal{H}}$ the $q_{\mathcal{H},0}$ -th query to the hash oracle was $(M \parallel r_{i_0})$. Then x_{i_0} has weight less than t and satisfies $Hx_{i_0}^T = E_{\mathcal{H}(M), i}^{-1}(P(i_0)) + \mathcal{H}(M \parallel r_{i_0}) = s^* + \tilde{s} + \tilde{s} = s^*$. Thus, x_{i_0} is a t -decoding of s^* .

The running time τ' of \mathcal{D} is essentially the running time τ of \mathcal{A} and the cost of Nq_Σ syndrome computation, whose cost is mt^2 . Note that replacing the public

key of user i_0 does not alter the probability of success of the simulation more than the advantage $\text{Adv}_{2^m, 2^m - mt}^{GD}(\tau')$ of the best adversary at solving the *permuted Goppa code distinguishing*: otherwise \mathcal{D} would provide a better distinguisher. We denote $\epsilon_{GPBD} = \text{Succ}_{2^m, 2^m - mt}^{GPBD}(\tau')$ and $\epsilon_{GD} = \text{Adv}_{2^m, 2^m - mt}^{GD}(\tau')$; we obtain:

$$\left(\epsilon + \frac{q\varepsilon}{2^{mt}} \binom{N}{\ell} \left(\frac{q\varepsilon}{N-\ell} \right)^{N-\ell} + \epsilon_{GD} \right) \frac{1}{q\varepsilon q_{\mathcal{H}}(N-t+1)\binom{N}{t}} \leq \epsilon_{GPBD}$$

this concludes the proof.

4.3 Efficiency of the Proposed Scheme

In these section, we give some secure parameters for our scheme and the resulting efficiency of the protocol.

Choice of Parameters. As seen in the security proof, our scheme will be unforgeable w.r.t. insider corruption if both GD and GPBD problems are difficult for the public code. The best known algorithm that solves GPBD problem are [BLP08] and an unpublished generalized birthday attack stated by Bleichenbacher. They work exponentially in the length and rate of the code. Very few is known about the distinguishability of Goppa codes. In practice, the only structural attack [LS01] consists in enumerating all Goppa codes and then testing equivalence with the public key. The code equivalence problem can be solved efficiently thanks to Sendrier's algorithm [Sen00]. They are $2^{tm}/t$ binary t -errors correcting Goppa codes of length $n = 2^m$. Due to properties of Goppa codes, only one out of mn^3 must be tested and finally, the cost of equivalence testing cannot be lower than $n(tm)^2$ (a Gaussian elimination) [CFS01]. Then a distinguisher for permuted Goppa codes has a cost not less than $tm2^{m(t-2)}$ elementary operations.

In order to reduce the number of decoding needed to sign a message, the parameter t has to be as small as possible, say no more than $t = 10$. Originally, the authors of [CFS01] proposed to use parameters $m = 16$ and $t = 9$ but this set of parameters is subject to attacks via generalized birthday attack. Choosing parameters $m = 22$ and $t = 9$ prevents this attack, reaching its time complexity to $2^{81.4}$ [FS09].

Performances. Each public key $H^{(u)}$ is a $2^m \times mt$ matrix which takes about 1.2MBytes to be stored for parameters $m = 16$ and $t = 9$ and 99MBytes for parameters $m = 22$ and $t = 9$. For a ring of N users, signature generation requires $N - \ell$ syndrome computing (whose cost is mt^2), n polynomial evaluations and $\ell(t!)$ decodings, each of them requiring t^2m^3 operations. Verification requires $N + 1$ polynomial evaluations and N syndrome computations.

For a ring of N users, the signature is composed of N vectors x_i of $\mathbb{F}_2^{2^m}$ of weight less than t , N random values r_i in $\{0, \dots, 2^{mt} - 1\}$ and an interpolation polynomial of degree $N - \ell$ where ℓ is the number of signers. As mentioned in [CFS01], there is no need to store the full vector x_i but only an index characterizing the vector from the $\sum_{j=1}^t \binom{2^m}{j}$ vectors of weight less than t . The storage

of the polynomial of $\mathbb{F}_{2^{mt}}$ of degree $N - \ell$ requires $(N - \ell) \times mt$ bits. Thus the size of the signature is

$$\left(\left\lfloor \log_2 \sum_{j=1}^t \binom{2^m}{j} \right\rfloor + 2mt + 1 \right) \times N - mt\ell.$$

With parameters $(m, t) = (16, 9)$, this quantity is equal to $414N - 144\ell$ bits but achieve a security level of $2^{63.3}$. With parameters $(m, t) = (22, 9)$ that reach a security level of $2^{81.7}$, the size of a signature is $579N - 198\ell$ bits.

5 Conclusion

We have proposed a new ℓ -out-of- N threshold ring signature scheme which is provably secure in both (equivalent) random oracle and ideal cipher models and which achieves a small signature size ($414N - 144\ell$ and $579N - 198\ell$ bits for respective security levels of $2^{63.3}$ and $2^{81.7}$). However, the large size of public keys and the relative slowness of signature generation are prohibitive for a short term practical use of our scheme. Many efforts are actually made to reduce the size of the keys used in code-based cryptography (see [BCGO09, MB09] for a recent proposal and a work in progress) and a significant advance should greatly improve the performances of our scheme.

References

- [AFG⁺08] Augot, D., Finiasz, M., Gaborit, P., Manuel, S., Sendrier, N.: Fast syndrome-based hash function. SHA-3 Proposal: FSB (2008), <http://www-roc.inria.fr/secret/CBCrypto/index.php?pg=fsb>
- [AHR05] Adida, B., Hohenberger, S., Rives, R.L.: Ad-hoc-group signatures from hijacked keypairs. In: DIMACS Workshop on Theft in e-commerce (2005), <http://theory.lcs.mit.edu/~rivest/publications.html>
- [AMCG08] 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. 31–46. Springer, Heidelberg (2008)
- [BCGO09] Berger, T., Cayrel, P.L., Gaborit, P., Otmani, A.: Reducing key length of the McEliece cryptosystem. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 60–76. Springer, Heidelberg (2009)
- [BKM06] 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)
- [BLP08] Bernstein, J., Lange, T., Peters, C.: Attacking and defending the McEliece cryptosystem. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 31–46. Springer, Heidelberg (2008)
- [BMvT78] Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.: On the inherent intractability of certain coding problems. IEEE Trans. Inform. Th. 24 (1978)

- [BR93] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
- [BSS02] Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg (2002)
- [CFS01] Courtois, N., 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)
- [CGH04] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. Journal of the ACM 51(4), 557–594 (2004)
- [CGS07] Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (2007)
- [COV07] Cayrel, P.L., Otmani, A., Vergnaud, D.: On Kabatianskii-Krouk-Smeets signatures. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 237–251. Springer, Heidelberg (2007)
- [CPS08] Coron, J.-S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (2008)
- [CvH92] Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
- [Dal08] Dallot, L.: Towards a concrete security proof of Courtois, Finiasz and Sendrier signature scheme. In: Lucks, S., Sadeghi, A.-R., Wolf, C. (eds.) WEWoRC 2007. LNCS, vol. 4945, pp. 65–77. Springer, Heidelberg (2008)
- [DH76] Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inform. Th. 22(6), 644–654 (1976)
- [DKNS04] Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad-hoc groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
- [FS09] 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), <http://eprint.iacr.org/2009/414>
- [JSI96] Jakobson, M., Sako, K., Implantiazzo, R.: Designted verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
- [KI01] Kobara, K., Imai, I.: Semantically secure McEliece public-key cryptosystems — conversions for McEliece PKC. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 19–35. Springer, Heidelberg (2001)
- [LM09] Lyubashevsky, V., Micciancio, D.: On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In: Tai, X.-C., et al. (eds.) CRYPTO 2009. LNCS, vol. 5677, pp. 577–594. Springer, Heidelberg (2009)
- [LN09] Leurent, G., Nguyen, P.Q.: How risky is the random-oracle model? In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 445–464. Springer, Heidelberg (2009)
- [LS01] Loidreau, P., Sendrier, N.: Weak keys in McEliece public-key cryptosystem. IEEE Trans. Inform. Th. 47(3), 1207–1212 (2001)

- [MB09] Misoczki, R., Barreto, P.S.L.M.: Compact McEliece keys from goppa codes. Cryptology ePrint Archive, Report 2009/187 (2009), <http://eprint.iacr.org/>
- [McE78] McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Technical report, DSN Progress report # 42-44, Jet Propulsion Laboratory, Pasadena, California (1978)
- [Nao02] Naor, M.: Deniable ring authentification. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 481–498. Springer, Heidelberg (2002)
- [OTD08] Otmani, A., Tillich, J.P., Dallot, L.: Cryptanalysis of a McEliece cryptosystem based on quasi-cyclic LDPC codes. In: Faugre, J.C., Wang, D. (eds.) Proceedings of the first international conference on symbolic computation and cryptography. LMIB, pp. 69–81 (2008)
- [RST01] 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)
- [Sen00] Sendrier, N.: Finding the permutation between equivalent codes: the support splitting algorithm. IEEE Trans. Inform. Th. 46, 1193–1203 (2000)
- [Sen02] Sendrier, N.: Cryptosystèmes à clé publique basés sur les codes correcteurs d’erreurs. Habilitation à diriger les recherches, Université Pierre et Marie Curie, Paris 6, Paris, France, Mars (2002) (in French)
- [Sha79] Shamir, A.: How to share a secret. Commun. of the ACM 22(11), 612–613 (1979)
- [Ste90] Stern, J.: An alternative to the Fiat-Shamir protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 173–180. Springer, Heidelberg (1990)
- [Ste96] Stern, J.: A new paradigm for public key identification. IEEE Trans. Inform. Th. 42(6), 1757–1768 (1996)
- [SW07] 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)
- [ZLC07] Zheng, D., Li, X., Chen, K.: Code-based ring signature scheme. International Journal of Network Security 5(2), 154–157 (2007)

A New Protocol for the Nearby Friend Problem

Sanjit Chatterjee, Koray Karabina, and Alfred Menezes

Department of Combinatorics & Optimization, University of Waterloo,
Waterloo, Ontario, N2L 3G1, Canada
`{s2chatte, kkarabin, ajmeneze}@uwaterloo.ca`

Abstract. The question of location privacy has gained a special significance in the context of location-based services for mobile devices. The challenge is to allow the users to benefit from location-based services without disclosing their private location information unless necessary and that too only to the party eligible to receive that information. In this work, we investigate the so-called nearby friend problem. The problem has emerged in the context of location-based services such as social networking and is closely related to the issue of location privacy. In particular, we are interested in the question of how Alice can efficiently determine whether a friend Bob is at a nearby location or not. This has to be achieved without a third party and where Alice neither reveals any information about her own location nor can she extract any information about Bob's actual location when they are not nearby. Similarly, no eavesdropper should be able to gain any information about their actual locations, whether they are actually nearby or not. The problem becomes more challenging as both Alice and Bob are restricted in computational power and communication bandwidth. Starting from an earlier work by Zhong et al., we formalize the protocol definition and the security model and then propose a new protocol that solves the problem in the proposed security model. An interesting feature of the protocol is that it does not depend on any other cryptographic primitive, thus providing a new approach to solve the nearby friend problem. Our basic protocol and its extensions compare favorably with the earlier solutions for this problem. The protocol might be of use in other privacy-preserving applications.

1 Introduction

The rapid advent of mobile computing and consequent introduction of new technologies like smart phones and GPS together with the ever increasing mobility of the human populace have opened up the possibility of a plethora of services that could not even be conceived of a decade earlier. Different kinds of location-based services are increasingly becoming popular among the users of mobile devices [20]. However, this also calls forth a concern about the privacy of the location information of the users of such devices. So one is confronted with two contradictory goals – users would like to take advantage of different services that cater to their needs based on a particular location while at the same time

protecting the very privacy of that location information. How to meet these apparently contradictory goals, or in other words to get the best of both worlds, is a major technological challenge.

In this work we focus our attention on the following scenario. Suppose a user Alice, in a distributed mobile computing environment, wants to determine whether one of her friends, Bob, is in a nearby location or not. This is the so-called *nearby friend problem* and is relevant in the context of social networking and buddy tracking applications [2]. The trivial solution for Alice is to contact Bob and either reveal her own location to him or ask Bob to do so. But this comes at the cost of compromising one of the friend's location privacy altogether even when they are not actually nearby. Instead the users would like to have some control over their private location information so that Alice can learn Bob's location only if they are actually nearby and Bob is willing to reveal his location to Alice. The problem might be solved in a privacy-preserving way if we assume the existence of an *ideal* trusted third party with whom each party maintains a secret communication channel. Parties send their private location information to that trusted party who then decides whether they are nearby or not and informs them accordingly.

Based on the above discussion one may notice that the nearby friend problem is somewhat akin to the socialist millionaires' problem [15] in cryptology where the two participants learn whether they both have the same wealth or not without first disclosing their wealth. This can also be viewed as a concrete instantiation of the privacy-preserving set operations problem [16] or the two-party private matching problem [11]. However, the challenge here is to solve it in the resource-constrained environment of mobile devices.

We would like to solve the problem in a distributed setting without taking recourse to any trusted party and moreover solve it in such a way that the computational requirement, communication bandwidth, and number of communication rounds can be kept to a minimum. This, of course, has to be achieved in a secure way in an adversarial environment, i.e., the participants need the assurance that neither its communicating partner receives any information other than what (s)he is entitled to nor should it be possible for an eavesdropper to infringe upon their privacy.

Some initial attempts to solve the above problem can be found in the works of Atallah and Du [3] and Koenen and Oleshchuk [17]. However, the solutions are less than satisfactory as the former requires a semi-trusted third party and several rounds of communications while the latter has some security vulnerabilities (see [22]). In a pioneering paper, Zhong, Goldberg and Hengartner [22] proposed three protocols to solve the above problem based on a cryptographic primitive called homomorphic encryption [12]. They assume that Alice and Bob establish a secure channel that provides both confidentiality and authentication prior to running the protocols. Such a communication channel can be established, for example, through a TLS connection [8]. Their first protocol, called Louis, requires the service of an online semi-trusted third party. The second protocol, called Lester, does not require any third party but has the disadvantage that,

with some additional work, Alice might be able to determine Bob’s location even if they are not considered to be in nearby locations. The third protocol, called Pierre, does not suffer from any of the above deficiencies. It is quite efficient, in particular when one uses the homomorphic encryption scheme due to Cramer, Genarro and Schoenmakers [7] in the elliptic curve setting. Zhong [21] later proposed another protocol, called Wilfrid, based on the idea of private set intersection of [11]. This too makes use of homomorphic encryption and a communication channel that is both confidential and authenticated.

Our Contribution. We take the work of Zhong et al. [22] as our point of departure by formalizing the statement of the nearby friend problem as well as its security model. We then proceed to propose a new protocol, called \mathcal{NFP} -I, in the line of Diffie-Hellman type of simple password exponential key exchange (SPEKE) [14]. An advantage of our protocol is that it only assumes the existence of an authenticated channel between the two parties, not a confidential one. Also note that we do not require any other cryptographic primitive such as homomorphic encryption that was used in [22] and [21]. The protocol is quite simple and we propose it as a new primitive that might serve as a building block for more complex privacy-preserving applications such as private matching [11]. We provide a security analysis to show that the protocol meets its desired objectives in the security model under the decision Diffie-Hellman assumption and a variant of it. We also discuss several extensions of this basic protocol. Our protocol is quite efficient and compares favorably with the currently best protocol known, Wilfrid, that achieves the same functionality (see Table 1).

Table 1. Performance comparison between Wilfrid and \mathcal{NFP} -I when implemented in a group G where the discrete log problem is assumed to be hard. Both protocols require two communication steps. The message size is the number of elements of G .

	Alice → Bob		Bob → Alice		decide whether nearby: # exps by Alice
	# exps	message size	# exps	message size	
Wilfrid	4	4	3	2	1
\mathcal{NFP} -I	1	1	2	2	1

The remainder of this paper is organized as follows. In Section 2 we introduce a formal definition of the nearby friend problem and its security model. In Section 3 we propose the basic protocol \mathcal{NFP} -I and discuss its security. In Section 4 we discuss some extensions of the basic primitive. Finally we conclude in Section 5 with some open problems in this emerging area.

2 Problem Definition and Security Model

In this section we propose a formal definition and security model for the nearby friend problem. The protocol definition is developed from the informal description of the problem in [22].

2.1 Nearby Friend Problem

Let \mathcal{L} be the publicly known set of all possible locations. The elements of \mathcal{L} can be expressed in two-dimensional coordinates such as floating point numbers or integers to represent some geographical location or block on the surface of the earth. Alternatively, they can be some description of a location such as the ZIP code or the street address. We assume there exists a natural notion of nearbiness for the elements of \mathcal{L} such that given $L_1, L_2 \in \mathcal{L}$ it is easy to decide whether they are nearby or not. For example, when locations are represented by Cartesian coordinates, given two such elements $L_1, L_2 \in \mathcal{L}$ one can compute the Euclidean distance between them. In this context, we say two location points $L_1, L_2 \in \mathcal{L}$ are nearby (denoted as $L_1 \equiv L_2$) if the Euclidean distance between L_1 and L_2 is less than or equal to some threshold distance d .

A scheme to solve the nearby friend problem is a two-party protocol \mathcal{NFP} between an initiator \mathcal{I} and a responder \mathcal{R} . It is assumed that both parties agree upon a description of \mathcal{L} (and the corresponding notion of nearbiness) before running the protocol. It is also assumed that the parties establish a communication link which is authenticated, but not necessarily confidential, before executing the protocol. Such an authenticated link can be established, for example, by deriving a session key using a standard key agreement protocol and then MACing all messages using that session key. \mathcal{I} 's input to \mathcal{NFP} is her secret location information $L_{\mathcal{I}} \in \mathcal{L}$ while \mathcal{R} 's input is his secret location information $L_{\mathcal{R}} \in \mathcal{L}$. Both parties exchange some messages based on their secret input and what they receive from the other party. At the end, the protocol generates an output for the initiator, namely the initiator will learn whether $L_{\mathcal{I}} \equiv L_{\mathcal{R}}$ or not. (The responder does not learn any information about the initiator's location $L_{\mathcal{I}}$.)

2.2 Security Model

To model (all) possible leakage of information, we consider two different kinds of computationally-bounded adversaries – passive and active. In the former case, the target of an eavesdropper who observes the exchanged messages in a protocol run is either to extract any information about \mathcal{I} or \mathcal{R} 's true location or to just decide whether they are actually nearby or not. The case of a passive adversary is relevant in our context as we only assume the existence of an authenticated link between \mathcal{I} and \mathcal{R} . The work of Zhong et al., in contrast, assumes the existence of a communication channel (such as TLS connection) which provides both confidentiality and authentication. To prevent active attacks, on the other hand, one of \mathcal{I} or \mathcal{R} is treated as the adversary where the goal is to extract any information about the other party's location. In the following description, a protocol transcript stands for the collection of all the messages transmitted between the two parties during a typical protocol run.

Passive Adversary. We would like to provide the adversary with as much power as possible while limiting its task to the minimum. Intuitively the goal is to show that even if the adversary has some a priori information about \mathcal{I} 's (or \mathcal{R} 's) possible locations it should not be able to distinguish the protocol transcript

for one location of \mathcal{I} (or \mathcal{R}) from any other. We also want the assurance that after observing the protocol messages the adversary should not be able to infer whether the participants are actually nearby or not. In other words, the protocol transcript when \mathcal{I} and \mathcal{R} are actually nearby should not be distinguishable from a protocol transcript when they are not. This is formally modeled through the following two games between an adversary \mathcal{A} and a challenger \mathcal{C} . In the following and all the subsequent games between \mathcal{A} and \mathcal{C} it is assumed that the set of location points \mathcal{L} and the measure of nearbiness are publicly known and hence available to \mathcal{A} .

The initiator's privacy. This is modelled by the following game.

\mathcal{G}_1 : \mathcal{A} chooses two distinct locations $L_{\mathcal{I},0}, L_{\mathcal{I},1} \in \mathcal{L}$ for \mathcal{I} and one location $L_{\mathcal{R}} \in \mathcal{L}$ for \mathcal{R} .¹ The challenger \mathcal{C} selects $b \in_R \{0, 1\}$ and provides \mathcal{A} with a protocol transcript corresponding to $L_{\mathcal{I},b}$ as \mathcal{I} 's location and $L_{\mathcal{R}}$ as \mathcal{R} 's location. \mathcal{A} outputs a guess b' and wins if $b = b'$. We define the advantage of the adversary \mathcal{A} in attacking the scheme \mathcal{NFP} with respect to \mathcal{I} 's location as

$$\text{Adv}_{\mathcal{NFP}, \mathcal{A}}^{\mathcal{I}} = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that \mathcal{NFP} preserves location privacy for \mathcal{I} against a passive adversary if $\text{Adv}_{\mathcal{NFP}, \mathcal{A}}^{\mathcal{I}}$ is negligible for all possible \mathcal{A} .

The responder's privacy. The aim of the adversary in this case is similar to that in the former. Formally we have:

\mathcal{G}_2 : \mathcal{A} chooses one location $L_{\mathcal{I}} \in \mathcal{L}$ for \mathcal{I} and two distinct locations $L_{\mathcal{R},0}, L_{\mathcal{R},1} \in \mathcal{L}$ for \mathcal{R} . \mathcal{C} selects $b \in_R \{0, 1\}$ and provides \mathcal{A} with a protocol transcript corresponding to $L_{\mathcal{I}}$ as \mathcal{I} 's location and $L_{\mathcal{R},b}$ as \mathcal{R} 's location. \mathcal{A} outputs a guess b' and wins if $b = b'$. We define the advantage of the adversary \mathcal{A} in attacking the scheme \mathcal{NFP} with respect to \mathcal{R} 's location as

$$\text{Adv}_{\mathcal{NFP}, \mathcal{A}}^{\mathcal{R}} = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that \mathcal{NFP} preserves location privacy for \mathcal{R} against a passive adversary if $\text{Adv}_{\mathcal{NFP}, \mathcal{A}}^{\mathcal{R}}$ is negligible for all possible \mathcal{A} .

A protocol secure against a passive adversary in terms of the above two games gives the assurance that after observing a protocol transcript neither can the adversary distinguish between two locations of \mathcal{I} (or \mathcal{R}) nor would it be possible to decide whether the participants are actually nearby or not.

Active Adversary. The aim here is to model the case of malicious participants. In particular, we are interested in formulating the security assurance of \mathcal{I} vis-a-vis \mathcal{R} and vice versa. Since one of the participants in the protocol is now treated as the adversary, (s)he is allowed to behave arbitrarily in the protocol run.

¹ It should be understood that there are no further restrictions on \mathcal{A} 's choices for $L_{\mathcal{I},0}, L_{\mathcal{I},1}$ and $L_{\mathcal{R}}$. For example, \mathcal{A} can choose $L_{\mathcal{R}}$ to be nearby to $L_{\mathcal{I},0}$ or $L_{\mathcal{I},1}$ (or both).

Initiator's privacy wrt the responder. This is stronger than the case of \mathcal{G}_1 . Namely, we have the following:

\mathcal{G}_3 : \mathcal{A} plays the role of \mathcal{R} and chooses two distinct locations $L_0, L_1 \in \mathcal{L}$ for \mathcal{I}, \mathcal{C} picks L_b where $b \in_R \{0, 1\}$ as \mathcal{I} 's location point and executes the protocol with \mathcal{A} . \mathcal{A} 's task is to output a guess b' of b and it wins if $b = b'$. The advantage of the adversary \mathcal{A} in the role of the responder in attacking the scheme \mathcal{NFP} with respect to \mathcal{I} 's location is defined as

$$\text{Adv}_{\mathcal{NFP}, \mathcal{R}}^{\mathcal{I}} = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

We say that \mathcal{NFP} preserves location privacy for \mathcal{I} against an active adversary if $\text{Adv}_{\mathcal{NFP}, \mathcal{R}}^{\mathcal{I}}$ is negligible for all possible \mathcal{A} .

Responder's privacy wrt the initiator. Note that it is easy for a malicious \mathcal{I} to verify a guess $L'_{\mathcal{R}}$ of \mathcal{R} 's location by choosing her own location as nearby to $L'_{\mathcal{R}}$. This kind of *probing attack* cannot be prevented because of the very nature of the problem. The only safeguard for \mathcal{R} is not to participate in the protocol if he has some reason to believe that \mathcal{I} is behaving suspiciously. The aim here is rather to ensure that a malicious \mathcal{I} does not learn any information (other than what the protocol entitles her) about \mathcal{R} 's location $L_{\mathcal{R}}$ when the latter is uniformly distributed in \mathcal{L} . This is captured in the following game.

\mathcal{G}_4 : \mathcal{A} initiates the protocol and provides the challenger with a message as in a valid protocol run. In response \mathcal{C} chooses a location point $L_{\mathcal{R}}$ for \mathcal{R} uniformly at random from the set of all possible locations \mathcal{L} and completes the protocol with \mathcal{A} based on that location. \mathcal{A} 's task is to output a guess $L'_{\mathcal{R}}$ of \mathcal{R} 's location and it wins if $L'_{\mathcal{R}} \equiv L_{\mathcal{R}}$. Suppose there are (at most) k locations in \mathcal{L} which are considered as nearby to a particular location. The advantage of the adversary \mathcal{A} in the role of the initiator in attacking the scheme \mathcal{NFP} with respect to \mathcal{R} 's location is defined as

$$\text{Adv}_{\mathcal{NFP}, \mathcal{I}}^{\mathcal{R}} = \left| \Pr[L'_{\mathcal{R}} \equiv L_{\mathcal{R}}] - \frac{k+1}{|\mathcal{L}|} \right|.$$

For a justification of the threshold probability $(k+1)/|\mathcal{L}|$, see the argument for Claim 3 in Section 3. We say that \mathcal{NFP} preserves location privacy for \mathcal{R} against an active adversary if $\text{Adv}_{\mathcal{NFP}, \mathcal{I}}^{\mathcal{R}}$ is negligible for all possible \mathcal{A} .

3 The Protocol

We introduce our protocol to solve the nearby friend problem. The protocol, which we call \mathcal{NFP} -I, can also be viewed as a basic primitive to solve more complex private matching problems as we discuss later. The protocol resembles a Diffie-Hellman type of key exchange [9], and enjoys some attractive features in terms of performance and security.

3.1 Construction

A location corresponds to some point on a surface area such as the surface of the earth. In the protocol we assume that the set of locations consists of some distinct areas which are disjoint. Each such area defines a cell and each cell is identified by a unique description. This cell identifier can be the name of a city or the ZIP code of an area or some other description such as the name of an institution. For example, “Cirencester-Royal Agricultural College” (the venue of the IMA Conference on Cryptography and Coding) or “London-Royal Holloway College” can serve as the identifier of two different cells. Such a description should be fixed a priori and two parties are said to be *nearby* if and only if they belong to the same cell. Zhong has proposed a similar strategy for the Wilfrid protocol [21].

NFP-I protocol. In the protocol description, \mathcal{L} is the set of locations, G is a multiplicative group of prime order p , and g is a publicly known generator of G . More concretely, we will take G to be the group $E(\mathbb{F}_q)$ of points on a randomly selected elliptic curve E of prime order p defined over a finite field of prime order $q \approx 2^{256}$. For example, E could be the curve P-256 specified in NIST’s FIPS 186-3 standard [10]. Furthermore, we will assume that $3 \leq |\mathcal{L}| \leq 2^{40}$, and that the adversary’s running time is bounded by 2^{80} . Under these conditions, the adversary is unable to compute discrete logarithms in G . Moreover, since the discrete logarithm problem in G is random self-reducible, the adversary is also unable to solve a single instance given up to 2^{40} instances of the problem.

Before executing the protocol, the parties need to establish an authenticated channel between them. We also need an injective function f that, given the description of a cell $L \in \mathcal{L}$, maps it to a unique non-identity element of G . Such a map-to-point function can be obtained in a way as suggested in [5,6,13] in the case where G is an elliptic curve group. For such an instantiation, f behaves like a random function in the sense that the discrete logarithm of $f(L)$ to the base g is presumably hard to compute; f is treated as a random oracle in the security argument of [5,6]. We assume that f is public. The protocol is presented in Figure 1.

PUBLIC INFORMATION: A set of locations \mathcal{L} , a multiplicative group $G = \langle g \rangle$ of prime order p , and an injective function $f : \mathcal{L} \rightarrow G^*$.

INPUT: The initiator \mathcal{I} ’s input is her secret location $L_{\mathcal{I}}$ and the responder \mathcal{R} ’s input is his secret location $L_{\mathcal{R}}$.

1. \mathcal{I} computes $g_{\mathcal{I}} = f(L_{\mathcal{I}})$, chooses $\alpha \in_R \mathbb{Z}_p^*$, and computes $a = g_{\mathcal{I}}^\alpha$. \mathcal{I} sends a to \mathcal{R} .
2. Upon receiving a from \mathcal{I} , \mathcal{R} first computes $g_{\mathcal{R}} = f(L_{\mathcal{R}})$, chooses $\beta \in_R \mathbb{Z}_p^*$, and computes $b_1 = g_{\mathcal{R}}^\beta$ and $b_2 = a^\beta$. \mathcal{R} sends (b_1, b_2) to \mathcal{I} .
3. Upon receiving (b_1, b_2) from \mathcal{R} , \mathcal{I} computes $b'_2 = b_1^\alpha$. \mathcal{I} outputs **nearby** if $b'_2 = b_2$; otherwise it outputs **notnearby**.

Fig. 1. The *NFP-I* protocol

The protocol \mathcal{NFP} -I is quite efficient in terms of computation and communication bandwidth. After the execution of the protocol \mathcal{I} will find \mathcal{R} to be nearby if they belong to the same cell.

3.2 Security

We show that the protocol \mathcal{NFP} -I maintains all the security attributes defined in Section 2. In particular, \mathcal{I} 's location privacy is preserved information theoretically while that of \mathcal{R} is based on the hardness of the decision Diffie-Hellman problem for a passive adversary and a variant of it in case of an active adversary.

Claim 1. *The initiator's privacy is information theoretically secure with respect to a passive as well as an active adversary. That is, no adversary can win the game \mathcal{G}_1 or \mathcal{G}_3 with nonzero advantage even when it has unbounded computational power.*

Argument. We establish the claim wrt \mathcal{G}_1 ; the case for \mathcal{G}_3 is analogous. \mathcal{A} picks two distinct locations $L_{\mathcal{I}_0}, L_{\mathcal{I}_1}$ for \mathcal{I} and one location $L_{\mathcal{R}}$ for \mathcal{R} . Suppose that $f(L_{\mathcal{I}_0}) = g_{\mathcal{I}_0}$, $f(L_{\mathcal{I}_1}) = g_{\mathcal{I}_1}$, and $f(L_{\mathcal{R}}) = g_{\mathcal{R}}$. Suppose that \mathcal{C} selects $b = 0$; the case $b = 1$ is similar. Then \mathcal{C} provides \mathcal{A} with a transcript of the form $\langle g_{\mathcal{I}_0}^\alpha, g_{\mathcal{R}}^\beta, g_{\mathcal{I}_0}^{\alpha\beta} \rangle$ where α and β are randomly selected from \mathbb{Z}_p^* . But G is a cyclic group and hence $g_{\mathcal{I}_0} = g_{\mathcal{I}_1}^k$ for some $k \in \mathbb{Z}_p^*$. So the transcript can also be viewed as $\langle g_{\mathcal{I}_1}^{k\alpha}, g_{\mathcal{R}}^\beta, g_{\mathcal{I}_1}^{k\alpha\beta} \rangle$. Hence \mathcal{A} cannot determine whether this transcript corresponds to $(L_{\mathcal{I}_0}, L_{\mathcal{R}})$ where the randomizer for \mathcal{I} is α or to $(L_{\mathcal{I}_1}, L_{\mathcal{R}})$ where the randomizer for \mathcal{I} is $k\alpha$. \square

Recall that the decision Diffie-Hellman (DDH) problem in G is to decide, given $\langle g, g^\alpha, g^\beta, h \rangle$, whether h is equal to $g^{\alpha\beta}$, where $\alpha, \beta \in_R \mathbb{Z}_p$, and h is either $g^{\alpha\beta}$ or a random element of G . The decision Diffie-Hellman assumption in G asserts that the DDH problem is hard in G .

Claim 2. *The location privacy of \mathcal{R} is preserved with respect to a passive adversary under the decision Diffie-Hellman assumption in G where f is treated as a random oracle.*

Reductionist argument. Given an adversary \mathcal{A} with a non-negligible advantage ϵ in game \mathcal{G}_2 we show how to construct a DDH solver. The challenger is provided with a DDH problem instance $\langle g, g^\alpha, g^\beta, h \rangle$. \mathcal{A} selects one location $L_{\mathcal{I}}$ for \mathcal{I} and two distinct locations $L_{\mathcal{R}_0}$ and $L_{\mathcal{R}_1}$ for \mathcal{R} . At any point during the game \mathcal{A} can ask for an evaluation of f on any location in \mathcal{L} . For each such distinct query \mathcal{C} chooses a distinct random $x \in \mathbb{Z}_p^*$ and returns g^x as the output of f . Let's suppose $f(L_{\mathcal{I}}) = g^{x_{\mathcal{I}}}$, $f(L_{\mathcal{R}_0}) = g^{y_0}$ and $f(L_{\mathcal{R}_1}) = g^{y_1}$. \mathcal{C} chooses a random $b \in \{0, 1\}$ and returns the transcript $\langle (g^\alpha)^{x_{\mathcal{I}}}, (g^\beta)^{y_b}, h^{x_{\mathcal{I}}} \rangle$. If $h = g^{\alpha\beta}$ then this is a proper protocol transcript for $L_{\mathcal{I}}, L_{\mathcal{R}_b}$. Otherwise, the transcript is independent of b . Thus, \mathcal{A} 's probability of success in the former case will be $1/2 \pm \epsilon$ and that in the latter case will be $1/2$. Hence, \mathcal{C} has an advantage $\epsilon/2$ in solving the DDH problem. On the other hand, if DDH problem is easy in G then one can trivially

break the protocol. Hence the security of \mathcal{NFP} -I in terms of \mathcal{G}_2 is equivalent to the hardness of DDH problem in G . \square

Claim 3. *No computationally bounded malicious initiator will be able to find the location of the responder with a probability of success greater than $2/|\mathcal{L}|$.*

Heuristic argument. We first consider a probing attack. \mathcal{A} guesses some location $L_1 \in \mathcal{L}$, computes $f(L_1) \in G^*$, and sets $\tilde{g} = f(L_1)^\alpha$ for some $\alpha \in \mathbb{Z}_p^*$ as its message. As a response the challenger chooses a location L_B uniformly at random from \mathcal{L} , computes $h = f(L_B) \in G^*$, chooses $\beta \in_R \mathbb{Z}_p^*$, and provides $(b_1, b_2) = (h^\beta, \tilde{g}^\beta)$ as the message coming from the responder. If $b_1^\alpha = b_2$ \mathcal{A} returns L_1 , otherwise it returns $L_2 \in_R \mathcal{L} \setminus \{L_1\}$. The probability of success for \mathcal{A} in this game is $2/|\mathcal{L}|$. Since we cannot prevent this kind of probing attack we want the assurance that this is the best \mathcal{A} can do.

Let $S = \{h_1, h_2, \dots, h_n\}$ be the (random) subset of G which corresponds to the range of f . \mathcal{A} knows S because \mathcal{A} has access to f . Note that for any $\tilde{g} \in G^*$ and $i \in \{1, \dots, n\}$ we have $h_i = \tilde{g}^{k_i}$ for some $k_i \in \mathbb{Z}_p^*$. But the randomness property of f assures that no computationally bounded \mathcal{A} can find two distinct elements $h_i, h_j \in S$ so that \mathcal{A} knows the discrete log of both h_i, h_j to the base $\tilde{g} \in G^*$ of its choice. If \mathcal{A} can successfully find L_B given $(h^\beta, \tilde{g}^\beta)$ then that amounts to finding $h = f(L_B)$ where $L_B \in_R \mathcal{L}$ and where \tilde{g} is chosen by \mathcal{A} . Let $h = \tilde{g}^\gamma$ for some $\gamma \in \mathbb{Z}_p^*$. We consider two mutually exclusive cases: (i) \mathcal{A} knows γ and (ii) \mathcal{A} does not know γ .

(i) \mathcal{A} knows γ . In this case \mathcal{A} can easily obtain h as $h = \tilde{g}^\gamma$. But $h = f(L_B)$ is chosen from S after \mathcal{A} has output \tilde{g} . So the best \mathcal{A} can do is to use a plug-and-pray strategy – choose a random $h' \in S$ and then set $\tilde{g} = (h')^k$ for some known k . The probability that $h = h'$ in this case is $1/|\mathcal{L}|$ and this corresponds to $L_1 = L_B$ in the probing attack discussed above.

(ii) \mathcal{A} does not know γ . If \mathcal{A} can successfully predict L_B then that amounts to finding $h \in S$ given $\langle \tilde{g}^{\gamma\beta}, \tilde{g}^\beta \rangle$ where $\tilde{g} \in G$ is first chosen by \mathcal{A} after which $\beta \in_R \mathbb{Z}_p^*$ and $h = \tilde{g}^\gamma \in_R S$ are chosen by the responder. This is a hard problem when \mathcal{A} has no role in the choice of \tilde{g} and $h = \tilde{g}^\gamma$ comes from the whole group G . (This is the so-called divisible computational Diffie-Hellman (DCDH) problem where the task is to find $g^{x/y}$ given $\langle g, g^x, g^y \rangle$ for randomly selected $x, y \in \mathbb{Z}_p^*$. In [4] it has been shown that the DCDH problem is equivalent to the computational Diffie-Hellman problem.) We assume that DCDH remains hard even when the adversary is allowed to choose the base \tilde{g} and \tilde{g}^γ comes from the random subset S . So it would appear that the best strategy for \mathcal{A} is to return a random guess $h' \in S$ of h . The probability of success in this case is again $1/|\mathcal{L}|$ and this corresponds to the case $L_2 = L_B$ in the probing attack. \square

This heuristic argument can be further formalized in terms of a reduction to a *non-standard* version of the decision Diffie-Hellman problem as shown below. Abdalla and Pointcheval introduced several such interactive assumptions in [1]. These are termed by the authors as chosen-basis decisional Diffie-Hellman assumptions (CDDH1 and CDDH2) and password-based chosen-basis decisional

Diffie-Hellman assumptions (PCDDH1 and PCDDH2). They showed how one can use a PCDDH1 solver (resp. PCDDH2 solver) to solve the CDDH1 problem (resp. CDDH2 problem). They also provided lower bounds for CDDH1 and CDDH2 problem in the generic group model of Shoup [18]. However, Szydlo [19] observed that one can easily break the CDDH1 and CDDH2 problem. No such attack is yet known for the other two assumptions, namely PCDDH1 and PCDDH2.

Here we show how an adversary in terms of the game \mathcal{G}_4 against \mathcal{NFP} -I can be used to construct a PCDDH2 solver. Note that the reduction only goes in one way and it is not known whether an efficient adversary against PCDDH2 will also make \mathcal{NFP} -I vulnerable.

Password-based chosen-basis decisional Diffie-Hellman assumption (PCDDH2). Let $G = \langle g \rangle$ be a cyclic group of some prime order p , and let \mathcal{P} be a random function from $\{1, \dots, n\}$ into G . The problem is defined in terms of the following interactive game between an adversary \mathcal{B} and a challenger \mathcal{C} .

\mathcal{B} , given oracle access to \mathcal{P} , chooses two elements $g_1, g_2 \in G$ and gives them to \mathcal{C} . \mathcal{C} chooses a random $k \in \{1, \dots, n\}$, a random $r \in \mathbb{Z}_p^*$, and a random $b \in \{0, 1\}$. \mathcal{C} then obtains $u = \mathcal{P}(k)$ and computes $(g_1/u)^r$. If $b = 0$, it outputs $\langle g_2^r, (g_1/u)^r, k \rangle$, otherwise it outputs $\langle g_2^r, h, k \rangle$, where h is a random element of G . The task of \mathcal{B} is to guess the value of b . The PCDDH2 assumption asserts that this problem is hard in G .

Reductionist argument. Given an adversary \mathcal{A} with a non-negligible advantage against \mathcal{NFP} -I in \mathcal{G}_4 , we show how one can construct an algorithm \mathcal{B} to solve the PCDDH2 problem. \mathcal{B} sets $|\mathcal{L}| = n$ and then uses \mathcal{P} to simulate f . When \mathcal{A} sends some \tilde{g} as a message from the initiator it sets $g_1 = \mathbf{1}$, where $\mathbf{1}$ is the identity element of G and $g_2 = \tilde{g}$ and sends them to \mathcal{C} . When it receives the challenge $\langle g_3, g_4, k \rangle$ from \mathcal{C} , \mathcal{B} sends $\langle g_4^{-1}, g_3 \rangle$ as the corresponding message in \mathcal{NFP} -I. If $g_4 = (g_1/u)^r$ then this corresponds to $\langle u^r, \tilde{g}^r \rangle$, which is a valid protocol transcript for a location L_B such that $f(L_B) = \mathcal{P}(k)$. Otherwise the transcript is independent of L_B . Hence, any advantage of \mathcal{A} in finding L_B can be converted to an advantage of \mathcal{B} to solve the PCDDH2 problem. \square

4 Extensions

In this section we consider several extensions of the basic primitive. So far we have treated locations as detached cells and two parties are considered *nearby* if and only if they are in the same cell. This is perfectly acceptable for some applications where cells are few and far between (say the large cities of the world). In such a scenario if the protocol informs \mathcal{I} that \mathcal{R} is *not nearby* then with overwhelming probability they are not in geographically close locations. However, this might not be the case if cells are dense. As an illustrative example consider the following situation. \mathcal{I} 's location identifier is “London-Royal Holloway College-Founder's Building” and that of \mathcal{R} is “London-Royal Holloway College-Windsor Building”. Since these two buildings are represented as different cells, \mathcal{NFP} -I will return them as *not nearby* even though they are physically quite close.

One way to model such scenarios is to allow a hierarchy of cells, where cells in the lowest level represent smallest areas (say a building), which are contained in the next level cells (say a street), and so on. Such a granular description was first proposed by Zhong in [21]. Returning to our previous example, using a two-level description \mathcal{I} 's location will be $\{\text{"London-Royal Holloway College"}, \text{"London-Royal Holloway College-Founder's Building"}\}$ and that of \mathcal{R} $\{\text{"London-Royal Holloway College"}, \text{"London-Royal Holloway College-Windsor Building"}\}$. \mathcal{I} and \mathcal{R} may first run the protocol on the higher-level set and if they are in the same cell then on the lower level. However, they can easily send all the information in a single run.

We have described the situation with $\ell = 2$ level hierarchy. However it is easy to extend it to any $\ell > 2$ depending upon the application. The modified version maintains all the security attributes of the original protocol because both \mathcal{I} and \mathcal{R} use different randomizers for different levels. The protocol will also maintain its better performance over Wilfrid.

For still some other applications it might be more useful to consider a contiguous area. Suppose \mathcal{I} is at some geographical location on the surface of the earth and is interested to know whether \mathcal{R} is within a disc of radius d centered at her own location. In other words, \mathcal{I} should be able to find \mathcal{R} as *nearby* as long as the Euclidean distance between the two does not exceed d .

The Lester protocol in [22] provides a solution to this problem. However, the solution is not fully satisfactory. To determine whether \mathcal{R} is within a distance d , \mathcal{I} needs to solve a discrete log problem in the range of $[0, d \cdot 2^t]$, where t is some safety factor chosen by \mathcal{R} . On the other side, \mathcal{R} has no knowledge (and control) on d . In fact, the protocol allows \mathcal{I} to extract some information about \mathcal{R} 's location even when the distance between them is greater than d . For example, with twice (resp. three times) as much work \mathcal{I} can determine whether \mathcal{R} is within a distance $2d$ (resp. $3d$).

In the Pierre protocol [22], instead of considering a disc around \mathcal{I} , a contiguous surface area is divided into square grid cells of side length d and the coordinates of each party is expressed in integral units of d . After execution of the protocol, \mathcal{I} will find \mathcal{R} if they are in the same cell or \mathcal{R} is in one of the eight adjacent cells.

In the following we use a slightly different discretization technique. Instead of using square grid cells we tile the surface by regular hexagons (see Figure 2). This gives a better approximation of a circle around \mathcal{I} 's location. We note that Pierre will also benefit from this kind of discretization strategy.

Assume that the locations are expressed in integer coordinates (X, Y) , with the positive X and Y axes making an angle of 60 degrees and each hexagonal cell has side length d . With this representation, the centers of the cells will be at the points (dx, dy) with x, y integers and $x - y \equiv 0 \pmod{3}$. Given a location (X_A, Y_A) we can find the center of the corresponding cell (x_A, y_A) as follows. Let $x_0 = \lfloor X_A/d \rfloor$ and $y_0 = \lfloor Y_A/d \rfloor$. If $x_0 - y_0 \equiv 1 \pmod{3}$ then $(x_A, y_A) = (x_0, y_0 + 1)$; else if $x_0 - y_0 \equiv 2 \pmod{3}$ then $(x_A, y_A) = (x_0 + 1, y_0)$; else if $X_A + Y_A \leq d(x_0 + y_0 + 1)$ then $(x_A, y_A) = (x_0, y_0)$; else $(x_A, y_A) = (x_0 + 1, y_0 + 1)$.

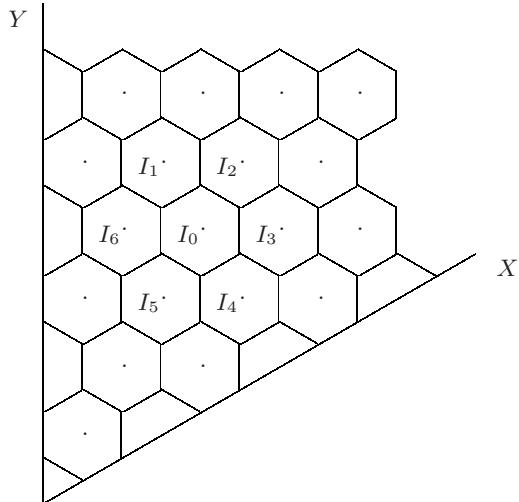


Fig. 2. Discretization of the surface into hexagonal cells

In the protocol, \mathcal{I} first finds the center of the cell she is in (labelled I_0) and then the center of the six adjacent cells (labelled I_1, \dots, I_6 as shown in Figure 2). After executing the protocol \mathcal{I} will find \mathcal{R} to be nearby if and only if he is in one of these seven cells. She will also learn exactly in which cell he is. Let \mathcal{M} be the set of the hexagonal cells where each cell is identified by its center point. Note that \mathcal{M} depends upon d and \mathcal{I} and \mathcal{R} must agree upon this value prior to running the protocol.

Let G be a multiplicative group of prime order p and let g be a publicly known generator of G . In the protocol description, $f : \mathcal{M} \rightarrow G^*$ is an injective function that takes as input a center point of a cell and outputs a random element in G^* . Such a map-to-point function can be constructed in an analogous way as described in Section 3.1. Before executing the protocol, the parties establish an authenticated channel between them. The protocol is presented in Figure 3.

Security. In the protocol $\mathcal{NFP}\text{-II}$, \mathcal{I} uses a separate randomizer α_j for each I_j , hence the argument in Claim 1 for $\mathcal{NFP}\text{-I}$ can be trivially extended to $\mathcal{NFP}\text{-II}$. So the location privacy of the initiator is information theoretically preserved with respect to a passive as well as an active adversary. Similarly, for a passive adversary we can extend the argument in Claim 2 to show that breaking the location privacy of \mathcal{R} is equivalent to solving the DDH problem in G . Thus $\mathcal{NFP}\text{-II}$ provides the same assurances as $\mathcal{NFP}\text{-I}$ in terms of the security games $\mathcal{G}_1, \mathcal{G}_2$ and \mathcal{G}_3 .

However, the situation is a little different when we consider the location privacy of the responder with respect to a malicious initiator. \mathcal{I} can now deviate from the protocol and choose seven different cells (not necessarily adjacent) as her input. If \mathcal{R} happens to be in one of these cells then \mathcal{I} can easily detect that in the verification step (i.e., Step 3) of $\mathcal{NFP}\text{-II}$. If not it returns a random cell

PUBLIC INFORMATION: A description of the set of locations \mathcal{M} , a multiplicative group $G = \langle g \rangle$ of prime order p , and an injective function $f : \mathcal{M} \rightarrow G^*$.

INPUT: The initiator \mathcal{I} 's input is her secret cell and the six adjacent cells $L_{\mathcal{I}} = (I_0, I_1, I_2, I_3, I_4, I_5, I_6)$ and the responder \mathcal{R} 's input is his secret cell $L_{\mathcal{R}}$.

1. For $0 \leq j \leq 6$, \mathcal{I} computes $g_{\mathcal{I},j} = f(I_j)$, chooses $\alpha_j \in_R \mathbb{Z}_p^*$, and computes $a_j = g_{\mathcal{I},j}^{\alpha_j}$. \mathcal{I} sends $(a_0, a_1, a_2, a_3, a_4, a_5, a_6)$ to \mathcal{R} .
2. Upon receiving $(a_0, a_1, a_2, a_3, a_4, a_5, a_6)$ from \mathcal{I} , \mathcal{R} first computes $g_{\mathcal{R}} = f(L_{\mathcal{R}})$, chooses $\beta \in_R \mathbb{Z}_p^*$, and then computes $b = g_{\mathcal{R}}^\beta$ and $h_j = a_j^\beta$ for $0 \leq j \leq 6$. \mathcal{R} sends to \mathcal{I} the tuple $(b, h_0, h_1, h_2, h_3, h_4, h_5, h_6)$.
3. Upon receiving $(b, h_0, h_1, h_2, h_3, h_4, h_5, h_6)$ from \mathcal{R} , \mathcal{I} computes $h'_j = b^{\alpha_j}$ for $0 \leq j \leq 6$. \mathcal{I} outputs **nearby** if $h_j = h'_j$ for some $0 \leq j \leq 6$. Otherwise it outputs **notnearby**.

Fig. 3. The \mathcal{NFP} -II protocol

from the rest. So its probability of success in the probing attack will be $8/|\mathcal{L}|$ (it is $2/|\mathcal{L}|$ in \mathcal{NFP} -I). In Claim 3 we argued that the probing attack is the best strategy for a malicious initiator in \mathcal{NFP} -I. A similar argument can be put forth for \mathcal{NFP} -II also based on related but stronger complexity assumptions.

Performance. In \mathcal{NFP} -II, \mathcal{I} performs seven group exponentiations to compute her message while \mathcal{R} performs eight exponentiations to generate his message. \mathcal{I} needs at most seven more exponentiations to decide whether \mathcal{R} is nearby or not. The message from \mathcal{I} to \mathcal{R} consists of seven elements of G and that from \mathcal{R} to \mathcal{I} consists of eight elements of G . After executing the protocol \mathcal{I} will find \mathcal{R} to be nearby if they belong to the same cell or if \mathcal{R} is in one of the six cells adjacent to \mathcal{I} 's. \mathcal{I} will also learn that \mathcal{R} is in the j th cell wrt \mathcal{I} if $h'_j = h_j$, $0 \leq j \leq 6$.

In Table 2 we give a rough performance comparison of Pierre, the Pierre protocol modified to use hexagonal tiling (which we call Pierre*), Wilfrid using hexagonal tiling, and \mathcal{NFP} -II. We assume that the protocols are implemented in a multiplicative group in the discrete log setting. We only count exponentiation in the underlying group G as that is the most computationally intensive part. All the protocols require two communication steps and the messages are elements of G . Note that Pierre and Wilfrid both require that the parties establish a communication channel that is both confidential and authenticated prior to running the protocol and uses homomorphic encryption, while \mathcal{NFP} -II only requires an authenticated channel and does not use any other cryptographic primitive.

In terms of overall performance, Pierre is currently the best choice in this case, while \mathcal{NFP} -II is marginally better than Wilfrid. In \mathcal{NFP} -II we can further reduce the message size from \mathcal{R} to \mathcal{I} by individually hashing the group elements h_0, \dots, h_6 using a standard hash function and sending the hash digests. Such an optimization is not possible for the other protocols as the messages consist of ciphertexts of the underlying homomorphic encryption scheme.

Table 2. Performance comparison between Pierre, Wilfrid and \mathcal{NFP} -II

	$\mathcal{I} \rightarrow \mathcal{R}$		$\mathcal{R} \rightarrow \mathcal{I}$		decide whether nearby: # exps by \mathcal{I}
	# exps	message size	# exps	message size	
Pierre	6	6	8	6	≤ 3
Pierre*	6	6	6	4	≤ 2
Wilfrid	16	16	5	2	1
\mathcal{NFP} -II	7	7	8	8	≤ 7

Remark 1. In Pierre, a malicious responder can always fool the initiator into believing that they are in nearby locations. This may not be a problem as long as \mathcal{I} can physically check that \mathcal{R} is not actually in the location where she is supposed to be. Nor is this an issue for the security of \mathcal{I} as \mathcal{R} cannot gain any information about her location. In fact our security model does not account for this kind of dishonest responder. However, our protocol and Wilfrid appear to resist such behavior.

5 Concluding Remarks

We formulated a definition and security model of the nearby friend problem in the context of location-based services. We proposed a protocol that efficiently solves the problem in the proposed security model. Our approach is distinct from the previous solutions to this problem as it does not depend on any other cryptographic primitive and requires only an authenticated link. We believe that the basic primitive we proposed will be useful in other contexts.

The nearby friend problem can be viewed as a concrete instantiation of the more abstract private matching problem. In the latter a client having secret input set X interacts with a server having secret input set Y to determine $X \cap Y$. This problem has been solved earlier using homomorphic encryption and the solution further specialized for the nearby friend problem.

As in the nearby friend problem, our basic primitive may act as a distinct (and perhaps efficient) alternative to the homomorphic encryption based approach to private matching. For example, \mathcal{NFP} -I can be used to determine whether two singleton sets are equal or not. Similarly, its hierarchical version discussed in Section 4 can be used to find the intersection of two ordered lists. It would be interesting to investigate whether this approach can be extended to solve the general private matching problem or whether it is applicable in the multi-party setting.

Acknowledgment

We thank David Wagner (University of Waterloo) for suggesting the discretization strategy used in Protocol \mathcal{NFP} -II and the anonymous reviewers of IMACC 2009 for their comments.

References

1. Abdalla, M., Pointcheval, D.: Interactive Diffie-Hellman assumptions with applications to password-based authentication. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 341–356. Springer, Heidelberg (2005)
2. Amir, A., Efrat, A., Myllymaki, J., Palaniappan, L., Wampler, K.: Buddy tracking - efficient proximity detection among mobile friends. *Pervasive and Mobile Computing* 3, 489–511 (2007)
3. Atallah, M., Du, W.: Secure multi-party computation problems and their applications: a review and open problems. In: Proceedings of the 2001 Workshop on New Security Paradigms – NSPW 2001. Association for Computing Machinery, pp. 13–22 (2001)
4. Bao, F., Deng, R., Zhu, H.: Variations of Diffie-Hellman Problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (2003)
5. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 586–615. Springer, Heidelberg (2001)
6. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 297–319. Springer, Heidelberg (2001)
7. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
8. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol, Technical report, Version 1.1 1483 RFC 4346, Internet Engineering Task Force (2006), <http://www.ietf.org/rfc/rfc4346.txt>
9. Diffie, W., Hellman, M.: New directions in cryptography. *IEEE Transaction on Information Theory* 22, 644–654 (1976)
10. FIPS 186-3, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology (2009)
11. Freedman, M., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
12. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: ACM Symposium on Theory of Computing – STOC 1982, Association for Computing Machinery, pp. 365–377 (1982)
13. Icart, T.: How to hash on elliptic curves. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 303–316. Springer, Heidelberg (2009)
14. Jablon, D.: Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review* 26, 5–26 (1996)
15. Jakobsson, M., Yung, M.: Proving without knowing: On oblivious, agnostic and blindfolded provers. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 186–200. Springer, Heidelberg (1996)
16. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
17. Køien, G., Oleshchuk, V.: Location privacy for cellular systems; analysis and solution. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 40–58. Springer, Heidelberg (2006)

18. 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)
19. Szydlo, M.: A note on chosen-basis decisional Diffie-Hellman Assumptions. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 166–170. Springer, Heidelberg (2006)
20. Wang, S., Min, J., Yi, B.: Location based services for mobiles: Technologies and standards. In: IEEE International Conference on Communication – ICC (2008)
21. Zhong, G.: Distributed approaches for location privacy, Masters Thesis, University of Waterloo (2008)
22. Zhong, G., Goldberg, I., Hengartner, U.: Louis, Lester and Pierre: Three protocols for location privacy. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 62–76. Springer, Heidelberg (2007)

Distributing the Key Distribution Centre in Sakai–Kasahara Based Systems

Martin Geisler¹ and Nigel P. Smart²

¹ Dept. of Computer Science,
University of Aarhus,
IT-parken, Aabogade 34,
DK-8200 Aarhus N,
Denmark
mg@cs.au.dk

² Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom
nigel@cs.bris.ac.uk

Abstract. One major drawback with using Sakai–Kasahara based identity-based secret keys is that it appears hard to distribute the trust of the key generation centre. For other types of identity based key this distribution can be done using non-interactive and simple techniques. In this short note we explain how this can be done for Sakai–Kasahara style keys using a simple application of a general technique from multi-party computation. The self-checking property of the resulting private key we show can be used to insulate the protocol from malicious adversaries for many practical parameter sizes.

1 Introduction

Distributing the trust placed in trusted third parties is an important problem in the real world, for example one can distribute the signing key for a certificate authority. This is not only an important issue from the point of view of security, but it also helps maintain resilience against network outages or to allow a form of disaster planning. Many of the techniques in this area are based on the method of threshold secret sharing introduced by Shamir in [20]. The techniques used can be applied in many discrete logarithm based situations, and have even been applied to the more complex area of RSA style signature generation [22]. In 1984, Shamir [21] also invented a form of cryptography based on identities as opposed to public keys. There is a similar need for the trusted centre to be distributed in IBE schemes, just as the CA should be in certificate based systems.

Currently, there are three main types of identity based secret key in use, all of which are based on pairings on elliptic curves, namely *full-domain hash*, *commutative blinding* and *exponent inversion*. A classification which was first proposed

in [9]. The first set of *full-domain hash* style keys, is typified by the systems of Boneh–Franklin [8] and Sakai et al [17]. In this construction distribution of the key generation centre is easily performed using many of the ideas which have been used in the traditional discrete logarithm based public key setting. The second set, denoted by the term *commutative blinding*, is typified by the Boneh–Boyen encryption scheme [6] (BB-1). Here modifications can be applied to the techniques applied in the full-domain hash case to obtain a distributed key generation centre based on threshold cryptography [7].

The third type of keys are those introduced by Sakai and Kasahara in [18], which are often referred to as those of *exponent inversion* type. These provide the most efficient, in the random oracle model, ID-based encryption scheme known. A basic ID-based scheme was using this concept was proved secure in [11], then a more efficient and conceptually simpler hybrid ID-based encryption scheme was presented in [12]. This key construction can also be used to construct identity based signatures and signcryption [2], and a related construction (also of exponent inversion type) underlies the second Boneh–Boyen encryption scheme (BB-2). Both the schemes in [2] and [12] are in the current IEEE 1363.3 draft.

The problem that this paper investigates is that the exponent inversion type keys currently have no efficient distributed key generation solution. In this paper we present a solution, which requires for each ID-based key a secure multi-party computation to be performed amongst the servers. This is not ideal, for the other types of ID-based keys one does not require a complex distributed key extraction protocol in the distributed generation setting. However, the protocol we require is very simple and requires only one execution of a distributed distributed multiplication protocol modulo q , where q is the size of the underlying groups, followed by each server executing some simple local group computations. We focus on the Sakai–Kasahara type keys, since these are the most relevant for practice (due to the above mentioned standardisation effort), however we will also note the changes which are needed for the BB-2 type keys as well.

The multi-party computation protocol we shall use is in the information theoretic model. The basic ideas in this setting can be traced back to the papers of Ben-Or et al [4] and Chaum et al [10]. The protocol we shall use, using Shamir secret sharing and the evaluation of what is in effect a trivial arithmetic circuit, was presented in [16] and then extended to any linear secret sharing scheme in [13]. The simple technique for deriving modular inverses in this setting, which we use, was first introduced in [1]. The exact protocol details we propose to use can be found explicitly described in the survey by Cramer et al [14]. We present some experimental data on our protocol which makes use of the VIFF system [24].

We end this introduction by pointing out that our protocol is neither deep, nor very high-tech. It is simply applying a known technique from multi-party computation to the problem situation, and then noticing that the resulting protocol is not as inefficient as one usually obtains for multi-party computation protocols for real world problems. What is surprising is that no-one has noticed this before in the community looking at Sakai–Kasahara based pairing protocols.

2 Notation and Problem Statement

Let $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T denote groups of large prime order q , which are equipped with a bilinear pairing,

$$\hat{t}: \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T.$$

We assume that \mathbb{G}_1 and \mathbb{G}_2 are generated by P_1 and P_2 . We will write \mathbb{G}_1 and \mathbb{G}_2 additively and \mathbb{G}_T multiplicatively.

We assume there is some global master secret x and we define master public key for the scheme by $R = [x]P_1$. The user secret key extraction algorithm for Sakai–Kasahara keys is to represent an identity ID as an element in \mathbb{F}_q and then compute the public key Q_{ID} and the private key S_{ID} via the equations

$$\begin{aligned} Q_{\text{ID}} &= R + [\text{ID}]P_1, \\ S_{\text{ID}} &= \left[\frac{1}{x + \text{ID}} \right] P_2. \end{aligned}$$

The problem with constructing keys as above is that there is a single point of failure, in that the person who holds the master secret key x is able to compute all secret keys. Hence, any compromise of the computer which holds x leads to a break of the entire system. A standard solution to this problem is to share x amongst a number of computers in such a way that no-one computer can recover x , but that a coalition of the computers can still compute S_{ID} . It is how this calculation is performed that we treat in this paper.

We first define explicitly the parameters of our problem. We let $1 < t \leq m$ denote integers; the value m denotes the number of computers which will share the value of x , whilst t denotes the threshold value, i.e., the maximum number of colluding parties the protocol can tolerate without compromising the privacy. On the other hand $t + 1$ colluding computers will be able to recover the secret x . We will develop a protocol, in the honest-but-curious case, that succeeds as long as more than $2t$ computers are honest-but-curious. For the malicious case we require that more than $3t$ parties take part in the computation. The number of parties which take part in an interaction will be denoted by n , hence $n \leq m$.

We need to define the following three algorithms, each of which take as input the group descriptions $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.

Setup(): This is a distributed protocol which runs between the m servers and results in each server obtaining a share x_i of the master secret x . The public output of this protocol is the value $R = [x]P_1$. Note, that a coalition of upto t entities should not be able to determine any information about x .

Extract(ID): This is a distributed protocol run between a set of n servers, where $t \leq n \leq m$. It produces n outputs $S_{\text{ID}}^{(i)}$ which are the shares of the secret value S_{ID} . In the case of honest-but-curious parties we require that $n > 2t$, whilst in the case of malicious parties we require that $n > 3t$.

Combine($S_{\text{ID}}^{(1)}, \dots, S_{\text{ID}}^{(n)}$): This is algorithm is run by the user. It takes the shares produced by the Extract(ID) protocol and produces the valid user secret key S_{ID} .

We assume without loss of generality that the n servers returning a value of $S_{\text{ID}}^{(i)}$, are numbered 1 to n .

The goal in the case of honest-but-curious adversaries is that as long as the number of colluding servers is less than or equal to t , then no information leaks about the underlying master secret x due to any run of the protocol. In the case of malicious adversaries we also wish to ensure that colluding malicious adversaries, not only cannot learn any extra information, they should also not be able to make the user accept an invalid secret key in the combine stage.

One should note that, unlike the Boneh-Franklin or Boneh-Boyen key situation, the Extract(ID) method for Sakai–Kasahara keys which we present is a relatively complicated protocol and not a simple algorithm run independently by each server. We leave it as an open problem to construct a more efficient method to perform the distributed Extract(ID) method.

3 Basic Protocol

In this section we detail exactly how the three protocols above are executed in the case of honest-but-curious adversaries. Our protocol makes use of the t -out-of- n secret sharing scheme of Shamir [20], and the multi-party computation protocol described in [13,14].

Setup(): Using the property that Shamir secret sharing is a PRSS (Pseudo-Random Secret Sharing) scheme the m servers produce a t -out-of- m secret sharing of a random number x modulo q . Underlying this sharing is a polynomial $Q(X)$ modulo q of degree t and each server i obtains the value $x_i = Q(i)$. The servers compute the value $R_i = [x_i]P_1$, and then the master public key is computed by

$$R = \sum_{i=1}^m [c_i]R_i$$

where

$$c_i = \prod_{j=1, j \neq i}^m \frac{-j}{i-j}.$$

Extract(ID): Assume $n > 2t$ servers are active, they execute the following protocol which applies a technique of [1] to our problem.

- Locally the servers compute a sharing (z_i) of the value of $z = x + \text{ID}$. This can be computed by simply adding the shares (x_i) of x with the public constant ID.
- Again using the PRSS the servers obtain a sharing (r_i) of a random integer r .
- Using a single invocation of the honest-but-curious multiplication protocol from [13,14] the servers compute a sharing (s_i) of $s = z \cdot r \pmod{q}$.
- The servers recover s by revealing their shares s_i .
- Locally the servers then compute $w_i = r_i/s \pmod{q}$.
- The servers locally compute $S_{\text{ID}}^{(i)} = [w_i]P_2$, and send this value securely to the requesting user. The servers then terminate.

Note that the values w_i computed by the servers are a sharing of the $w = 1/(x + \text{ID}) \pmod q$. Note, that we cannot reveal w_i directly since that would allow the parties to recover x , hence w is revealed indirectly via the values of $S_{\text{ID}}^{(i)}$. This does not cause a problem due to our **Combine** method.

Combine($S_{\text{ID}}^{(1)}, \dots, S_{\text{ID}}^{(n)}$): The user secret key is then recovered via

$$S_{\text{ID}} = \sum_{i=1}^n [c'_i] S_{\text{ID}}^{(i)}$$

where

$$c'_i = \prod_{j=1, j \neq i}^n \frac{-j}{i-j}.$$

The above protocol assumes, due to the multiplication protocol used in the Extract(ID) stage, that $2t + 1 \leq n$. So for the minimal value of $t = 1$, we have that n must be at least three. From general results on multi-party computation such a bound is the best possible [4,10].

3.1 Security Analysis

We argue that the above protocol provides a secure distributed key generation process in the presence of honest-but-curious adversaries. Firstly, note that the distribution of keys (both master keys and user secret keys) is identical to the situation where we have a single authority. Secondly, note that the underlying multi-party computation protocol is information theoretically secure against honest-but-curious adversaries (assuming at most t dishonest servers).

Each user in the combine stage obtains their secret key, yet the shares they obtain of this key reveal no other information, due to the properties of the underlying secret sharing scheme. To see this we note there is a trivial reduction from a the view of the user in the distributed key generation situation to one in the non-distributed key generation situation: In the distributed game the simulator simply extracts the private key S_{ID} from the non-distributed game, and then generates at random a Shamir secret sharing (s_i) of zero. The random shares in the distributed game are then given by

$$S_{\text{ID}}^{(i)} = [s_i] P_2 + S_{\text{ID}}.$$

Note that the discrete logarithms of the $S_{\text{ID}}^{(i)}$ form a random Shamir secret sharing of the discrete logarithm of S_{ID} , thus the simulation is perfect.

4 Security against Malicious Adversaries

In moving to the case of malicious adversaries one needs to deal with two issues. Firstly the servers could behave maliciously in the multiplication protocol step. Secondly, they may carry out this step correctly, yet attempt to subvert the key

generation protocol by passing an incorrect share back to the user. We deal with these two issues in two distinct ways.

The first issue simply requires that we replace the honest-but-curious multiplication protocol with one which is secure against malicious adversaries. This requires that we must have $n > 3t$, and the number of honest participants is equal to at most $n - t$, (since $t + 1$ dishonest participants can trivially break the protocol). This ensures that at least $n - t$ parties at the end of the protocol run obtain a valid sharing of the w . All parties then produce $S_{\text{ID}}^{(i)} = [w_i]P_2$ as before. Except, we are only guaranteed that $n - t$ of these values are correct, since up to only at least $n - t$ parties are honest and hence will validly follow the protocol, this is the our second issue.

Solving this seems to pose a major stumbling block, however, we side-step this issue by using the fact that Sakai–Kasahara keys are self checking. Namely, for a valid pair of public/private keys $(Q_{\text{ID}}, S_{\text{ID}})$ we have that

$$\hat{t}(Q_{\text{ID}}, S_{\text{ID}}) = \hat{t}(P_1, P_2).$$

We also use that fact that *in practice* the values of t and n will not be that large, typical examples could indeed be $t = 1$ and $n = 4$. We thus present a different **Combine** algorithm for our protocol which runs in time $O(^n C_{n-t})$. This is clearly inefficient for large values of n , yet for the values one could consider in a practical application it is very manageable.

Hence, we can execute the following version of the **Combine** algorithm. The combiner goes through each of the ${}^n C_{n-t}$ subsets of $\{S_{\text{ID}}^{(1)}, \dots, S_{\text{ID}}^{(l)}\}$ of size $n - t$, and performs the recombination step with this subset only. The recombined key is then verified to be correct. If it is correct then we terminate, otherwise the next subset of size $n - t$ is taken. As long as at least $n - t$ of the servers are honest the above protocol will output the correct secret key.

If we examine the situation where we keep n to be as small as possible, i.e. $n = 3t + 1$, then the workload of this step is equal to

$$W_t = \frac{(3t + 1)!}{(2t + 1)! \cdot t!}$$

applications of the combining algorithm. For small values of t we find that this work-effort is given by the values in the following table.

t	1	2	3	4	5
W_t	4	21	120	715	4368

We note that this operation is not performed by the servers, but is performed by the client on receipt of the shares from the servers. Hence, the increased time may not be considered too much of a burden as it is only performed once per secret key request, and is performed by the requestor. Additionally, the requestor can trade space for time when recombining different $S_{\text{ID}}^{(i)}$ subsets. Firstly, the $[c'_i]S_{\text{ID}}^{(i)}$ values can computed only once and reused, and secondly, having computed S_{ID} from the subset $\{S_{\text{ID}}^{(1)}, \dots, S_{\text{ID}}^{(l)}\}$ it is easy to compute S_{ID} from $\{S_{\text{ID}}^{(2)}, \dots, S_{\text{ID}}^{(l+1)}\}$

by subtracting $[c'_1]S_{ID}^{(1)}$ and adding $[c'_{l+1}]S_{ID}^{(l+1)}$. All subsets can be computed in this fashion using only a field subtraction and addition for each subset.

We note that whilst this might not be a polynomial time solution, it is very efficient for the type of values which would be used in practice. Also it is protecting against a denial-of-service attack by the malicious servers against the clients, as opposed to the malicious servers trying to recover the underlying master secret key. We also note that most key centre distribution techniques for certificate based schemes are only secure in the honest-but-curious model, since key distribution is used more for resilience and the servers are themselves heavily protected.

5 Implementation Results

The question arises as to whether the above protocol will be suitable in a practical situation. A key generation centre for an identity based cryptographic system will need to produce a large number of keys for distinct identities ID. Hence, if the protocol is too slow then the protocol will not be suitable in real life situation.

Recently a number of practical systems for performing secure multi-party computation have been developed. Of specific interest in our situation if the VIFF (Virtual Ideal Functionality Framework) system [24,15], which itself grew out of the SIMAP [23] and SCET [19] systems. The SIMAP system has been used very successfully in practice, most notably for the Danish sugar beat auction [5].

VIFF is a fully asynchronous framework for specifying secure multi-party computations. It is implemented as a Python library. Each player executes a Python program, and the programs communicate using standard SSL connections. VIFF provides the communication infrastructure and computations primitives.

The implementation of the basic protocol from Section 3 is shown below as a function called `extract`. The function references a global variable `runtime`, which is an instance of the central `Runtime` class provided by VIFF. Among other things, this class provides methods for generating Shamir shares using PRSS (`prss.share_random`) and for reconstructing such shares (`open`). Due to extensive use of operator overloading, the addition ($x_i + ID$) and the multiplication ($z_i * r_i$) are in fact done on secret shared values, the latter invoking a secure multiplication protocol.

```
def extract(ID, x_i):
    z_i = x_i + ID
    r_i = runtime.prss.share_random(Zq)
    s_i = z_i * r_i
    s = runtime.open(s_i)
    w_i = gather_shares([r_i, s])
    w_i.addCallback(lambda (r_i, s): r_i / s)
return w_i
```

After opening s_i the players wait until r_i and s are ready. The result w_i is prepared and an anonymous function is added to a list of callbacks; this function is executed to make the local division.

The Zq variable represents a finite field with a 256 bit prime modulus. The modulus used is

$$q = 0x80AE401A5230143EFC6ADD72979CAEADB078F3F2EFD3EE07C901B94BC45C61F5.$$

This is the group order of a Barreto-Naehrig curve [3], and is typical of a group order which would be used in practice.

Using a value of $t = 1$, the minimum value of n for the honest-but-curious case is $n = 3$, whilst the minimum value of n for the malicious case is $n = 4$. We found the following run-times for the above protocol. We ignore the time for the point multiplication step of the protocol as this is done outside the VIFF system and is essentially the cost of a non-distributed key generation centre. Hence, the following times denote the extra cost required for performing distributed key generation as opposed to centralised generation. All times are given in milli-seconds and count *wall-time*, hence it includes the time needed for data transmission etc.

Semi-Honest $n = 3$	Semi-Honest $n = 4$	Malicious $n = 4$
3 ms	4 ms	6 ms

The machines used in the benchmarks were equipped with hyper-threaded Intel Xeon CPUs (3.06 GHz clock speed) and 1 GiB of RAM. They were connected with a fast LAN, which is not typical of a *real world* installation but does give a lower bound on the resources required. Our computation time does not include the final local elliptic curve point multiplication. Hence the times represent the minimum *extra* time per key generation that are required by the servers to deal with a key generation request, compared with the case where a single centralised server is used.

All results are the average time for one inversion when executing a bulk computation with 100 inversions in parallel. Doing many operations in parallel is important in order to minimise the impact of network latencies, otherwise the latency will dominate the time. If only a single inversion is done, the time per inversion roughly doubles.

As can be seen from our timings the overall extra time needed to compute each identity based secret key is relatively small and would be easily accommodated in a deployed system, for example one could easily accommodate upto one million key extraction requests per day. We conclude that distributed key generation for Sakai-Kasahara style keys requires a relatively straight forward application of standard techniques from multi-party computation.

6 Extensions

We end the paper by outlining two extensions to the methods introduced in this paper.

6.1 Interactive Threshold Signature Scheme

We note that our techniques allow one to distribute the signing phase of the Zhang et al. signature scheme [25]. The public key of this scheme is $R = xP_2$, and to sign a message one computes

$$S = \left[\frac{1}{x + H(m)} \right] P_1$$

for some hash function $H : \{0, 1\}^* \rightarrow \mathbb{F}_q$. Verification is performed by checking whether

$$\hat{t}(S, R + [H(m)]P_2) = \hat{t}(P_1, P_2).$$

Note, that we have reversed the groups in this scheme compared to the Sakai–Kasahara scheme so as to obtain shorter signatures. Each signature is thus given by an element of \mathbb{G}_1 , and verification requires only one pairing computation. It is clear that our techniques for distributed key generation of Sakai–Kasahara keys also apply to distributing the signing operation for this signature scheme.

6.2 Distributed BB-2 Key Generation

Our techniques apply to many exponent-inversion key generation procedures, the most notable other system in this family is the BB-2 scheme from [6]. The BB-2 key generation procedure is as follows: For a master public key of $(P_1, R_x, R_y) = (P_1, xP_1, yP_1)$, with master secret key $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$, the key extraction phase is given by $S_{\text{ID}} = (r, K_{\text{ID}})$, where $r \in \mathbb{F}_q$ is chosen at random and

$$K_{\text{ID}} = \left[\frac{1}{\text{ID} + x + r \cdot y} \right] P_2.$$

We note that our techniques easily extend to this situation by the following distributed extraction algorithm:

- Using the PRSS property the servers obtain a sharing (r_i) of a random integer r .
- The servers compute a sharing (s_i) of $s = r \cdot y$ using the distributed multiplication protocol.
- The servers locally compute a sharing (z_i) of $z = \text{ID} + x + s$.
- The servers then invert z using the inversion method of [1], to obtain a sharing (w_i) .
- The servers then output (r_i) and $[w_i]P_2$.

This clearly requires two multiplication protocols, as opposed to one for the Sakai–Kasahara keys. The requesting user can recover r from (r_i) using the standard recovery method for Shamir secret sharing, and the value of K_{ID} from $[w_i]P_2$ via the method described previously. Note, that BB-2 keys are also self-checking since we must have

$$\hat{t}([\text{ID}]P_1 + R_x + [r]R_y, K_{\text{ID}}) = \hat{t}(P_1, P_2).$$

Hence, our self-checking procedure for preventing malicious servers responding with invalid keys will still apply.

Acknowledgements

The authors would like to thank Andy Dancer of Trend Micro Ltd for suggesting the problem, and Dario Catalano for providing a pointer to the literature. The authors would like to thank the EU FP7 project CACE. The second author was supported by a Royal Society Wolfson Merit Award.

References

1. Bar-Ilan, J., Beaver, D.: Non-cryptographic fault-tolerant computing in a constant number of rounds on interaction. In: Principles of Distributed Computing – PODC 1989, pp. 201–209. ACM Press, New York (1989)
2. Barreto, P.S.L.M., Libert, B., McCullagh, N., Quisquater, J.-J.: Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 515–532. Springer, Heidelberg (2005)
3. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: 20th STOC, pp. 1–10 (1988)
5. Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T.: Multiparty computation goes live. Cryptology ePrint Archive, Report 2008/068 (2008), <http://eprint.iacr.org/>
6. 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)
7. Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 226–243. Springer, Heidelberg (2006)
8. Boneh, D., Franklin, M.: Identity based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Boyen, X.: General ad-hoc encryption from exponent inversion IBE. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 394–411. Springer, Heidelberg (2007)
10. Chaum, D., Crepeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: 20th STOC, pp. 11–19 (1988)
11. Chen, L., Cheng, Z.: Security proof of Sakai–Kasahara’s identity-based encryption scheme. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 442–459. Springer, Heidelberg (2005)
12. Chen, L., Cheng, Z., Malone-Lee, J., Smart, N.P.: Efficient ID-KEM based on the Sakai–Kasahara key construction. IEE Proceedings - Information Security 153, 19–26 (2006)
13. Cramer, R., Damgård, I., Maurer, U.: General secure multiparty computation from any linear secret sharing scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)

14. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation, an introduction. Manuscript, May 25th (2008)
15. Damgård, I., Geisler, M., Krøigaard, M., Nielsen, J.B.: Asynchronous multiparty computation: Theory and implementation. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 160–179. Springer, Heidelberg (2009)
16. Gennaro, R., Rabin, M., Rabin, T.: Simplified VSS and fast-track multi-party computation with applications to threshold cryptography. In: Principles of Distributed Computing – PODC 1989, pp. 101–111. ACM Press, New York (1998)
17. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing over elliptic curve (in Japanese). In: The 2001 Symposium on Cryptography and Information Security, Oiso, Japan (January 2001)
18. Sakai, R., Kasahara, M.: ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054 (2003)
19. SCET: Secure Computing Economy and Trust. Homepage:
<http://sikkerhed.alexandra.dk/uk/projects/scet>
20. Shamir, A.: How to share a secret. Communications of the ACM 22, 612–613 (1979)
21. 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)
22. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
23. SIMAP: Secure Information Management and Processing. Homepage:
<http://sikkerhed.alexandra.dk/uk/projects/simap>
24. VIFF: Virtual Ideal Functionality Framework. Homepage: <http://viff.dk/>
25. Zhang, F., Safavi-Naini, R., Susilo, W.: An efficient signature scheme from bilinear pairings and its applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004)

Key Predistribution Schemes and One-Time Broadcast Encryption Schemes from Algebraic Geometry Codes

Hao Chen¹, San Ling², Carles Padró³, Huaxiong Wang², and Chaoping Xing²

¹ Software Engineering Institute, East China Normal University, Shanghai, China
haochen@sei.ecnu.edu.cn

² Division of Mathematical Sciences, School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore
{lingsan,HXWang,xingcp}@ntu.edu.sg

³ Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya,
Barcelona, Spain
cpadro@ma4.upc.edu

Abstract. Key predistribution schemes (KPSs) and one-time broadcast encryption schemes (OTBESs) are unconditionally secure protocols for key distribution in networks. The efficiency of these schemes has been measured in previous works in terms of their information rate, that is, the ratio between the length of the secret keys and the length of the secret information that must be stored by every user. Several constructions with optimal information rate have been proposed, but in them the secret keys are taken from a finite field with at least as many elements as the number of users in the network. This can be an important drawback in very large networks in which the nodes have limited computational resources as, for instance, wireless sensor networks. Actually, key predistribution schemes have been applied recently in the design of key distribution protocols for such networks.

In this paper we present a method to construct key predistribution schemes from linear codes that provide new families of KPSs and OTBESs for an arbitrarily large number of users and with secret keys of constant size. As a consequence of the Gilbert-Varshamov bound, we can prove that our KPSs are asymptotically more efficient than previous constructions, specially if we consider KPSs that are secure against coalitions formed by a constant fraction of the users. We analyze as well the KPSs that are obtained from families of algebraic geometry linear codes that are above the Gilbert-Varshamov bound, as the ones constructed from the curves of Garcia and Stichtenoth. Finally, we discuss how the use of KPSs based on algebraic geometry codes can provide more efficient OTBESs.

1 Introduction

Key predistribution schemes and one-time broadcast encryption schemes are unconditionally secure protocols for key distribution in networks.

A *key predistribution scheme* (KPS) is a method by which a *trusted authority* (TA) distributes secret information among a set of users in such a way that every user in a group in some specified family of *privileged subsets* is able to compute a common key associated with that group. In addition, certain coalitions of users (*forbidden subsets*) outside a privileged subset must not be able to find out any information on the value of the key associated to that subset.

One-time broadcast encryption schemes (OTBES) are closely related to key predistribution schemes. Such a scheme consists of two phases. In the first one, the trusted authority privately distributes some secret information to every user. In the second phase, the TA selects a member in the family of privileged subsets and broadcasts through an open channel an encrypted common key for that subset. Every user in the privileged subset must be able to decrypt the common key by using its secret information, while any forbidden coalition obtains no information on the common key. Broadcasting some public information in an OTBES makes it possible, in general, to reduce the amount of secret information that every user receives in the predistribution phase.

Key predistribution schemes were introduced by Blom [3], while the first one-time broadcast encryption scheme was proposed by Berkovits [2]. Several other authors have studied these topics [1,4,5,6,14,18,23,26,27,29,30,31]. The survey by Stinson [29] contains detailed descriptions of the main proposed constructions. The complexity of KPSs and OTBESs is evaluated in most of these works in terms of their *information rates*, that is, the ratio between the length of the common keys and the length of the secret information stored by the users or the length of the broadcast message.

In some recent works [12,13,19,21], the KPSs proposed by Blom [3] and by Blundo et al. [5] have been used to design key distribution protocols for wireless sensor networks, in which the nodes have strong limitations in their computing and communication capabilities.

The KPSs in [3,5] are based on the evaluation of multivariate symmetric polynomials over a finite field \mathbb{F}_q . The common keys are elements in \mathbb{F}_q , and every user receives as its secret information several elements in \mathbb{F}_q . According to the bounds in [4], these schemes have optimal information rate. Nevertheless, the size of the finite field \mathbb{F}_q depends on the number n of users because it is required that $q \geq n$. Therefore, for a network with many nodes, as it is the case for wireless sensor networks, the bit length of the secret information of every user is increased by a $\log n$ factor (unless otherwise is stated, all logarithms in this paper are defined to the base 2). The construction of KPSs in which the size of the common keys is independent from the number of users have important implications in the design of key distribution protocols for wireless sensor networks.

Observe that a similar problem occurred in secret sharing. In the threshold secret sharing scheme by Shamir [28], which is the keystone of many secure multi-party computation protocols, the secret is an element in a finite field \mathbb{F}_q , where q must be greater than the number of users. By using algebraic geometric error correcting codes, Chen and Cramer [9] introduced a new family of secret sharing

schemes that makes it possible to perform secure multi-party computation over fields whose size is much smaller than the number of users. One of the ingredients of the construction in [9] is the connection between linear codes and secret sharing [22], which is based on the generalization by Brickell [7] of Shamir's polynomial-based secret sharing scheme to linear secret sharing schemes.

Padró et al. [26,27] observed that most of the proposed key predistribution schemes and one-time broadcast encryption schemes are *linear*, that is, all random variables involved in those schemes are defined by linear mappings. In this way, the previous polynomial-based constructions of KPSs and OTBESs [3,5,6] were generalized to linear KPSs and OTBESs in [26,27].

2 Our Results

Similarly to secret sharing schemes, there exists a connection between linear key predistribution schemes and linear codes, which is described in this paper for the first time. This connection is exploited to find new constructions of KPSs by using linear codes with good properties, specially algebraic geometry codes. Our techniques provide families of KPSs in which the secret keys have constant size for an arbitrarily large number of users, while in previous constructions the length of the secret keys grows with the logarithm of the number of users. In the case that the KPSs must be secure against a constant fraction of the users, which is a reasonable assumption in wireless sensor networks, we prove that our KPSs are asymptotically more efficient than the ones given by Blundo et al. [5]. The proof is based on the Gilbert-Varshamov bound. We discuss as well the properties the KPSs that are obtained from algebraic geometry codes on the explicit family of curves with many rational points given by Garcia and Stichtenoth [15]. Finally, we apply these new techniques to the construction of new, more efficient OTBESs.

3 Definitions and Notation

Given a set \mathcal{U} of users with $|\mathcal{U}| = n$, consider a family $\mathcal{P} \subseteq 2^{\mathcal{U}}$ of *privileged subsets* and a family $\mathcal{F} \subseteq 2^{\mathcal{U}}$ of *forbidden subsets*. In a $(\mathcal{P}, \mathcal{F}, n)$ -key predistribution scheme, or $(\mathcal{P}, \mathcal{F}, n)$ -KPS for short, every user $i \in \mathcal{U}$ receives from a *trusted authority* (TA) a *fragment* $u_i \in U_i$ such that, for every $P \in \mathcal{P}$, a *common key* $k_P \in K$ can be computed from u_i by every user $i \in P$, while the coalitions of users $F \in \mathcal{F}$ with $F \cap P = \emptyset$ do not obtain any information about k_P from their fragments. Formally, if U_i and K_P , for every $i \in \mathcal{U}$ and $P \in \mathcal{P}$, denote the random variables corresponding, respectively, to the fragment u_i and the common key k_P , then for every $P \in \mathcal{P}$,

- $H(K_P|U_i) = 0$ for every $i \in P$, and
- if $F \in \mathcal{F}$ is such that $P \cap F = \emptyset$, then $H(K_P|(U_j)_{j \in F}) = H(K_P)$.

Observe that all common keys k_P are taken from the same set K . Moreover, we assume that all values of k_P are equally probable, that is, $H(K_P) = \log |K|$ (all logarithms in this paper are defined to the base 2).

Clearly, the family \mathcal{F} of forbidden subsets must be monotone decreasing. In a (\mathcal{P}, w, n) -KPS, where $1 \leq w \leq n$, the family \mathcal{F} consists of all subsets of \mathcal{U} with at most w users. In a (t, \mathcal{F}, n) -KPS (or a $(\leq t, \mathcal{F}, n)$ -KPS), where $2 \leq t \leq n$, the family \mathcal{P} of privileged subsets consists of all subsets of exactly t users (or, respectively, at most t users) of \mathcal{U} . In this paper, we are interested mainly in (t, w, n) -KPS.

A $(\mathcal{P}, \mathcal{F}, n)$ -one-time broadcast encryption scheme, or $(\mathcal{P}, \mathcal{F}, n)$ -OTBES for short, consists of two phases. In the first one, the *key predistribution phase*, the TA privately distributes to every user $i \in \mathcal{U}$ a fragment $u_i \in U_i$. In the second one, the *broadcast phase*, for a selected privileged subset $P \in \mathcal{P}$ and a *secret message* key $m_P \in K$, the TA publicly broadcasts a *broadcast message* $b_P \in B_P$ that is an encryption of m_P . Every user $i \in P$ can compute m_P from its fragment u_i and the broadcast message b_P , while, even after seeing the broadcast message, the users in a forbidden subset $F \in \mathcal{F}$ with $F \cap P = \emptyset$ do not obtain any information about k_P . As before, we can formally state these properties by using entropies.

- $H(K_P | (U_i)_{i \in \mathcal{U}}) = H(K_P)$ for every $P \in \mathcal{P}$. That is, secret message m_P is independent from the fragments distributed in the key predistribution phase.

In addition, for every $P \in \mathcal{P}$,

- $H(K_P | B_P, U_i) = 0$ for every $i \in P$, and
- if $F \in \mathcal{F}$ is such that $P \cap F = \emptyset$, then $H(K_P | B_P, (U_j)_{j \in F}) = H(K_P)$.

The *information rate of a KPS* is defined as $\rho = \frac{\log |K|}{\max_{i \in \mathcal{U}} H(U_i)}$, that is, the ratio between the length of the common keys and the maximum length of the fragments stored by the users. The *information rate of an OTBES* is defined analogously, but in this case we need a new parameter because we have to take into account as well the length of the broadcast message. The *broadcast information rate of an OTBES* is $\rho_B = \frac{\log |K|}{\max_{P \in \mathcal{P}} H(B_P)}$. Instead of the information rates, in this paper we will mainly measure the efficiency of KPSs and OTBESs by the actual bit-length of the fragments stored by the users and the bit-length of the broadcast message. This is due to the fact that in the known constructions that optimize the information rate, the length of the secret keys increases with the number of users, and hence the same happens with the fragments stored by the users.

4 Known Constructions of Key Predistribution Schemes and One-Time Broadcast Encryption Schemes

4.1 Key Predistribution Schemes with Optimal Information Rate

We describe in this section some KPSs proposed in the literature. Their common feature is that, by the bounds given in [4], they have optimal information rate. More details about these constructions can be found in [29].

A trivial (\mathcal{P}, n, n) -KPS is constructed by distributing, for every $P \in \mathcal{P}$, a random common key $k_P \in K$ to all users in P . For instance, in a trivial (t, n, n) -KPS, every user receives as its fragment $\binom{n-1}{t-1}$ elements in the set K of possible values of the common keys.

In the $(\leq n, \mathcal{F}, n)$ -KPS by Fiat and Naor [14], the set K is assumed to be an abelian group. For every $F \in \mathcal{F}$, a random value $s_F \in K$ is distributed to all users in $\mathcal{U} - F$. The common key for a set $P \subseteq \mathcal{U}$ is $k_P = \sum_{F \in \mathcal{F}, P \cap F = \emptyset} s_F$. In a Fiat-Naor $(\leq n, w, n)$ -KPS, the fragment of every user consists of $\sum_{j=0}^w \binom{n-1}{j}$ elements in K .

Blom [3] presented a $(2, w, n)$ -KPS based on polynomial evaluation. This scheme was generalized by Blundo et al. [5] to the (t, w, n) -KPS that is described in the following. Recall that a polynomial f on t variables is said to be *symmetric* if $f(x_1, \dots, x_t) = f(x_{\sigma 1}, \dots, x_{\sigma t})$ for every permutation σ . In the Blundo et al. (t, w, n) -KPS, the set K is a finite field \mathbb{F}_q with $q \geq n = |\mathcal{U}|$. Consider n distinct elements s_1, \dots, s_n in the finite field \mathbb{F}_q . These values are known by all users. The TA chooses uniformly at random a symmetric polynomial $f(x_1, x_2, \dots, x_t)$ on t variables and coefficients in \mathbb{F}_q with degree at most w on each variable. Every user $i \in \mathcal{U}$ receives as its fragment u_i the symmetric polynomial on $t-1$ variables that is obtained from f by fixing the first variable to s_i , that is,

$$u_i(x_2, \dots, x_t) = f(s_i, x_2, \dots, x_t).$$

The common key corresponding to a privileged subset $P = \{i_1, i_2, \dots, i_t\}$ is

$$k_P = f(s_{i_1}, s_{i_2}, \dots, s_{i_t}) \in \mathbb{F}_q.$$

Since the polynomial f is symmetric, this value can be computed by every user in P . Therefore, the fragment u_i of every user consists of $\binom{t+w-1}{t-1}$ elements in the field \mathbb{F}_q , because this is the number of coefficients of a symmetric polynomial on $t-1$ variables with degree at most w on each variable. Therefore, the fragment bit-length of the Blundo et al. (t, w, n) -KPS is $\binom{t+w-1}{t-1} \log q \geq \binom{t+w-1}{t-1} \log n$.

4.2 Tradeoff between Storage and Communication in One-Time Broadcast Encryption Schemes

It is not possible to optimize both information rate and broadcast information rate of OTBESs. A (t, w, n) -OTBES with broadcast information rate equal to 1, which is the best possible value, is easily constructed from any given (t, w, n) -KPS. For a privileged set $P \subseteq \mathcal{U}$, the broadcast message is $b_P = m_P + k_P$, that is, the message m_P is encrypted by using the secret common key k_P corresponding to the set P in the considered KPS. On the other extreme, a (t, w, n) -OTBES

that optimizes the information rate is trivially constructed. In the key predistribution phase, every user $i \in \mathcal{U}$ receives a random key $k_i \in K$. For a privileged set $P \subseteq \mathcal{U}$, the broadcast message for $m_P \in K$ is $b_P = (b_i)_{i \in P} = (m_P + k_i)_{i \in P} \in K^t$. Observe that the amount of secret information stored by the users is decreased by increasing the length of the broadcast message.

Several constructions providing a good tradeoff between storage and communication have been proposed [1,6,25,29,30,31]. The main problem of some of these proposals is that the secret message has to be taken from a very large set. Therefore, even though good values for the information rates are achieved, the actual lengths of the broadcast message and the secret information stored by the users are too large.

Blundo, Frota Mattos and Stinson [6] showed how to construct, for a prime power $q \geq n$ and for every integer ℓ with $1 \leq \ell \leq t$, a (t, w, n) -OTBES whose information rate and broadcast information rate are, respectively,

$$\rho = \binom{t-1}{\ell-1} / \binom{t+w-1}{\ell-1} \quad \text{and} \quad \rho_B = \frac{\ell}{t}.$$

Observe that an OTBES with broadcast information rate equal to 1 is obtained when $\ell = t$, while the OTBES with $\ell = 1$ optimizes the information rate. These two schemes coincide with the easy constructions we described at the beginning of this section. A tradeoff between the information rate and the broadcast information rate is provided by the different (t, w, n) -OTBESs that are obtained by taking the different values of $\ell = 1, \dots, t$.

The KIO construction [29] is a method to construct OTBESs by combining KPSs with secret sharing schemes. This method was applied in [30] using combinatorial designs. By using ramp secret sharing schemes instead of perfect ones, the OTBESs in [29,30] were improved in [31].

5 Linear Key Predistribution Schemes and Linear Codes

We present in this section a new method to construct KPSs. Specifically, we show how to construct a (t, w, n) -KPS from any given linear error correcting code. In addition, we analyze how KPSs from error correcting codes improve the efficiency of the previously proposed (t, w, n) -KPSs, specially when algebraic geometry codes are considered. Recall that the (t, w, n) -KPSs by Blundo et al [5], which generalize the $(2, w, n)$ -KPSs by Blom [3], have optimal information rate. But, since in those schemes the common keys must be taken from a finite field \mathbb{F}_q with $q \geq n$, the bit length of the fragments can be improved, specially if the number n of users is very large.

Padró et al. [26,27] observed that most of the proposed key predistribution schemes and one-time broadcast encryption schemes were *linear*, that is all random variables involved in those schemes are defined by linear mappings. By taking that into account, a general framework to study linear KPS and OTBES was proposed in [26,27]. In addition, new constructions that generalized previous ones were proposed. In particular, the following result is a direct consequence of [26, Definition 4.2 and Theorem 4.3].

Proposition 1 ([26]). Let V be a vector space with $\dim V = k$ over a finite field \mathbb{F}_q and let $\{v_1, \dots, v_n\}$ be a set of vectors in V such that every subset of $w+1$ vectors is linearly independent. Then, for every t with $2 \leq t \leq q$, there exists a (t, w, n) -KPS with common keys in \mathbb{F}_q and such that the bit-length of the fragment of every user is $\binom{t+k-2}{t-1} \log q$.

We briefly describe in the following how this (t, w, n) -KPS is constructed. Let $\mathcal{U} = \{1, \dots, n\}$ be the set of users. The TA selects uniformly at random a symmetric t -linear map

$$T: V^t = V \times \dots \times V \rightarrow \mathbb{F}_q.$$

The fragment corresponding to user $i \in \mathcal{U}$ is the symmetric $(t-1)$ -linear map $T_i: V^{t-1} \rightarrow \mathbb{F}_q$ that is obtained by fixing the first variable of T to the vector v_i . That is, $T_i(u_1, \dots, u_{t-1}) = T(v_i, u_1, \dots, u_{t-1})$ for every $(u_1, \dots, u_{t-1}) \in V^{t-1}$. The common key corresponding to a privileged subset $P = \{i_1, \dots, i_t\} \subseteq \mathcal{U}$ is $k_P = T(v_{i_1}, \dots, v_{i_t}) \in \mathbb{F}_q$. The fragment bit-length of this scheme is deduced from the dimension of the vector space of the symmetric $(t-1)$ -linear maps $T: V^{t-1} \rightarrow \mathbb{F}_q$.

Theorem 2. Let $C \subset \mathbb{F}_q^n$ be an $[n, k]$ linear code such that the dual code C^\perp has minimum distance d^\perp . Then, for every t with $2 \leq t \leq q$, there exists a $(t, d^\perp - 2, n)$ -KPS with common keys in \mathbb{F}_q and such that the bit length of the fragment of every user is $\binom{t+k-2}{t-1} \log q$.

Proof. Apply Theorem 1 to the vectors $v_1, \dots, v_n \in \mathbb{F}_q^k$ corresponding to the columns of a generator matrix of C . Since every $d^\perp - 1$ columns are linearly independent, the result follows. \square

If $C \subseteq \mathbb{F}_q^n$ is the Reed-Solomon code, we recover the KPS by Blundo et al [5]. Let C^\perp be the trivial $[n, 1, n]$ code over \mathbb{F}_q , spanned by the vector $(1, \dots, 1) \in \mathbb{F}_q^n$. Then C is an $[n, n-1]$ code over \mathbb{F}_q . The $(2, n-2, n)$ -KPS that is obtained from C is almost the same as the one proposed by Matsumoto and Imai [23].

Example 3. If the number n of users is not too large, the many known constructions of codes over small fields can be used to construct KPSs. For $q = 2, 3, 5, 7, 8, 9$ and $n \leq 256$, the best known codes can be found in the tables in [17]. For example, since there exists a $[256, 224, 9]$ linear code over \mathbb{F}_2 , we have a $(2, 7, 256)$ -KPS over \mathbb{F}_2 with fragment bit length $\binom{t+k-2}{t-1} = 32$. Observe that the fragment bit length of Blom's $(2, 7, 256)$ -KPS is at least $(w+1) \log q = 8 \log 256 = 64$ because we have to take $q \geq n$.

Example 4. Analogously, since there exists a $[240, 210, 10]$ linear code over \mathbb{F}_3 , for $t = 2, 3$ we obtain a $(t, 8, 240)$ -KPS over \mathbb{F}_3 with fragment bit length $\binom{t+30-2}{t-1} \log 3$, which is 48 if $t = 2$ and 738 if $t = 3$. In the KPS by Blundo et al., the bit

length is 72 if $t = 2$ and 357 if $t = 3$. In the second case, the bit length of the Blundo et al. KPS is smaller, but computations must be done in the larger field \mathbb{F}_{241} instead of \mathbb{F}_3 .

We see in the previous examples that, if the number of users is not too large and the common keys are taken from very small fields, more efficient KPSs can be obtained by using some of the best known codes [17] with suitable parameters.

Nevertheless, our main interest is to construct KPSs for an arbitrarily large number of users, with common secret keys of constant length that are secure against coalitions formed by a constant fraction of the users. Specifically, we want to construct infinite families of (t, w, n) -KPSs over a fixed base field \mathbb{F}_q such that t is a constant value with $2 \leq t \leq q$ and $w = cn$ for some constant c with $0 < c < 1$.

By the asymptotic Gilbert-Varshamov Bound, for every prime power q and for every δ with $0 \leq \delta < (q-1)/q$, there exists a sequence (C_n) of linear codes over \mathbb{F}_q such that C_n has length n , minimum distance $d_n \geq \delta n$, and dimension k_n with $\lim k_n/n = 1 - H_q(\delta)$, where H_q is the q -ary entropy function. In particular, there exist a positive integer n_0 and a constant α with $0 < \alpha < 1$ such that $k_n \geq (1 - \alpha)n$ for all $n \geq n_0$.

Consider $c < \delta < (q-1)/q$. Then $\delta n \geq cn + 2$ if n is not too small. For these parameters, the dual code C_n^\perp has dimension at most αn and dual minimum distance at least $cn + 2 = w + 2$. Therefore, for every large enough n , there exists a (t, cn, n) -KPS over a fixed base field \mathbb{F}_q with fragment bit length at most $\binom{t + \alpha n - 2}{t - 1} \log q$. Asymptotically, the fragment bit length is $O(n^{t-1})$. Since the size of the base field depends on the number of users, the fragment bit length for a Blundo et al. (t, cn, n) -KPS is at least $\binom{t + cn - 1}{t - 1} \log n$, which asymptotically is $O(n^{t-1} \log n)$. Therefore, our construction based on error correcting codes is asymptotically better by a factor of $\log n$.

6 Key Predistribution Schemes from Algebraic Geometry Codes

The proof of the Gilbert-Varshamov bound is existential. Nevertheless, constructions of codes over this bound have been obtained by using Goppa's algebraic geometry codes [16]. We analyze in the following the parameters of the KPSs that are obtained by considering linear codes from algebraic curves. We need to recall some basic facts about algebraic geometry codes (see for instance [32] or [9] for more information). In this paper, a *curve over \mathbb{F}_q* will mean an absolutely irreducible, projective, and smooth algebraic curve defined over \mathbb{F}_q . Let X be a curve over \mathbb{F}_q and let g be its genus. Let Q, P_1, \dots, P_n be distinct rational points on X . Consider the divisor $G = mQ$, where $2g - 1 \leq m < n$, and the set of rational functions $L(G) = \{f : (f) + G \geq 0\}$, which is a vector space with dimension $\deg(G) - g + 1 = m - g + 1$. Then

$$C = \{(f(P_1), \dots, f(P_n)) : f \in L(G)\} \subseteq \mathbb{F}_q^n$$

is an $[n, k]$ linear code with $k = m - g + 1$ and its dual code has minimum distance $d^\perp \geq \deg(G) - 2g + 2 = m - 2g + 2$.

Theorem 5. *Let X be a curve over \mathbb{F}_q , and let g be its genus and N the number of \mathbb{F}_q -rational points on X . Consider positive integers t, w, n with $2 \leq t \leq q$ and $2g + w < n \leq N - 1$. Then there exists a (t, w, n) -KPS with fragment bit length $\binom{t+w+g-1}{t-1} \log q$.*

Proof. By taking $m = 2g + w$, a linear code C with dimension $k = g + w + 1$ and dual minimum distance $d^\perp \geq m - 2g + 2 = w + 2$ is obtained. By Theorem 2, the code C provides a KPS with the required parameters. \square

The case $g = 0$ corresponds to Reed-Solomon codes, and hence the KPS by Blundo et al. [5] is obtained. By using curves with higher genus, we can obtain efficient KPSs over a constant base field for an arbitrarily large number of users. We analyze in the following the family of KPSs that is obtained from the family of curves given by Garcia and Stichtenoth [15]. Let q be a prime power. There exists a family of curves $(C_j)_{j>0}$ defined over \mathbb{F}_{q^2} such that the number of \mathbb{F}_{q^2} -rational points on C_j is $N_j \geq (q-1)q^j$ and its genus is $g_j \leq q^j$. By Theorem 5, for every positive integers j, t, w with $2 \leq t \leq q$ and $2q^j + w < (q-1)q^j - 1$, there exists a (t, w, n) -KPS over the base field \mathbb{F}_{q^2} with $n = (q-1)q^j - 1$ and fragment bit-length at most

$$\binom{t+w+q^j-1}{t-1} 2 \log q \leq \binom{t+w+\frac{n}{q-1}}{t-1} 2 \log q$$

Only KPSs over fields of the form \mathbb{F}_{q^2} are obtained in this way, but there exist other families of algebraic geometry codes that provide similar results for general fields \mathbb{F}_q [32].

We proceed to compare this family of KPSs with the ones by Blundo et al. [5]. For simplicity, we consider only the case $t = 2$. In addition, we consider KPSs that are secure against coalitions formed by a constant fraction of the users. By using the curves of Garcia-Stichtenoth, we obtain an infinite family of $(2, w, n)$ -KPSs over a fixed base field \mathbb{F}_{q^2} with fragment bit-length at most $(w + \frac{n}{q-1} + 2)2 \log q$. The fragment bit-length of the Blundo et al. $(2, w, n)$ -KPS is at least $(w+1) \log n$. We assume $w = cn$ for some constant c such that $0 < c < 1 - 2/(q-1)$. The upper bound on c guarantees that the condition $w < n - 2q^j$ is satisfied. In this situation, our KPS improves the secret information bit-length of Blundo et al. KPS if

$$j \geq 2 \left(1 + \frac{2}{c(q-1)} \right).$$

In the (t, w, n) -KPS by Blundo et al. [5], every coalition $F \subseteq \mathcal{U}$ with $|F| = w + 1$ can compute the secret information of *all* privileged subsets. In our more general construction based on error correcting codes, the common keys that a coalition $F \subseteq \mathcal{U}$ with $|F| \geq w + 1$ can obtain depend on the users involved in it. In a way, our construction has *ramp* security. From [26, Definition 4.2 and Theorem 4.3] a coalition $F \subseteq \mathcal{U}$ can obtain the secret key of a privileged subset $P \subseteq \mathcal{U}$ if and

only if one of the vectors v_i with $i \in P$ is a linear combination of the vectors in $\{v_j : j \in F\}$. Recall that the vector v_i is the i -th column of a generator matrix of the code C . For instance, in the construction in Theorem 5 the coalitions with at least $w + 2g + 1$ users can obtain the common keys of all privileged subsets, while the coalitions F with $w + 1 \leq |F| \leq w + 2g$ will obtain only a part of the common keys.

By using the known results about the weight distribution of algebraic geometric codes, we can prove some partial results about how many coalitions of ℓ users, where $\deg(G) - 2g \leq \ell \leq \deg(G)$, can get the full information of the common key. Thus the drawback of Blundo et al. KPSs that the adversary can get the full information if there are $w + 1$ attackers is not true for our KPSs, only some special coalitions of $w + 1$ attackers can get the full information of the common key.

7 Key Predistribution Schemes from Hermitian Codes

Theorem 5 provides a bound on the fragments bit length for KPSs constructed from algebraic geometry codes. This bound is derived from general properties of such codes. In this section, we discuss some cases in which the bound from Theorem 5 can be improved when considering families of algebraic geometry codes for which better bounds on the dual minimum distance are known.

Codes from Hermitian curves are an example of this situation. Let X be the curve over \mathbb{F}_{q^2} defined by $y^q + y = x^{q+1}$, which has $q^3 + 1$ rational points and genus $g = q(q - 1)/2$. Let Q be the rational point at the infinity and consider the divisor $G = mQ$ with $2g - 1 \leq m < q^3$. Then we obtain a code with length $n = q^3$, dimension $k = m - q(q - 1)/2 + 1$, and dual minimum distance $d^\perp \geq d^{\perp*} = m - q(q - 1) + 2$. By applying Theorem 5, we obtain a (t, w, q^3) -KPS over \mathbb{F}_{q^2} with $w = d^{\perp*} - 2 = m - q(q - 1)$ such that the fragment length is equal to

$$2 \left(\frac{t + w + \frac{q(q-1)}{2} - 1}{t - 1} \right) \log q. \quad (1)$$

However, in many cases d^\perp is greater than $d^{\perp*}$. Actually the values of the minimum distances of all Hermitian codes have been determined in [35]. In particular, if $m = 2q^2 - q - 2 - aq - b$ with $0 \leq b \leq a \leq q - 1$, then $d^\perp = m - q(q - 1) + 2 + b = q^2 - aq$. By setting $b = a$, we obtain from these codes, for every $t = 2, \dots, q^2$ and $a = 0, \dots, q - 1$, a (t, w, q^3) -KPS over \mathbb{F}_{q^2} with $w = d^\perp - 2 = q^2 - aq - 2$ such that the bit length of every fragment is

$$2 \left(t + \frac{\frac{3q^2}{2} - \frac{(2a+1)q}{2} - a - 3}{t - 1} \right) \log q,$$

which is smaller than the value (1) that is obtained from Theorem 5.

In addition, it is proved in [34] that there exist special \mathbb{F}_{q^2} -rational divisors on the Hermitian curves providing codes with better minimum distances. These codes have been constructed in [24]. By the results in [34], if $q > 5$ and

$$m \leq \frac{q^2 - q - 3}{2 + \log_q e} - 2,$$

there exist $[n, k, d]$ codes with $n = q^3$, and $k = q^3 - (q^2 - q)/2 - m + 2$, and

$$d \geq \frac{q^2 - q - 3 + 2m}{4 + \log_q e}.$$

Here $e \approx 2.71$ is the basis of the natural logarithm. By using these codes, another family of algebraic geometry KPSs is obtained.

Proposition 6. *If $q \geq 5$ and $w \leq (5q^2 - 5q - 51)/15$, then there exists a (t, w, q^3) -KPS over \mathbb{F}_{q^2} with fragment bit length*

$$2 \left(t + \frac{\frac{q^2 - q}{2} + \frac{(w+2)(4+\log_q e) - (q^2 - q - 3)}{2}}{t - 1} - 3 \right) \log q.$$

8 Constructions from Special Divisors on Garcia-Stichtenoth Towers of Curves

In this section, we apply the results in [8] to find a family of (t, w, n) -KPS over a constant size field \mathbb{F}_{q^2} and with n arbitrarily large. The fragment bit length of the improved $(2, n/q, n)$ -KPS from Theorem 7 below is smaller by a factor of around n/q than the value that is given by Theorem 5.

Consider the family $(C_j)_{j>0}$ of curves over \mathbb{F}_{q^2} given by Garcia and Stichtenoth [15]. Then the genus $g_j = g(C_j)$ of C_j is $(q^{j/2} - 1)^2$ if j is even and $(q^{(j+1)/2} - 1)(q^{(j-1)/2} - 1)$ if j is odd. The number of the \mathbb{F}_{q^2} -rational points in the curve C_j is $N_j = N(C_j) \geq q^j(q - 1)$. It is proved in [8] that the minimum distance d of the residual algebraic geometry codes from the divisor $G = (2q^j - u)P_\infty$, where $u < \min\{4q^{j/2}, 2q^{j-4}\}$, and the set of $q^j(q - 1)$ $GF(q^2)$ points satisfies $d \geq q^{j-1} - 6q^{j-2} + 4q^{j/2}$ when $q \geq 7$ and $j \geq 5$. Thus we have the following result.

Theorem 7. *For any given prime power satisfying $q \geq 7$, positive integers $i \geq 5$, and $t \leq q$, and $n = q^j(q - 1)$, we have a $(t, q^{j-1} - 6q^{j-2} + 4\lfloor q^{j/2} \rfloor - 2, n)$ -KPS over \mathbb{F}_{q^2} such that the bit-length of every fragment is*

$$2 \left(t + 2q^j - \min\{\lfloor 4q^{j/2} \rfloor, 2q^{j-4}\} - g_j - 1 \right) \log q.$$

Next theorem provides a non-constructive result that further improves the fragment bit length of the KPSs from Garcia-Stichtenoth curves. It follows from [33, Proposition 2.3] and some computation.

Theorem 8. *Consider $n \leq g_j(q - 1) - 1$ rational points on the curve C_j , and a positive integer m with $2g_j - 1 < m < n$. Set $r = \mu g_j$ where μ is such that $0 < \mu < 2/(q + 1)$ and*

$$\frac{\mu}{2} + \frac{2}{\log q} H_2 \left(\frac{\mu}{2} \right) < 1 + (q - 1) \left(\log_q \left(\frac{q}{q - 1} \right) - \frac{H_2(\frac{m-2g}{n})}{\log_2 q} \right).$$

Then, for every large enough j , there exists an algebraic geometry $[n, k, d]$ code on the curve C_j satisfying $k = m - g_j + 1$ and $d \geq n - m + r + 1$. As a consequence, there exists a $(t, n - m + r - 1, n)$ -KPS over \mathbb{F}_{q^2} such that the fragment bit length is

$$2 \binom{t + n + g_j - m - 3}{t - 1} \log_2 q.$$

9 One-Time Broadcast Encryption Schemes over Constant Size Fields

In the previous sections, we showed how to construct KPSs over constant size fields for an arbitrarily large number of users by using algebraic geometry codes. We present in the following an application of our results to the construction of OTBESs over constant size fields.

We describe in the following the family of (t, w, n) -OTBESs proposed by Blundo, Frota Mattos and Stinson [6], which is an improvement of the proposal by Beimel and Chor [1]. Consider a prime power $q \geq n$ and a positive integer ℓ that is a divisor of t .

- Let $P \subseteq \mathcal{U}$ be a set of users with $|P| = t$. Consider the collection of subsets of P with ℓ users. Since ℓ divides t , this $\binom{\ell}{t}$ blocks can be partitioned into $r = \binom{\ell-1}{t-1}$ parallel classes (see, for instance, [20]). Each of these classes consists of t/ℓ blocks that form a partition of P . We denote these classes by C_1, \dots, C_r and the blocks in C_i are denoted by $B_{i,j}$, where $j = 1, \dots, t/\ell$.
- The key predistribution phase is done according to the Blundo et al. $(\ell, t + w - \ell, n)$ -KPS over \mathbb{F}_q . Therefore, for every set $Q \subseteq \mathcal{U}$ with $|Q| = \ell$, there is a common key $k_Q \in \mathbb{F}_q$ that can be computed by the users in Q . In addition, it can be proved that, if $|P| = t$, the vector $(k_Q : Q \subseteq P, |Q| = \ell)$ is uniformly distributed in $\mathbb{F}_q^{(\binom{t}{\ell})}$.
- To encrypt a secret message $m_P = (m_1, \dots, m_r) \in \mathbb{F}_q^r$, where $r = \binom{t-1}{\ell-1}$, addressed to the users in a set P with $|P| = t$, the TA computes $b_{i,j} = k_{B_{i,j}} + m_i$ and broadcasts the message $(b_{i,j})_{1 \leq i \leq r, 1 \leq j \leq t/\ell}$.

The secret information bit-length is $\binom{t+w-1}{\ell-1} \log q$, while the broadcast message length is $\binom{t}{\ell} \log q$. Observe that we require $q \geq n$ because a Blundo et al. KPS is used in the predistribution phase. By using instead KPSs from algebraic geometry codes, for instance the ones constructed from Garcia-Stichtenoth curves in Section 6, we can modify the previous OTBES from [6] in order to obtain OTBESs over constant size fields.

The construction in [6] can be extended to all value of $\ell = 1, \dots, t$ but, if ℓ does not divide t , the construction requires a set of $\binom{t}{\ell}$ vectors in \mathbb{F}_q^r such that every subset of $r = \binom{t-1}{\ell-1}$ vectors is a basis, and hence we need that $q \geq \binom{t}{\ell}$. This may be avoided if ℓ is a divisor of t . The general construction is a consequence of [27, Theorem 11 and Example 12]. We did not find a method to obtain OTBESs over constant size fields from this general construction.

Stinson and Wei [31] proposed a construction of OTBESs that combines KPSs, set systems (like, for instance, combinatorial designs), and ramp secret sharing schemes. Since they use Shamir-like ramp schemes, the size of the secret keys, and hence the size of the fragments, depends on the parameters of the set system. Constructions of ramp secret sharing schemes over constant size fields based on algebraic-geometry codes have been recently presented in [9,10,11]. Unfortunately, because of the bounds on the parameters involved in those constructions, it is not possible to directly apply the algebraic-geometry ramp secret sharing schemes to the OTBESs in [31] in order to obtain another family of OTBESs over constant size fields.

Acknowledgments. H. Chen's research was partially supported by National Natural Science Foundation of China under Grant 10871068. S. Ling, H. Wang and C. Xing were supported by the National Research Foundation of Singapore under Research Grant NRF-CRP2-2007-03. C. Padró's research was partially supported by the Spanish Ministry of Science under Research Project TSI2006-02731.

References

1. Beimel, A., Chor, B.: Communication in key distribution schemes. *IEEE Trans. Inform. Theory* 40, 19–28 (1996)
2. Berkovits, S.: How To Broadcast A Secret. In: Davies, D.W. (ed.) *EUROCRYPT 1991*. LNCS, vol. 547, pp. 535–541. Springer, Heidelberg (1991)
3. Blom, R.: An Optimal Class of Symmetric Key Generation Systems. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) *EUROCRYPT 1984*. LNCS, vol. 209, pp. 335–338. Springer, Heidelberg (1985)
4. Blundo, C., Cresti, A.: Space requirements for broadcast encryption. In: De Santis, A. (ed.) *EUROCRYPT 1994*. LNCS, vol. 950, pp. 287–298. Springer, Heidelberg (1995)
5. Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly-Secure Key Distribution for Dynamic Conferences. In: Brickell, E.F. (ed.) *CRYPTO 1992*. LNCS, vol. 740, pp. 471–486. Springer, Heidelberg (1993)
6. Blundo, C., Frota Mattos, L.A., Stinson, D.R.: Trade-offs Between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution. In: Koblitz, N. (ed.) *CRYPTO 1996*. LNCS, vol. 1109, pp. 387–400. Springer, Heidelberg (1996)
7. Brickell, E.F.: Some ideal secret sharing schemes. *J. Combin. Math. and Combin. Comput.* 9, 105–113 (1989)
8. Chen, H.: Codes on Garcia-Stichtenoth curves with true minimum distance greater than Feng-Rao distance. *IEEE Transactions on Information Theory* 45(8), 706–709 (1999)
9. Chen, H., Cramer, R.: Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computation over Small Fields. In: Dwork, C. (ed.) *CRYPTO 2006*. LNCS, vol. 4117, pp. 521–536. Springer, Heidelberg (2006)
10. 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)

11. Chen, H., Cramer, R., de Haan, R., Pueyo, I.C.: Strongly Multiplicative Ramp Schemes from High Degree Rational Points on Curves. In: Smart, N.P. (ed.) EU-ROCRYPT 2008. LNCS, vol. 4965, pp. 451–470. Springer, Heidelberg (2008)
12. Delgosha, F., Fekri, F.: Threshold Key-Establishment in Distributed Sensor Networks Using a Multivariate Scheme. In: Proceedings of the 25th IEEE International Conference on Computer Communications INFOCOM 2006, pp. 1–12 (2006)
13. Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J., Khalili, A.: A pairwise key predistribution scheme for wireless sensor networks. ACM Trans. Inf. Syst. Secur. 8, 228–258 (2005)
14. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
15. Garcia, A., Stichtenoth, H.: On the Asymptotic Behaviour of Some Towers of Function Fields over Finite Fields. J. Number Theory 61, 248–273 (1996)
16. Goppa, V.D.: Codes on algebraic curves. Soviet Math. Dokl. 24, 170–172 (1981)
17. Grassl, M.: Bounds on the minimum distance of linear codes,
<http://www.codetables.de>
18. Kurosawa, K., Yoshida, T., Desmedt, Y., Burmester, M.: Some Bounds and a Construction for Secure Broadcast Encryption. In: Ohta, K., Pei, D. (eds.) ASI-ACRYPT 1998. LNCS, vol. 1514, pp. 420–433. Springer, Heidelberg (1998)
19. Lee, J., Stinson, D.R.: Deterministic Key Predistribution Schemes for Distributed Sensor Networks. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 294–307. Springer, Heidelberg (2004)
20. van Lint, J.H., Wilson, R.M.: A Course in Combinatorics. Cambridge University Press, Cambridge (1992)
21. Liu, D., Ning, P., Li, R.: Establishing pairwise keys in distributed sensor networks. ACM Trans. Inf. Syst. Secur. 8, 41–77 (2005)
22. Massey, J.L.: Minimal codewords and secret sharing. In: Proceedings of the 6th Joint Swedish-Russian Workshop on Information Theory, pp. 269–279 (1993)
23. Matsumoto, T., Imai, H.: On the Key Predistribution System: A Practical Solution to the Key Distribution Problem. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 185–193. Springer, Heidelberg (1988)
24. Matthews, G.L.: Weierstrass semigroups and codes from the quotients of Hermitian curve. Des. Codes Cryptogr. 37, 473–492 (2005)
25. Padró, C., Gracia, I., Martín, S.: Improving the trade-off between storage and communication in broadcast encryption schemes. Discrete Appl. Math. 143, 213–220 (2004)
26. Padró, C., Gracia, I., Martín Molleví, S., Morillo, P.: Linear Key Predistribution Schemes. Des. Codes Cryptogr. 25, 281–298 (2002)
27. Padró, C., Gracia, I., Martín, S., Morillo, P.: Linear broadcast encryption schemes. Discrete Appl. Math. 128, 223–238 (2003)
28. Shamir, A.: How to share a secret. Commun. of the ACM 22, 612–613 (1979)
29. Stinson, D.R.: On some methods for unconditionally secure key distribution and broadcast encryption. Des. Codes Cryptogr. 12, 215–243 (1997)
30. Stinson, D.R., van Trung, T.: Some New Results on Key Distribution Patterns and Broadcast Encryption. Des. Codes Cryptogr. 14, 261–279 (1998)
31. Stinson, D.R., Wei, R.: An application of ramp schemes to broadcast encryption. Inform. Process. Lett. 69, 131–135 (1999)

32. Tsfasman, M.A., Vlăduț, S.G.: Algebraic-Geometric Codes. Kluwer Academic Publishers Group, Dordrecht (1991)
33. Xing, C.: Algebraic geometry codes with asymptotic parameters better than Gilbert-Varshamov bound and Tsfasman-Vladut-Zink bound. IEEE Transactions on Information Theory 47(1), 347–352 (2002)
34. Xing, C., Chen, H.: Improvement on parameters of one-point AG codes from Hermitian curves. IEEE Transactions on Information Theory 47(2), 535–537
35. Yang, K., Kumar, P.V.: On the true minimum distance of Hermitian codes. Coding Theory and Algebraic Geometry, Lecture Notes in Math. 1518, 99–107 (1992)

Attribute-Based Encryption Supporting Direct/Indirect Revocation Modes

Nuttapong Attrapadung and Hideki Imai

Research Center for Information Security (RCIS),
National Institute of Advanced Industrial Science and Technology (AIST)
Akihabara-Daibiru Room 1003, 1-18-13, Sotokanda,
Chiyoda-ku, Tokyo 101-0021 Japan
{n.attrapadung,h-imai}@aist.go.jp

Abstract. Attribute-based encryption (ABE) enables an access control mechanism over encrypted data by specifying access policies among private keys and ciphertexts. In this paper, we focus on ABE that supports revocation. Currently, there are two available revocable ABE schemes in the literature. Their revocation mechanisms, however, differ in the sense that they can be considered as direct and indirect methods. *Direct revocation* enforces revocation directly by the sender who specifies the revocation list while encrypting. *Indirect revocation* enforces revocation by the key authority who releases a key update material periodically in such a way that only non-revoked users can update their keys (hence, revoked users' keys are implicitly rendered useless). An advantage of the indirect method over the direct one is that it does not require senders to know the revocation list. In contrast, an advantage of the direct method over the other is that it does not involve key update phase for all non-revoked users interacting with the key authority. In this paper, we present the first *Hybrid Revocable ABE* scheme that allows senders to select on-the-fly when encrypting whether to use either direct or indirect revocation mode; therefore, it combines best advantages from both methods.

1 Introduction

Attribute-based encryption (ABE) enables an access control mechanism over encrypted data using access policies and ascribed attributes among private keys and ciphertexts. ABE was introduced first by Sahai and Waters [20] and refined by many subsequent works [16,5,19,9,15,22,2]. In an ABE system, an encryptor specifies a set of attributes, which could be any keywords describing the ciphertext, directly in the encryption algorithm (which can be run by anyone knowing the universal public key issued priorly by an authority). A user in the system possesses a key associated with an access policy, stating what kind of ciphertext that she can decrypt. Users' keys are priorly given from the key authority. Such a user can decrypt a ciphertext if the policy associated to her key is satisfied by the attribute set associated with the ciphertext. An example application of ABE is pay-TV system with package policy (called target broadcast system

in [16]). There, a ciphertext will be associated with an attribute set, such as $\omega = \{ \text{"TITLE:24"}, \text{"GENRE:SUSPENSE"}, \text{"SEASON:2"}, \text{"EPISODE:13"} \}$, while a policy such as $\mathbb{A} = \text{"SOCCER"} \vee (\text{"TITLE:24"} \wedge \text{"SEASON:5"})$ will be associated to TV program package keys that user receives when subscribes.

1.1 Motivation

Revocation mechanism is necessary for any encryption schemes that involve many users, since some private keys might get compromised at some point. In simpler primitives such as public key infrastructure and ID-based encryption (IBE), there are many revocation methods proposed in the literature [17,1,18,7,12,6,4].

In attribute-based setting, Boldyreva et al. [6] only recently proposed a revocable ABE scheme (extended from their main contribution, a revocable IBE). Their scheme uses a key update approach roughly as follows. The sender will encrypt with the attribute set ω as usual, and in addition, he also specifies the present time slot attribute, *e.g.*, “TIME:2009.WEEK49”. The key authority, who possesses the current revocation list, periodically announces a key update material at each time slot so that only non-revoked users can update their key and use it to decrypt ciphertexts encrypted at the present time. We call this approach an *indirect* revocation, since the authority indirectly enables revocation by forcing revoked users to be unable to update their keys.

The indirect revocation has an advantage that senders do not need to know the revocation list. However, it also has a disadvantage that the key update phase can be a bottleneck since it requires communication from the key authority to *all* non-revoked users at *all* time slots. One of the main motivations for Boldyreva et al. [6] was also to reduce this cost from a naive approach which would require the update key of size $O(n - r)$ group elements. Here n is the number of users, r is the number of revoked users. Their scheme reduces this to $O(r \log(\frac{n}{r}))$, by using the classic Complete-subtree method [1,18] combined in a non-trivial way with the fuzzy IBE scheme of [20].

In order to eliminate this bottlenecked key update phase completely, Attrapadung and Imai [3] recently proposed an ABE system with *direct revocation*. Such a system allows senders to specify the revocation list directly when encrypting. Therefore, revocation can be done instantly and does not require the key update phase as in the indirect method. Despite this clear advantage, in contrast, its disadvantage is that it requires senders to possess the current revocation list. While the management of revocation list itself could be already a troublesome task, this requirement renders the system not being so purely attribute-based. (An ideal attribute-based setting should allow senders to just create ciphertext based solely on attributes and not to worry about revocation). The authors in [3] argued that, however, this setting is still reasonable for some applications such as the Pay-TV example above, where the sender is the TV program distributor company, who should possess the pirate key list to be revoked.

This nature of such an exact opposite tradeoff between the direct and indirect revocation motivates us to look for a more flexible system that supports both revocation methods so that we could have the best of both worlds.

1.2 Our Goal and Contributions

In this paper, we propose a new system called *Hybrid Revocable Attribute-Based Encryption (HR-ABE)*. This system allows a sender Alice to be able to select whether to use either direct or indirect revocation mode on-the-fly when encrypting a message. On the other hand, a user Bob possesses only one key but will be able to decrypt ciphertexts that were constructed in *either modes*.

An HR-ABE works as follows. When Alice selects the direct mode, she will specify the revocation list R directly into the encryption algorithm. On the other hand, when selecting the indirect mode, she is required only to specify the present time slot t (besides the usual attribute set input). A user Bob has one private key. Let A be the access policy associated to Bob's key. In addition, his key will be associated with a unique serial number id . If ciphertext was from the direct mode, he can decrypt solely by his key. Let ω be the attribute set associated with ciphertext. In this case, he can decrypt if ω satisfies A and $\text{id} \notin R$. If ciphertext was from indirect mode, he must obtain an update key $\text{uk}_{(R,t)}$ from the authority at time t . Again, he can decrypt if ω satisfies A , and $\text{id} \notin R$. Notice that in this latter case, the key authority specifies R when creating the update key, hence enforces revocation indirectly.

One trivial construction for HR-ABE is to use two sub-systems: a directly revocable ABE and an indirectly revocable ABE. A user key then consists of two keys, one from each sub-system. To encrypt in a desired mode, Alice just uses the corresponding sub-system. The problem for this approach is that the key size will be the sum of key sizes from both sub-systems. Therefore, our goal is to construct an efficient scheme which has the key size being roughly the same as in either the currently best directly or indirectly revocable ABE.

Another goal in designing a scheme is to base its security to the weakest assumption as possible. Since the currently most efficient (non-revocable) ABE [16] is based on the Decision Bilinear Diffie-Hellman (DBDH) assumption, we will also base the security of our scheme on this assumption.

The currently best indirectly revocable ABE that is based on DBDH assumption is the scheme of Boldyreva et al. [6] (given only implicitly in their paper, though). For the case of directly revocable ABE, to the best of our knowledge, no such scheme is available yet. However, in this paper, we give a notice that a variant of Attrapadung-Imai [3] achieves such a property.¹ Both the scheme of [6] and the variant of [3] have the key size of $2\ell \log(n)$ group elements. Therefore, the trivial combination yields the key size of $4\ell \log(n)$. Here ℓ is the number of attributes appear in the policy.

In this paper, we first formalize the notions of HR-ABE and propose a HR-ABE scheme with key size $2(\ell + 1) \log(n)$. This is roughly the same size as the two above one-mode revocable ABEs (and hence half size of the trivial combined scheme). The ciphertext size in direct mode is the same as the variant of [3]. The

¹ This variant itself is not trivial but we jump a step forward to construct a hybrid revocable ABE system, instead of only a directly revocable one. We will mention this variant in Section §6, though.

Table 1. Simple comparison among encryption primitives supporting revocation

	Indirectly revocable ABE	Directly revocable ABE	Broadcast encryption
Attribute-based setting	✓	✓	✗
Revocator	Authority	Sender	Sender
No need for key update	✗	✓	✓

ciphertext and update key sizes in indirect mode is the same as that of [6]. See Table 2 in Section §6 for comparison.

The security of our scheme is based on the DBDH assumption in the standard model. The security proof is itself quite non-trivial since in the proof, we have to simulate the key of same structure to be able to handle both attack models corresponding to two revocation modes.

1.3 Related Works

Broadcast Encryption. Broadcast encryption (BE) schemes [11,18,8,21,13] allow a sender to specify a receiver group when encrypting. The reader might wonder whether we can just use a public-key BE system instead of ABE in the case when the sender knows the revocation list by simply specifying all non-revoked users as the receiver group. The answer is that we cannot, since we focus on the *attribute-based setting*, which means that the sender is supposed not to even know whose access policy will match the attribute set associated to ciphertext. We provide a simple comparison in Table 1.

Other ABE Variants. The ABE system we have discussed so far is called Key-Policy ABE (KP-ABE) [16]. There is another opposite variant called Ciphertext-Policy ABE (CP-ABE) [5]. In CP-ABE, the roles of an attribute set and an access policy are swapped from what we described for KP-ABE. Attribute sets are associated to keys and access policies over these attributes are associated to ciphertexts. An example application of CP-ABE is secure mailing list system with access policy. There, a private key will be assigned for an attribute set, such as $\{\text{“MANAGER”}, \text{“AGE:30”}, \text{“INSTITUTE:ABC”}\}$, while policies over attributes such as $\text{“MANAGER”} \vee (\text{“TRAINEE”} \wedge \text{“AGE:25”})$ will be associated to ciphertexts.

In this paper, we will consider only the KP-ABE variant. However, the methodology can be applied to the case of CP-ABE similarly.

Previous Works on ABE. ABE was introduced by Sahai and Waters [20] in the context of a generalization of IBE called Fuzzy IBE, which is an ABE that allows only single threshold access structures. The first (and still being state-of-the-art) KP-ABE that allow any monotone access structures was proposed by Goyal et al. [16], while the first such CP-ABE, albeit with the security proof in the generic bilinear group model, was proposed by Bethencourt, Sahai, and Waters [5]. Ostrovsky, Sahai, and Waters [19] then subsequently extended both schemes to handle also any non-monotone structures; therefore, negated clauses

can be specified in policies. Goyal et al. [15] presented bounded CP-ABE in the standard model. Waters [22] recently proposed the first fully expressive CP-ABE in the standard model. Chase [9] presented multi-authority KP-ABE. Recently, Attrapadung and Imai [2] proposed a new ABE variant called Dual-Policy ABE which is a combination of both KP and CP ABE. Revocable ABE was first mentioned by Gollé et al. [14], but their scheme was only heuristic.

2 Preliminaries

2.1 Access Structures and Linear Secret Sharing

We first provide the notion of access structure and linear secret sharing scheme as follows. Such formalization is recapped from [22].

Definition 1 (Access Structures). Let $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\mathcal{P}}$ is monotone if for all B, C we have that if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (resp., monotonic access structure) is a collection (resp., monotone collection) $\mathbb{A} \subseteq 2^{\mathcal{P}} \setminus \{\emptyset\}$.

Definition 2 (Linear Secret Sharing Schemes (LSSS)). Let \mathcal{P} be a set of parties. Let M be a matrix of size $\ell \times k$. Let $\pi : \{1, \dots, \ell\} \rightarrow \mathcal{P}$ be a function that maps a row to a party for labeling. A secret sharing scheme Π for access structure \mathbb{A} over a set of parties \mathcal{P} is a linear secret-sharing scheme in \mathbb{Z}_p and is represented by (M, π) if it consists of two polynomial-time algorithms:

Share_(M, π): The algorithm takes as input $s \in \mathbb{Z}_p$ which is to be shared. It randomly chooses $y_2, \dots, y_k \in \mathbb{Z}_p$ and let $\mathbf{v} = (s, y_2, \dots, y_k)$. It outputs $M\mathbf{v}$ as the vector of ℓ shares. The share $\lambda_{\pi(i)} := M_i \cdot \mathbf{v}$ belongs to party $\pi(i)$, where we denote M_i as the i th row in M .

Recon_(M, π): The algorithm takes as input $S \in \mathbb{A}$. Let $I = \{i \mid \pi(i) \in S\}$. It outputs reconstruction constants $\{(i, \mu_i)\}_{i \in I}$ which has a linear reconstruction property: $\sum_{i \in I} \mu_i \cdot \lambda_{\pi(i)} = s$.

Lemma 1. ([22]) Let (M, π) be a LSSS for access structure \mathbb{A} over a set of parties \mathcal{P} , where M is a matrix of size $\ell \times k$. For all $S \notin \mathbb{A}$, there exists a polynomial time algorithm that outputs a vector $\mathbf{w} = (w_1, \dots, w_k) \in \mathbb{Z}_p^k$ such that $w_1 = 1$ (or w_1 can be chosen arbitrarily in \mathbb{Z}_p) and for all $i \in [1, \ell]$ where $\pi(i) \in S$ it holds that $M_i \cdot \mathbf{w} = 0$.

2.2 Bilinear Maps and Some Assumptions

Bilinear Maps. We briefly review some facts about bilinear maps. Let \mathbb{G}, \mathbb{G}_T be multiplicative groups of prime order p . Let g be a generator of \mathbb{G} . A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ for which the following hold: (1) e is bilinear; that is, for all $u, v \in \mathbb{G}$, $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$. (2) The map is non-degenerate: $e(g, g) \neq 1$. We say that \mathbb{G} is a bilinear group if the group action in \mathbb{G} can be computed efficiently and there exists \mathbb{G}_T for which the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is efficiently computable.

DBDH Assumption. Let \mathbb{G} be a bilinear group of prime order p . The DBDH (Decision Bilinear Diffie-Hellman) problem [7] in \mathbb{G} is stated as follows. Given a tuple $(g, g^a, g^b, g^s) \in \mathbb{G}^4$ and an element $Z \in \mathbb{G}_T$ as input, determine if $Z = e(g, g)^{abs}$. An algorithm \mathcal{A} that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the DBDH problem in \mathbb{G} if $|\Pr[\mathcal{A}(g, g^a, g^b, g^s, e(g, g)^{abs}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, g^s, Z) = 0]| \geq \epsilon$. We refer to the distribution on the left as \mathcal{P}_{BDH} and the one on the right as \mathcal{R}_{BDH} . We say that the DBDH assumption holds in \mathbb{G} if no polynomial-time algorithm has a non-negligible advantage in solving the problem.

2.3 Some Terminologies for Binary Tree

We denote some terminology for complete binary tree. Let $\mathcal{L} = \{1, \dots, n\}$ be the set of leaves. Let \mathcal{X} be the set of node names in the tree via some systematic naming order. For a leaf $i \in \mathcal{L}$, let $\text{Path}(i) \subset \mathcal{X}$ be the set of all nodes on the path from node i to the root (including i and the root).

For $R \subseteq \mathcal{L}$, let $\text{Cover}(R) \subset \mathcal{X}$ be defined as follows. First mark all the nodes in $\text{Path}(i)$ if $i \in R$. Then $\text{Cover}(R)$ is the set of all the unmarked children of marked nodes. It can be shown to be the minimal set that contains no node in $\text{Path}(i)$ if $i \in R$ but contains at least one node in $\text{Path}(i)$ if $i \notin R$. This function was widely used, e.g., in revocation scheme of [1] and the Complete-Subtree broadcast encryption [18]. It is known [1,18] that $|\text{Cover}(R)| \leq |R|(\log(n/|R|) + 1)$.

2.4 Lagrange Interpolation

For $i \in \mathbb{Z}$ and $S \subseteq \mathbb{Z}$, the Lagrange basis polynomial is defined as $\triangle_{i,S}(z) = \prod_{j \in S, j \neq i} \left(\frac{z-j}{i-j} \right)$. Let $f(z) \in \mathbb{Z}[z]$ be a d -th degree polynomial. If $|S| = d+1$, from a set of $d+1$ points $\{(i, f(i))\}_{i \in S}$, one can reconstruct $f(z)$ as follows.

$$f(z) = \sum_{i \in S} f(i) \cdot \triangle_{i,S}(z).$$

In our scheme, we will particularly use the interpolation for a first degree polynomial. In particular, let $f(z)$ be a first degree polynomial, one can obtain $f(0)$ from two points $(i_1, f(i_1)), (i_2, f(i_2))$ where $i_1 \neq i_2$ by computing

$$f(0) = f(i_1) \frac{i_2}{i_2 - i_1} + f(i_2) \frac{i_1}{i_1 - i_2}. \quad (1)$$

3 Definitions

3.1 Algorithm Definition

In this section, we provide the syntax definition of Hybrid Revocable Attribute-Based Encryption (HR-ABE). Let \mathcal{N} be the universe of attributes for ABE. Let \mathcal{A} denote a collection of access structures over \mathcal{N} which are allowed to be used in the scheme. Let $\mathcal{T}, \mathcal{M}, \mathcal{U}$ be the universes of time periods, messages, and user key serial numbers, respectively. A HR-ABE scheme consists of five algorithms:

Setup(n) \rightarrow (pk , msk). This is a randomized algorithm that takes an input n which is the size of \mathcal{U} . It outputs the public key pk and a master key msk .

Encrypt(mode, a , ω , m , pk) \rightarrow ct . This is a randomized algorithm that takes as input mode $\in \{\text{'dir'}, \text{'ind'}\}$ which representing direct or indirect revocation mode resp., an auxiliary input a being either

$$a = \begin{cases} \text{a revocation list } R \subseteq \mathcal{U} & \text{if mode} = \text{'dir'} \\ \text{a present time attribute } t \in \mathcal{T} & \text{if mode} = \text{'ind'}, \end{cases}$$

a set of attributes $\omega \subseteq \mathcal{N}$, a message $m \in \mathcal{M}$, and public key pk . It outputs a ciphertext ct .

KeyGen(id, \mathbb{A} , msk , pk) \rightarrow $\text{sk}_{(\text{id}, \mathbb{A})}$. This is a randomized algorithm that takes as input a serial number $\text{id} \in \mathcal{U}$, an access structure $\mathbb{A} \in \mathcal{A}$, the master key msk , and the public key pk . It outputs a private decryption key $\text{sk}_{(\text{id}, \mathbb{A})}$.

KeyUpdate(R , t , msk , pk) \rightarrow $\text{uk}_{(R, t)}$. This is a randomized algorithm that takes as input a revocation list $R \subseteq \mathcal{U}$, a present time attribute t , the master key msk , and the public key pk . It outputs a key update material $\text{uk}_{(R, t)}$.

Decrypt(ct , (mode, a , ω), $\text{sk}_{(\text{id}, \mathbb{A})}$, (id, \mathbb{A}), pk , b) \rightarrow m . This algorithm takes as input the ciphertext ct that was encrypted under (mode, a , ω), the decryption key $\text{sk}_{(\text{id}, \mathbb{A})}$ for user index id with access control structure \mathbb{A} , the public key pk , and an auxiliary input b being either

$$b = \begin{cases} \text{an empty string} & \text{if mode} = \text{'dir'} \\ (\text{uk}_{(R, t)}, (R, t)) & \text{if mode} = \text{'ind'}. \end{cases}$$

It outputs the message m or a special symbol \perp indicating an unsuccessful decryption.

We require the standard correctness of decryption, that is, if we let $\text{Setup} \rightarrow (\text{pk}, \text{msk})$ and $\text{KeyGen}(\text{id}, \mathbb{A}, \text{msk}, \text{pk}) \rightarrow \text{sk}_{(\text{id}, \mathbb{A})}$ then for all $m \in \mathcal{M}$; $\text{id} \in \mathcal{U}$; $R \subseteq \mathcal{U}$; $\mathbb{A} \in \mathcal{A}$; $\omega \in \mathcal{N}$, $t \in \mathcal{T}$:

Direct revocation mode. Let $\text{Encrypt}(\text{'dir'}, R, \omega, m, \text{pk}) \rightarrow \text{ct}$. If $\omega \in \mathbb{A}$ and $\text{id} \notin R$, then $\text{Decrypt}(\text{ct}, (\text{'dir'}, R, \omega), \text{sk}_{(\text{id}, \mathbb{A})}, (\text{id}, \mathbb{A}), \text{pk}) = m$.

Indirect revocation mode. Let $\text{Encrypt}(\text{'ind'}, t, \omega, m, \text{pk}) \rightarrow \text{ct}$, $\text{KeyUpdate}(R, t, \text{msk}, \text{pk}) \rightarrow \text{uk}_{(R, t)}$. If $\omega \in \mathbb{A}$ and $\text{id} \notin R$, then $\text{Decrypt}(\text{ct}, (\text{'ind'}, t, \omega), \text{sk}_{(\text{id}, \mathbb{A})}, (\text{id}, \mathbb{A}), \text{pk}, \text{uk}_{(R, t)}, (R, t)) = m$.

Augment Definition. In the real usage, as mentioned earlier in Section §1.2, we will wish to assign id as a *unique* serial number for each key generated so far by the key authority. In this case, the authority will maintain the internal state for recording the set \mathcal{S} of assigned id so far. More formally, it will use a simple stateful KeyGen^s algorithm defined as follows. Initially, $\mathcal{S} = \emptyset$.

KeyGen^s(id, \mathbb{A} , msk , pk) \rightarrow $\text{sk}_{(\text{id}, \mathbb{A})}$. If $\text{id} \in \mathcal{S}$ then return \perp . Else, run $\text{KeyGen}(\text{id}, \mathbb{A}, \text{msk}, \text{pk}) \rightarrow \text{sk}_{(\text{id}, \mathbb{A})}$. Increment the internal state $\mathcal{S} \leftarrow \mathcal{S} \cup \{\text{id}\}$. It then outputs $\text{sk}_{(\text{id}, \mathbb{A})}$.

We provide some further discussions on our syntax definition and comparisons to the previous model from [6] in Appendix §A.1.

3.2 Security Notions

We now give the definitions of security notions for HR-ABE. We consider selective-target security, where the adversary is required to specify the target mode, auxiliary input, and attribute set before seeing the public key. Although we consider such a static adversary, the queries can still be asked in an adaptive manner. Indeed, we consider two notions: semi-static and adaptive query model. The semi-static one is similar to the one considered in [13] (in the context of broadcast encryption). Below, we note that the Query phase can be continued after the Challenge phase. We provide the formal definition first then explain later.

Init. The adversary declares the target $(\text{mode}^*, a^*, \omega^*)$ of encryption mode, its corresponding auxiliary input, and the target attribute set ω^* . Recall that if $\text{mode}^* = \text{'dir'}$ then $a^* = R^*$ (the target revoked set) and if $\text{mode}^* = \text{'ind'}$ then $a^* = t^*$ (the target present time).

Setup. The challenger runs the Setup of HR-ABE and hands the public key pk to the adversary.

Query Phase. The challenger answers the queries as follows.

If $\text{mode}^* = \text{'dir'}$, then for each query:

If \mathcal{A} queries for $\mathcal{SK}(\text{id}, \mathbb{A})$ then If $\omega^* \in \mathbb{A}$ then If $\text{id} \notin R^*$ then return \perp $\text{KeyGen}(\text{id}, \mathbb{A}, \text{msk}, \text{pk}) \rightarrow \text{sk}_{(\text{id}, \mathbb{A})}$ Return $\text{sk}_{(\text{id}, \mathbb{A})}$	If \mathcal{A} queries for $\mathcal{UK}(R, t)$ then $\text{KeyUpdate}(R, t, \text{msk}, \text{pk}) \rightarrow \text{uk}_{(R, t)}$ Return $\text{uk}_{(R, t)}$
--	---

If $\text{mode}^* = \text{'ind'}$, then we consider two variants of attack models.

1. **Semi-static query model.** The adversary first announces \tilde{R} . Then, for each query:

If \mathcal{A} queries for $\mathcal{SK}(\text{id}, \mathbb{A})$ then If $\omega^* \in \mathbb{A}$ then If $\text{id} \notin \tilde{R}$ then return \perp $\text{KeyGen}(\text{id}, \mathbb{A}, \text{msk}, \text{pk}) \rightarrow \text{sk}_{(\text{id}, \mathbb{A})}$ Return $\text{sk}_{(\text{id}, \mathbb{A})}$	If \mathcal{A} queries for $\mathcal{UK}(R, t)$ then If $t = t^*$ then If $\tilde{R} \not\subseteq R$ then return \perp $\text{KeyUpdate}(R, t, \text{msk}, \text{pk}) \rightarrow \text{uk}_{(R, t)}$ Return $\text{uk}_{(R, t)}$
--	--

2. **Adaptive query model.** The challenger maintains two sets \mathcal{R}_s and \mathcal{R}_u . Set \mathcal{R}_s is the set of id for keys corresponding to \mathbb{A} such that $\omega^* \in \mathbb{A}$. Set \mathcal{R}_u is the set of id which is revoked via all updated key queries at $t = t^*$. Initially, $\mathcal{R}_s = \emptyset$ and $\mathcal{R}_u = \mathcal{U}$. For each query:

If \mathcal{A} queries for $\mathcal{SK}(\text{id}, \mathbb{A})$ then If $\omega^* \in \mathbb{A}$ then If $\text{id} \notin \mathcal{R}_u$ then return \perp $\text{Else } \mathcal{R}_s \leftarrow \mathcal{R}_s \cup \{\text{id}\}$ $\text{KeyGen}(\text{id}, \mathbb{A}, \text{msk}, \text{pk}) \rightarrow \text{sk}_{(\text{id}, \mathbb{A})}$ Return $\text{sk}_{(\text{id}, \mathbb{A})}$	If \mathcal{A} queries for $\mathcal{UK}(R, t)$ then If $t = t^*$ then If $\mathcal{R}_s \not\subseteq R$ then return \perp $\text{Else } \mathcal{R}_u \leftarrow \mathcal{R}_u \cap R$ $\text{KeyUpdate}(R, t, \text{msk}, \text{pk}) \rightarrow \text{uk}_{(R, t)}$ Return $\text{uk}_{(R, t)}$
--	--

Challenge. The adversary submits two equal length messages m_0 and m_1 . The challenger flips a random bit b and computes the challenge ciphertext ct^* of m_b on the target $(mode^*, a^*, \omega^*)$ and then gives ct^* to the adversary.

Guess. The adversary outputs a guess b' of b .

The advantage of an adversary in this game is defined as $\Pr[b = b'] - \frac{1}{2}$. We note that the above definition can be easily extended to handle chosen-ciphertext attacks by allowing decryption queries in the Query Phase.

Definition 3. An HR-ABE scheme is secure in the sense of indistinguishability against selective-target with semi-static (resp., adaptive) query attack if all polynomial time adversaries have at most a negligible advantage in the above game for respective variants. Denote the two notions by IND-sHRSS and IND-sHRA respectively.

Intuition for Security Notion. We explain the intuition behind the above notion. The condition for queries in the direct mode is quite straightforward: the adversary can query secret key only if $\omega^* \notin \mathbb{A}$ or $id \notin R^*$ but can query any update key (since it should be useless in this mode).

The condition for the indirect mode is somewhat more complicated. First, consider the adaptive query model, *i.e.*, the query model in IND-sHRA. Basically, if $sk_{id, \mathbb{A}}$ such that $\omega^* \in \mathbb{A}$ and uk_{R, t^*} such that $id \notin R$ are known to the adversary, it can be used to decrypt the challenge ciphertext. The important security property we wish to capture is that knowing only one of these two should be useless for any such pairs (*à la* security against collusion). But we never know which one of the two will be queried first, so we capture this by doing the house keeping of both kinds of queries simultaneously and adaptively by using \mathcal{R}_s and \mathcal{R}_u defined above. We call a user index id a *critical* user if $\mathcal{SK}(id, \mathbb{A})$ such that $\omega^* \in \mathbb{A}$ is asked. \mathcal{R}_s maintains such a set in this adaptive query model.

Next, we consider the semi-static query model, *i.e.*, the query model in IND-sHRSS. In this notion, the adversary must commit at the beginning of the query phase the set \tilde{R} of id that could potentially be critical. Eventually even if its corresponding \mathcal{SK} query is not asked, the query $\mathcal{UK}(R, t^*)$ such that $id \notin R$ is not permitted. Such a \mathcal{UK} query would have been allowed if the adaptive query model were considered. Hence, we conclude that the semi-static model is weaker.

We provide some comparisons to the security model of [6] in Appendix §A.2.

4 A Hybrid Revocable ABE Scheme

4.1 Intuition for Our Scheme

We intuitively explain how our scheme works. First recall the approach of the indirectly revocable IBE and ABE of Boldyreva et al. [6]. It utilizes the binary tree T with n leaves similarly as in the Complete-Subtree method [1,18]. Each user will be associated to a leaf which is unassigned yet. Each node x in the tree is assigned a random secret polynomial f_x of first degree in such a way that

$f_x(0) = \alpha$, where α denotes the master key. The scheme uses $Y^\alpha = e(g, g)^{s\alpha} \in \mathbb{G}_T$ as a message encapsulation key, where s is the randomness in encryption (and we denote $Y = e(g, g)^s$).

To indirectly revoke R at time t , the key authority uses the master key to construct an *atomic* update key corresponding to each node in $\text{Cover}(R)$. Only user who has an ancestor node in $\text{Cover}(R)$, say node x , can compute $Y^{f_x(t)}$. Also, only user whose access structure is satisfied by attribute set associated with ciphertext can compute $Y^{f_x(1)}$. Due to the fact that f_x is first degree, from this two elements, decryption can be done by interpolation in the exponent to yield $Y^{f_x(0)}$.

In our scheme, the indirect mode is done in exactly the same way as above. The rough idea is that we will enable the direct revocation by letting the sender herself generate an atomic update key, albeit at a *dummy time* which is exactly the name of node x for all $x \in \text{Cover}(R)$. Similarly, only user who has an ancestor node in $\text{Cover}(R)$, say node x , can compute $Y^{f_x(x)}$. From this and usual $Y^{f_x(1)}$, he interpolates to obtain $Y^{f_x(0)}$.

The difficulty is how to enable a sender to generate such an atomic update key herself since she is not the key authority and hence cannot construct as in the indirect mode. We solve this by providing a *partial* update key as a part of private key beforehand. This appears as $(D_x^{(3)}, D_x^{(4)})$ in Eq.(2) below, which indeed has the same form as the update key in Eq.(3). This partial key is enabled only when a proper remaining part of update key from sender come with ciphertext.

4.2 The Construction

Some Terminologies. In our scheme, we define the universe set of serial numbers \mathcal{U} as the set of leaves in the complete binary tree $\mathcal{L} = \{1, \dots, n\}$. Let m be the maximum size of attribute set allowed to be associated with a ciphertext, *i.e.*, we restrict $|\omega| \leq m$. Let d be the maximum of $|\text{Cover}(R)|$ for all $R \subseteq \mathcal{U}$. Our scheme supports any linear secret-sharing access structure which we denote its universe as $\mathcal{A}_{\text{LSSS}}$. Consequently, we let an access structure in its LSSS matrix form (M, π) (*cf.* Definition 2) be input directly to the algorithms in the scheme. We sometimes denote it as $\mathcal{A}_{(M, \pi)}$. Let the attribute space be $\mathcal{N} = \mathbb{Z}_p^*$ and the message space be $\mathcal{M} = \mathbb{G}_T$.

We assume that $\mathcal{T}, \mathcal{X} \subseteq \mathbb{Z}_p^* \setminus \{1\}$ and that $\mathcal{T} \cap \mathcal{X} = \emptyset$. This is wlog since we can extend to the case where domains are $\mathcal{T}' = \mathcal{X}' = \{0, 1\}^*$ by using a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^* \setminus \{1\}$ and then using $H(0||x) \in \mathcal{X}$ instead of $x \in \mathcal{X}'$ and $H(1||t) \in \mathcal{T}$ instead of $t \in \mathcal{T}'$ in the scheme. In this case, the collision resistance ensures that $H(0||x) \neq H(1||t)$ for any x, t , hence $\mathcal{T} \cap \mathcal{X} = \emptyset$. We are now ready to describe our scheme.

- **Setup(n):** The algorithm first picks a random generator $g \in \mathbb{G}$ and also randomly chooses $u_0, \dots, u_d, h_0, \dots, h_m \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$. The public key is: $\mathsf{pk} = (g, e(g, g)^\alpha, u_0, \dots, u_d, h_0, \dots, h_m)$. For all node $x \in \mathcal{X}$ in the tree, it randomly chooses $a_x \in \mathbb{Z}_p$ for defining a first degree polynomial $f_x(z) = a_x z + \alpha$. The master key is $\mathsf{msk} = (\alpha, \{a_x\})$. It outputs $(\mathsf{pk}, \mathsf{msk})$. Define a function $P : \mathbb{Z}_p \rightarrow \mathbb{G}$ by $P(x) = \prod_{j=0}^d u_j^{(x^j)}$. Define a function $F : \mathbb{Z}_p \rightarrow \mathbb{G}$ by $F(x) = \prod_{j=0}^m h_j^{(x^j)}$.

- **Encrypt(mode, a, ω, m, pk)**: The algorithm first picks a random $s \in \mathbb{Z}_p$. It then computes

$$C = m \cdot (e(g, g)^\alpha)^s, \quad C^{(1)} = g^s, \quad C_k^{(2)} = F(k)^s,$$

where the last term is for each $k \in \omega$. From this point, two encryption modes diverge as follows.

Direct revocation mode ($\text{mode} = \text{'dir'}$). In this case, we have $a = R$, the revocation list, as an input. It runs $\text{Cover}(R)$ to find a minimal node set that covers $\mathcal{U} \setminus R$. It then computes for each $x \in \text{Cover}(R)$: $C_x^{(3)} = P(x)^s$. It outputs the ciphertext as $\text{ct} = (C, C^{(1)}, \{C_k^{(2)}\}_{k \in \omega}, \{C_x^{(3)}\}_{x \in \text{Cover}(R)})$.

Indirect revocation mode ($\text{mode} = \text{'ind'}$). In this case, we have $a = t$, the present time attribute, as an input. It computes $C^{(3)} = P(t)^s$. It then outputs the ciphertext as $\text{ct} = (C, C^{(1)}, \{C_k^{(2)}\}_{k \in \omega}, C^{(3)})$.

- **KeyGen(id, (M, π), msk, pk)**: The input consists of a serial number $\text{id} \in \mathcal{U}$ (which is a leaf in the binary tree), a LSSS access structure $(M, \pi) \in \mathcal{A}_{\text{LSSS}}$, the master key and the public key. Let M be $\ell \times k$ matrix. The algorithm computes a key as follows.

For all $x \in \text{Path}(\text{id})$, it first shares $f_x(1)$ with the LSSS (M, π) as follows. It chooses $z_{x,2}, \dots, z_{x,k} \in \mathbb{Z}_p$ and lets $\mathbf{v}_x = (f_x(1), z_{x,2}, \dots, z_{x,k})$. For $i = 1$ to ℓ , it calculates the share $\lambda_{x,i} = \mathbf{M}_i \cdot \mathbf{v}_x$, where \mathbf{M}_i is the vector corresponding to i th row of M . It then randomly chooses $r_{x,1}, \dots, r_{x,\ell}, r_x \in \mathbb{Z}_p$. It outputs the private key as $\text{sk}_{(\text{id}, (M, \pi))} = ((D_{x,i}^{(1)}, D_{x,i}^{(2)})_{x \in \text{Path}(\text{id}), i \in [1, \ell]}, (D_x^{(3)}, D_x^{(4)})_{x \in \text{Path}(\text{id})})$ where

$$\begin{aligned} D_{x,i}^{(1)} &= g^{\lambda_{x,i}} F(\pi(i))^{r_{x,i}}, & D_{x,i}^{(2)} &= g^{r_{x,i}}, \\ D_x^{(3)} &= g^{f_x(x)} P(x)^{r_x}, & D_x^{(4)} &= g^{r_x}. \end{aligned} \tag{2}$$

- **KeyUpdate(R, t, msk, pk)**: The algorithm first runs $\text{Cover}(R)$ to find a minimal node set that covers $\mathcal{U} \setminus R$. For each $x \in \text{Cover}(R)$, it randomly chooses $r_x \in \mathbb{Z}_p$ and sets the key update material as $\text{uk}_{(R, t)} = (U_x^{(1)}, U_x^{(2)})_{x \in \text{Cover}(R)}$ where

$$U_x^{(1)} = g^{f_x(t)} P(t)^{r_x}, \quad U_x^{(2)} = g^{r_x}. \tag{3}$$

- **Decrypt(ct, (mode, a, ω), sk_(id, A), (id, A), pk, b)**: Suppose that ω satisfies (M, π) and $\text{id} \notin R$. (So that the decryption is possible). Let $I = \{i \mid \pi(i) \in \omega\}$. It then calculates the corresponding set of reconstruction constants $\{(i, \nu_i)\}_{i \in I} = \text{Recon}_{(M, \pi)}(\omega)$. Since $\text{id} \notin R$, it also finds a node x such that $x \in \text{Path}(\text{id}) \cap \text{Cover}(R)$. Then it computes the following

$$K' = \prod_{i=1}^{\ell} \left(\frac{e(D_{x,i}^{(1)}, C^{(1)})}{e(C_{\pi(i)}^{(2)}, D_{x,i}^{(2)})} \right)^{\nu_i}.$$

From this point, two modes diverges as follows.

$$K = \begin{cases} (K')^{\frac{x}{x-1}} \cdot \left(\frac{e(D_x^{(3)}, C^{(1)})}{e(C_x^{(3)}, D_x^{(4)})} \right)^{\frac{1}{1-x}} & \text{if mode} = \text{'dir'} \\ (K')^{\frac{t}{t-1}} \cdot \left(\frac{e(U_x^{(1)}, C^{(1)})}{e(C^{(3)}, U_x^{(2)})} \right)^{\frac{1}{1-t}} & \text{if mode} = \text{'ind'} \end{cases}$$

Finally, it obtains message $m = C/K$.

Correctness. We first claim that $K' = e(g, g)^{sf_x(1)}$, which can be verified as follows.

$$K' = \prod_{i=1}^{\ell} \left(\frac{e(g^{\lambda_{x,i}} F(\pi(i))^{r_{x,i}}, g^s)}{e(F(\pi(i))^s, g^{r_{x,i}})} \right)^{\nu_i} = \prod_{i=1}^{\ell} e(g, g)^{s\lambda_{x,i}\nu_i} = e(g, g)^{sf_x(1)},$$

where the first equality is due to the construction, the second one is from the property of bilinear map, and the third one is due to the linear reconstruction property of linear secret sharing scheme.

Now for direct revocation mode, we can verify its correctness as follows.

$$\begin{aligned} K &= (K')^{\frac{x}{x-1}} \cdot \left(\frac{e(g^{f_x(x)} P(x)^{r_x}, g^s)}{e(P(x)^s, g^{r_x})} \right)^{\frac{1}{1-x}} \\ &= (e(g, g)^{sf_x(1)})^{\frac{x}{x-1}} \cdot (e(g, g)^{sf_x(x)})^{\frac{1}{1-x}} = e(g, g)^{s\alpha}, \end{aligned}$$

which is indeed the Lagrange interpolation from two points $(1, f_x(1)), (x, f_x(x))$ to evaluate $f_x(0) = \alpha$ (in the exponent) as in Eq.(1).

Similarly, for indirect mode, we can verify its correctness as follows.

$$\begin{aligned} K &= (K')^{\frac{t}{t-1}} \cdot \left(\frac{e(g^{f_x(t)} P(t)^{r_x}, g^s)}{e(P(t)^s, g^{r_x})} \right)^{\frac{1}{1-t}} \\ &= (e(g, g)^{sf_x(1)})^{\frac{t}{t-1}} \cdot (e(g, g)^{sf_x(t)})^{\frac{1}{1-t}} = e(g, g)^{s\alpha}, \end{aligned}$$

which is indeed the Lagrange interpolation from two points $(1, f_x(1)), (t, f_x(t))$ to evaluate $f_x(0) = \alpha$ (in the exponent) as in Eq.(1).

Theorem 1. *If an adversary can break the above scheme with advantage ϵ in the sense of IND-sHRSS, then a simulator with advantage ϵ in solving the DBDH problem can be constructed.*

Discussion on the Revocable ABE of [6]. The indirectly revocable ABE of Boldyreva et al. [6] can be derived from our scheme by just neglecting the direct mode and deleting the term $(D_x^{(3)}, D_x^{(4)})$ from the private key. Note that such a scheme was not explicitly given in their paper though. We hence try to prove the security of their implicit scheme by ourselves in the adaptive query model. We found that by extending their proof from the IBE case to the ABE case, the security reduction is lost by multiplicative factor of $O(1/2^q)$, where q is the

number of queries to either secret key or update key oracle. Intuitively, this is due to the fact that the simulator in the proof has to guess whether a user will be critical user or revoked user at the target time t^* (see discussion in Section §3.2), in order to simulate answers to queries. If only one guess fails, it will abort. The simulation will thus succeed with only probability $1/2^q$. Since q is polynomial in security parameter, this loss is exponential. Therefore, the security in the adaptive query model would not hold for their scheme. We conclude here that their revocable ABE is proved secure only in the semi-static query model. Note that in their IBE case, the reduction cost is only $1/2$ since the simulator only needs to guess once for the only one critical user candidate, which is the challenge identity (denoted by ω^* in their paper).

Some Extensions. Note that n is fixed in our main scheme but can be extended as suggested in [6] by adding a new root over the old one, hence the universe becomes of size $2n$. The new key at the new root is distributed via key update. The chosen-ciphertext secure scheme can also be obtained similarly as in [6].

5 Security Proof

In this section, we give the security proof of our scheme as stated in Theorem 1.

Proof. Suppose there exists an adversary, \mathcal{A} , that has advantage ϵ in attacking the proposed scheme. We build a simulator \mathcal{B} that solves the Decision BDH problem in \mathbb{G} . \mathcal{B} is given as input a random DBDH challenge $\mathbf{Y} = (g, g^a, g^b, g^s, Z)$, where Z is either $e(g, g)^{abs}$ or a random element in \mathbb{G}_1 . \mathcal{B} proceeds as follows.

Init. The selective-target game begins with \mathcal{A} first outputting a target $(\text{mode}^*, \mathbf{a}^*, \omega^*)$ that it intends to attack. The proof for two cases of mode^* diverges here.

5.1 When Target Mode Is Direct Revocation

We first consider the case where $\text{mode}^* = \text{'dir'}$. In this case, we have the auxiliary input $\mathbf{a}^* = R^*$.

Setup. To generate pk , algorithm \mathcal{B} first defines

$$q(x) = x^{m - |\omega^*|} \cdot \prod_{k \in \omega^*} (x - k) = \sum_{j=0}^m q_j x^j.$$

We note that q_j 's terms can be computed completely from ω^* . From this we can ensure that $q(x) = 0$ if and only if $x \in \omega^*$. It then randomly picks a degree m polynomial in $\mathbb{Z}_p[x]$ as $\phi(x) = \sum_{j=0}^m \phi_j x^j$. It lets $h_j = (g^a)^{q_j} g^{\phi_j}$ for $j = 0, \dots, m$. We thus have $F(x) = \prod_{j=0}^m h_j^{(x^j)} = (g^a)^{q(x)} \cdot g^{\phi(x)}$.

Similarly, it also defines

$$p(y) = y^{d - |\text{Cover}(R^*)|} \cdot \prod_{x \in \text{Cover}(R^*)} (y - x) = \sum_{j=0}^d p_j y^j.$$

From this we can ensure that for all $x \in \mathcal{X}$, $p(x) = 0$ if and only if $x \in \text{Cover}(R^*)$ and that for all $t \in \mathcal{T}$, $p(t) \neq 0$. The latter is since $\mathcal{X} \cap \mathcal{T} = \emptyset$.

It then randomly picks a degree d polynomial in $\mathbb{Z}_p[x]$ as $\rho(y) = \sum_{j=0}^d \rho_j y^j$. It lets $u_j = (g^a)^{p_j} g^{\rho_j}$ for $j = 0, \dots, d$. We thus have $P(y) = \prod_{j=0}^d u_j^{(y^j)} = (g^a)^{p(y)} \cdot g^{\rho(y)}$.

The algorithm \mathcal{B} implicitly defines $\alpha = ab$. It gives \mathcal{A} the public key $\mathbf{pk} = (g, e(g^a, g^b), u_0, \dots, u_d, h_0, \dots, h_m)$. Since g, a, b, ϕ, ρ are chosen randomly and independently, \mathbf{pk} has an identical distribution to that in the actual construction.

Let $\mathcal{X}_{R^*} = \{x \in \text{Path}(\mathbf{id}) \mid \mathbf{id} \in R^*\}$. For all node x in the binary tree, it randomly chooses $a'_x \in \mathbb{Z}_p$ and implicitly pre-defines:

$$a_x = \begin{cases} a'_x - ab & \text{if } x \in \mathcal{X}_{R^*} \\ a'_x - \frac{ab}{x} & \text{if } x \notin \mathcal{X}_{R^*}. \end{cases} \quad (4)$$

Note that the simulator cannot compute these values, since ab is unknown. Intuitively, predefining is done so that we can compute

$$f_x(1) = a_x(1) + ab = (a'_x - ab) + ab = a'_x \quad \text{if } x \in \mathcal{X}_{R^*} \quad (5)$$

$$f_x(x) = a_x(x) + ab = (a'_x - \frac{ab}{x})x + ab = a'_x x \quad \text{if } x \notin \mathcal{X}_{R^*} \quad (6)$$

Query Phase. The adversary makes requests for private keys corresponding to user index and access structure pair $(\mathbf{id}, (M, \pi))$ subjected to condition that ω^* does not satisfy (M, π) or $\mathbf{id} \in R^*$. As a big picture, the algorithm \mathcal{B} simulates answers as follows.

<p>If \mathcal{A} queries for $\mathcal{SK}(\mathbf{id}, \mathbb{A}_{(M, \pi)})$ then</p> <ul style="list-style-type: none"> If $\omega^* \in \mathbb{A}_{(M, \pi)}$ then <ul style="list-style-type: none"> If $\mathbf{id} \notin R^*$ then return \perp Else do Case S1 If $\omega^* \notin \mathbb{A}_{(M, \pi)}$ then <ul style="list-style-type: none"> Do Case S2 	<p>If \mathcal{A} queries for $\mathcal{UK}(R, t)$ then</p> <ul style="list-style-type: none"> Do Case U
---	--

[**Case S1:** $\omega^* \in \mathbb{A}_{(M, \pi)}$ and $\mathbf{id} \in R^*$] First we claim that in this case we can compute $f_x(1)$ for all $x \in \text{Path}(\mathbf{id})$. This is since $\mathbf{id} \in R^*$, hence for all $x \in \text{Path}(\mathbf{id})$ we have $x \in \mathcal{X}_{R^*}$. Hence $f_x(1)$ can be computed by Eq.(5).

To create $(D_{x,i}^{(1)}, D_{x,i}^{(2)})_{i \in [1, \ell], x \in \text{Path}(\mathbf{id})}$, algorithm \mathcal{B} just computes as in the real construction since it knows $f_x(1)$.

To create $(D_x^{(3)}, D_x^{(4)})_{x \in \text{Path}(\mathbf{id})}$, it does as follows. For all $x \in \text{Path}(\mathbf{id})$, it randomly chooses $r'_x \in \mathbb{Z}_p$ and computes

$$D_x^{(3)} = (g^{a'_x})^x (g^b)^{-\frac{(1-x)\rho(x)}{p(x)}} P(x)^{r'_x}, \quad D_x^{(4)} = (g^b)^{-\frac{(1-x)}{p(x)}} g^{r'_x}.$$

Note that these can be computed since, in particular, $p(x) \neq 0$ due to the fact that for all $x \in \mathcal{X}$, $p(x) = 0$ iff $x \in \text{Cover}(R^*)$ but here since $x \in \mathcal{X}_{R^*}$ we

have $x \notin \text{Cover}(R^*)$. We claim that $(D_x^{(3)}, D_x^{(4)})$ is valid by implicitly letting $r_x = r'_x - \frac{b(1-x)}{p(x)}$ and seeing that

$$\begin{aligned} D_x^{(3)} &= (g^{a'_x})^x (g^b)^{\left(-\frac{(1-x)\rho(x)}{p(x)}\right)} P(x)^{r'_x} \\ &= (g^{a'_x x + (1-x)ab}) (g^{ab})^{-(1-x)} (g^b)^{\left(-\frac{(1-x)\rho(x)}{p(x)}\right)} P(x)^{r'_x} \\ &= (g^{f_x(x)}) ((g^a)^{p(x)} g^{\rho(x)})^{\left(-\frac{b(1-x)}{p(x)}\right)} P(x)^{r'_x} = g^{f_x(x)} P(x)^{r_x}. \end{aligned}$$

The validity of $D_x^{(4)}$ is immediate.

[Case S2: $\omega^* \notin \mathbb{A}_{(M, \pi)}$] To create $((D_{x,i}^{(1)}, D_{x,i}^{(2)}),_{i \in [1, \ell]}, (D_x^{(3)}, D_x^{(4)}))_{x \in \text{Path}(\text{id})}$, we categorize node $x \in \text{Path}(\text{id})$ whether it is in \mathcal{X}_{R^*} or not as follows.

Category 1: $x \in \text{Path}(\text{id}) \cap \mathcal{X}_{R^*}$. The simulator \mathcal{B} can create the key in exactly the same manner as in the case **S1**, since in this category $x \in \mathcal{X}_{R^*}$.

Category 2: $x \in \text{Path}(\text{id}) \setminus \mathcal{X}_{R^*}$. The simulator \mathcal{B} does as follows. Recall that $\omega^* \notin \mathbb{A}_{(M, \pi)}$. Hence from Lemma 1, there must exist a vector $\mathbf{w} = (w_1, \dots, w_k) \in \mathbb{Z}_p^k$ such that $w_1 = 1$ and that for all i where $\pi(i) \in \omega^*$, it holds that $\mathbf{M}_i \cdot \mathbf{w} = 0$.

To create $(D_{x,i}^{(1)}, D_{x,i}^{(2)}),_{i \in [1, \ell]}$, it randomly chooses $z'_{x,2}, \dots, z'_{x,k} \in \mathbb{Z}_p$ and lets $\mathbf{v}'_x = (0, z'_{x,2}, \dots, z'_{x,k})$. It implicitly defines a vector

$$\mathbf{v}_x = (a_x + \alpha)\mathbf{w} + \mathbf{v}'_x \stackrel{(4)}{=} \left(a'_x + \alpha\left(1 - \frac{1}{x}\right)\right)\mathbf{w} + \mathbf{v}'_x,$$

which will be used for creating shares of $f_x(1) = a_x + \alpha$ as in the construction. For simplicity, let $A = 1 - \frac{1}{x}$.

For i where $\pi(i) \in \omega^*$, it randomly chooses $r_{x,i} \in \mathbb{Z}_p$ and computes

$$D_{x,i}^{(1)} = g^{\mathbf{M}_i \cdot \mathbf{v}'_x} F(\pi(i))^{r_{x,i}} = g^{\mathbf{M}_i \cdot \mathbf{v}_x} F(\pi(i))^{r_{x,i}}, \quad D_{x,i}^{(2)} = g^{r_{x,i}} \quad (7)$$

where the right-hand equality of $D_i^{(1)}$ is due to $\mathbf{M}_i \cdot \mathbf{w} = 0$.

For i where $\pi(i) \notin \omega^*$, it randomly chooses $r'_{x,i} \in \mathbb{Z}_p$. It then computes

$$\begin{aligned} D_{x,i}^{(1)} &= (g^{a'_x})^{\mathbf{M}_i \cdot \mathbf{w}} g^{\mathbf{M}_i \cdot \mathbf{v}'_x} (g^b)^{\left(-\frac{A(\mathbf{M}_i \cdot \mathbf{w})\phi(\pi(i))}{q(\pi(i))}\right)} F(\pi(i))^{r'_{x,i}}, \\ D_{x,i}^{(2)} &= (g^b)^{-\frac{A(\mathbf{M}_i \cdot \mathbf{w})}{q(\pi(i))}} g^{r'_{x,i}}. \end{aligned} \quad (8)$$

Note that these can be computed since, in particular, $q(\pi(i)) \neq 0$ due to the fact that $q(x) = 0$ iff $x \in \omega^*$ and here $\pi(i) \notin \omega^*$. We claim that $(D_{x,i}^{(1)}, D_{x,i}^{(2)})$ is valid by implicitly letting $r_{x,i} = r'_{x,i} - b \frac{A(\mathbf{M}_i \cdot \mathbf{w})}{q(\pi(i))}$ and seeing that

$$\begin{aligned} D_{x,i}^{(1)} &= (g^{a'_x})^{\mathbf{M}_i \cdot \mathbf{w}} g^{\mathbf{M}_i \cdot \mathbf{v}'_x} (g^b)^{\left(-\frac{A(\mathbf{M}_i \cdot \mathbf{w})\phi(\pi(i))}{q(\pi(i))}\right)} F(\pi(i))^{r'_{x,i}} \\ &= (g^{a'_x + abA})^{\mathbf{M}_i \cdot \mathbf{w}} g^{\mathbf{M}_i \cdot \mathbf{v}'_x} (g^{ab})^{-A(\mathbf{M}_i \cdot \mathbf{w})} (g^b)^{\left(-\frac{A(\mathbf{M}_i \cdot \mathbf{w})\phi(\pi(i))}{q(\pi(i))}\right)} F(\pi(i))^{r'_{x,i}} \\ &= g^{\mathbf{M}_i \cdot \mathbf{v}_x} ((g^a)^{q(\pi(i))} g^{\phi(\pi(i))})^{-\frac{bA(\mathbf{M}_i \cdot \mathbf{w})}{q(\pi(i))}} F(\pi(i))^{r'_{x,i}} = g^{\mathbf{M}_i \cdot \mathbf{v}_x} F(\pi(i))^{r_{x,i}}, \end{aligned}$$

and the term $D_{x,i}^{(2)}$ is immediate.

To create $(D_x^{(3)}, D_x^{(4)})$, it randomly chooses $r_x \in \mathbb{Z}_p$ and computes

$$D_x^{(3)} = (g^{a'_x} x) P(x)^{r_x}, \quad D_x^{(4)} = g^{r_x},$$

which is valid since $f_x(x) = a'_x x$ due to Eq.(6).

[Case U] For any R, t , it computes $\text{uk}_{(R,t)} = (U_x^{(1)}, U_x^{(2)})_{x \in \text{Cover}(R)}$ as follows. It first randomly chooses $r'_x \in \mathbb{Z}_p$ and computes

$$\begin{aligned} \text{If } x \in \mathcal{X}_{R^*} & \begin{cases} U_x^{(1)} = (g^{a'_x})^t (g^b)^{(-\frac{(1-t)\rho(t)}{p(t)})} P(t)^{r'_x}, \\ U_x^{(2)} = (g^b)^{-\frac{(1-t)}{p(t)}} g^{r'_x}. \end{cases} \\ \text{If } x \notin \mathcal{X}_{R^*} & \begin{cases} U_x^{(1)} = (g^{a'_x})^t (g^b)^{(-\frac{(x-t)\rho(t)}{xp(t)})} P(t)^{r'_x}, \\ U_x^{(2)} = (g^b)^{-\frac{(x-t)}{xp(t)}} g^{r'_x}. \end{cases} \end{aligned}$$

Note that these can be computed since, in particular, $p(t) \neq 0$ for any $t \in \mathcal{T}$.

The term $(U_x^{(1)}, U_x^{(2)})$ can be proved valid with implicit randomness $r_x = r'_x - \frac{b(1-t)}{p(t)}$ for the case $x \in \mathcal{X}_{R^*}$ and $r_x = r'_x - \frac{(x-t)}{xp(t)}$ for the case $x \notin \mathcal{X}_{R^*}$.

Challenge. The adversary gives two message M_0, M_1 to the simulator. The simulator flips a coin b and creates $C = M_b \cdot Z$, $C^{(1)} = g^s$, for $k \in \omega^*$, $C_k^{(2)} = (g^s)^{\phi(k)}$, and for $x \in \text{Cover}(R^*)$, $C_x^{(3)} = (g^s)^{\rho(x)}$. We claim that if $Z = e(g, g)^{\text{abs}}$, then the above ciphertext is a valid challenge. The term $C, C^{(1)}$ is trivial. For all $k \in \omega^*$, we have $q(k) = 0$, hence

$$C_k^{(2)} = (g^s)^{\phi(k)} = ((g^a)^{q(k)} g^{\phi(k)})^s = F(k)^s.$$

Also, for $x \in \text{Cover}(R^*)$ we have $p(x) = 0$, hence

$$C_x^{(3)} = (g^s)^{\rho(x)} = ((g^a)^{p(x)} g^{\rho(x)})^s = P(x)^s,$$

which concludes our claim.

5.2 When Target Mode Is Indirect Revocation

We now consider the case where $\text{mode}^* = \text{'ind'}$. In this case, we have the auxiliary input $a^* = t^*$.

Setup. To generate pk , algorithm \mathcal{B} first defines $q(x), \phi(x)$ and corresponding h_j 's as in the previous mode. Similarly, it also defines

$$p(y) = y^{d-1} \cdot (y - t^*) = \sum_{j=0}^d p_j y^j.$$

From this we can ensure that for all $t \in \mathcal{T}$, $p(t) = 0$ if and only if $t = t^*$ and that for $x \in \mathcal{X}$, $p(x) \neq 0$. The latter is due to $\mathcal{T} \cap \mathcal{X} = \emptyset$.

It then randomly picks a degree d polynomial in $\mathbb{Z}_p[x]$ as $\rho(y) = \sum_{j=0}^d \rho_j y^j$. It lets $u_j = (g^a)^{p_j} g^{\rho_j}$ for $j = 0, \dots, d$. We thus have $P(y) = \prod_{j=0}^d u_j^{(y^j)} = (g^a)^{p(y)} \cdot g^{\rho(y)}$.

The algorithm \mathcal{B} implicitly defines $\alpha = ab$. It gives \mathcal{A} the public key $\mathsf{pk} = (g, e(g^a, g^b), u_0, \dots, u_d, h_0, \dots, h_m)$. Since g, a, b, ϕ, ρ are chosen randomly and independently, pk has an identical distribution to that in the actual construction.

Query Phase. Since we deal with the semi-static query notion, at the beginning of this phase the adversary \mathcal{A} announces \tilde{R} . Let $\mathcal{X}_{\tilde{R}} = \{x \in \text{Path}(\text{id}) \mid \text{id} \in \tilde{R}\}$. For all node x in the binary tree, it randomly chooses $a'_x \in \mathbb{Z}_p$ and implicitly pre-defines:

$$a_x = \begin{cases} a'_x - ab & \text{if } x \in \mathcal{X}_{\tilde{R}} \\ a'_x - \frac{ab}{t^*} & \text{if } x \notin \mathcal{X}_{\tilde{R}} \end{cases} \quad (9)$$

Note that the simulator cannot compute these values, since ab is unknown. Intuitively, predefining is done so that we can compute

$$f_x(1) = a_x(1) + ab = (a'_x - ab) + ab = a'_x \quad \text{if } x \in \mathcal{X}_{\tilde{R}} \quad (10)$$

$$f_x(t^*) = a_x(t^*) + ab = (a'_x - \frac{ab}{t^*})t^* + ab = a'_x t^* \quad \text{if } x \notin \mathcal{X}_{\tilde{R}} \quad (11)$$

The algorithm \mathcal{B} then simulates answers to queries as follows.

<p>If \mathcal{A} queries for $\mathcal{SK}(\text{id}, \mathbb{A})$ then</p> <ul style="list-style-type: none"> If $\omega^* \in \mathbb{A}_{(M, \pi)}$ then If $\text{id} \notin \tilde{R}$ then return \perp Else do Case S1' If $\omega^* \notin \mathbb{A}_{(M, \pi)}$ then Do Case S2' 	<p>If \mathcal{A} queries for $\mathcal{UK}(R, t)$ then</p> <ul style="list-style-type: none"> If $t = t^*$ then If $\tilde{R} \not\subseteq R$ then return \perp Else do Case U1' If $t \neq t^*$ then Do Case U2'
--	--

[**Case S1':** $\omega^* \in \mathbb{A}_{(M, \pi)}$ and $\text{id} \in \tilde{R}$] The algorithm \mathcal{B} computes $(D_{x,i}^{(1)}, D_{x,i}^{(2)})_{i \in [1, \ell]}$, $D_x^{(3)}, D_x^{(4)}$ for all $x \in \text{Path}(\text{id})$ in exactly the same manner as in the case **S1** albeit using set \tilde{R} instead of R^* and using polynomial p, ρ defined in Setup of this mode above.

To see why this is valid, notice that in this case for all $x \in \text{Path}(\text{id})$ we have $x \in \mathcal{X}_{\tilde{R}}$, where in such a case we define $a_x = a'_x - ab$ in Eq.(9), which is unchanged from the previous mode.

[**Case S2':** $\omega^* \notin \mathbb{A}_{(M, \pi)}$] To create $((D_{x,i}^{(1)}, D_{x,i}^{(2)})_{i \in [1, \ell]}, (D_x^{(3)}, D_x^{(4)}))_{x \in \text{Path}(\text{id})}$, we categorize node $x \in \text{Path}(\text{id})$ whether it is in $\mathcal{X}_{\tilde{R}}$ or not as follows.

Category 1: $x \in \text{Path}(\text{id}) \cap \mathcal{X}_{\tilde{R}}$. The simulator \mathcal{B} can create the key in exactly the same manner as in the case **S1'**, since in this category $x \in \mathcal{X}_{\tilde{R}}$.

Category 2: $x \in \text{Path}(\text{id}) \setminus \mathcal{X}_{\tilde{R}}$. The simulator \mathcal{B} does as follows. Recall that $\omega^* \notin \mathbb{A}_{(M, \pi)}$. Hence from Lemma 1, there must exist a vector $\mathbf{w} = (w_1, \dots, w_k) \in \mathbb{Z}_p^k$ such that $w_1 = 1$ and that for all i where $\pi(i) \in \omega^*$, it holds that $\mathbf{M}_i \cdot \mathbf{w} = 0$.

To create $(D_{x,i}^{(1)}, D_{x,i}^{(2)})_{i \in [1,\ell]}$, similarly to the case **S2**, the simulator \mathcal{B} does as follows. It randomly chooses $z'_{x,2}, \dots, z'_{x,k} \in \mathbb{Z}_p$ and lets $\mathbf{v}'_x = (0, z'_{x,2}, \dots, z'_{x,k})$. It implicitly defines a vector

$$\mathbf{v}_x = (a_x + \alpha)\mathbf{w} + \mathbf{v}'_x \stackrel{(9)}{=} \left(a'_x + \alpha\left(1 - \frac{1}{t^*}\right)\right)\mathbf{w} + \mathbf{v}'_x,$$

which will be used for creating shares of $f_x(1) = a_x + \alpha$ as in the construction. For simplicity, let $A = 1 - \frac{1}{t^*}$. The computation of $(D_{x,i}^{(1)}, D_{x,i}^{(2)})_{i \in [1,\ell]}$ can be done using Eq.(7) and Eq.(8) as in the case **S2**, here the only difference is the value of A . The validity thus can be verified similarly.

To create $(D_x^{(3)}, D_x^{(4)})$, it randomly chooses $r_x \in \mathbb{Z}_p$ and computes

$$D_x^{(3)} = (g^{a'_x})^x (g^b)^{\left(-\frac{(t^*-x)\rho(x)}{t^*p(x)}\right)} P(x)^{r'_x}, \quad D_x^{(4)} = (g^b)^{-\frac{(t^*-x)}{t^*p(x)}} g^{r'_x},$$

Note that these can be computed since, in particular, $p(x) \neq 0$ for any $x \in \mathcal{X}$. This can be proved valid with the implicit randomness $r_x = r'_x - \frac{b(t^*-x)}{t^*p(x)}$.

[Case U1': $t = t^*$ and $\tilde{R} \subseteq R$.] To create $(U_x^{(1)}, U_x^{(2)})_{x \in \text{Cover}(R)}$, \mathcal{B} randomly chooses $r_x \in \mathbb{Z}_p$ for each $x \in \text{Cover}(R)$ and computes

$$U_x^{(1)} = (g^{a'_x t^*}) P(t^*)^{r_x}, \quad U_x^{(2)} = g^{r_x}.$$

We prove that this is valid as follows. Since $\tilde{R} \subseteq R$, hence for all $x \in \text{Cover}(R)$ we have $x \notin \mathcal{X}_{\tilde{R}}$. Therefore from Eq.(11), we have $f_x(t^*) = a'_x t^*$.

[Case U2': $t \neq t^*$.] To create $(U_x^{(1)}, U_x^{(2)})_{x \in \text{Cover}(R)}$, \mathcal{B} randomly chooses $r'_x \in \mathbb{Z}_p$ for each $x \in \text{Cover}(R)$ and computes

$$\begin{aligned} \text{If } x \in \text{Cover}(R) \cap \mathcal{X}_{\tilde{R}} & \begin{cases} U_x^{(1)} = (g^{a'_x})^t (g^b)^{\left(-\frac{(1-t)\rho(t)}{p(t)}\right)} P(t)^{r'_x}, \\ U_x^{(2)} = (g^b)^{-\frac{(1-t)}{p(t)}} g^{r'_x}. \end{cases} \\ \text{If } x \in \text{Cover}(R) \setminus \mathcal{X}_{\tilde{R}} & \begin{cases} U_x^{(1)} = (g^{a'_x})^t (g^b)^{\left(-\frac{(t^*-t)\rho(t)}{t^*p(t)}\right)} P(t)^{r'_x}, \\ U_x^{(2)} = (g^b)^{-\frac{(t^*-t)}{t^*p(t)}} g^{r'_x}. \end{cases} \end{aligned}$$

Note that these can be computed since, in particular, $p(t) \neq 0$ due to the fact that for $t \in \mathcal{T}$, $p(t) = 0$ iff $t = t^*$.

The term $(U_x^{(1)}, U_x^{(2)})$ can be proved valid with implicit randomness $r_x = r'_x - \frac{b(1-t)}{p(t)}$ for the case $x \in \mathcal{X}_{R^*}$ and $r_x = r'_x - \frac{(t^*-t)}{t^*p(t)}$ for the case $x \notin \mathcal{X}_{R^*}$.

Challenge. The adversary gives two message M_0, M_1 to the simulator. The simulator flips a coin b and creates $C = M_b \cdot Z$, $C^{(1)} = g^s$, for $k \in \omega^*$, $C_k^{(2)} = (g^s)^{\phi(k)}$, and $C^{(3)} = (g^s)^{\rho(t^*)}$. We claim that if $Z = e(g, g)^{\text{abs}}$, then the above ciphertext is a valid challenge. The term $C, C^{(1)}$ is trivial. For all $k \in \omega^*$, we have $q(k) = 0$, hence

$$C_k^{(2)} = (g^s)^{\phi(k)} = ((g^a)^{q(k)} g^{\phi(k)})^s = F(k)^s.$$

Also, since $p(t^*) = 0$, hence

$$C^{(3)} = (g^s)^{\rho(t^*)} = ((g^a)^{p(t^*)} g^{\rho(t^*)})^s = P(t^*)^s,$$

which concludes our claim.

5.3 The Rest of the Proof

Guess. In either mode, finally, \mathcal{A} outputs $b' \in \{0, 1\}$ for the guess of b . If $b = b'$ then \mathcal{B} outputs 1 (meaning $Z = e(g, g)^{abs}$). Otherwise, it outputs 0 (meaning Z is random in \mathbb{G}_T).

We see that in both modes, the simulation is perfect. Furthermore, if \mathbf{Y} is sampled from \mathcal{R}_{BDH} then $\Pr[\mathcal{B}(\mathbf{Y}) = 0] = \frac{1}{2}$. On the other hand, if \mathbf{Y} is sampled from \mathcal{P}_{BDH} then we have $|\Pr[\mathcal{B}(\mathbf{Y}) = 0] - \frac{1}{2}| \geq \epsilon$. It follows that \mathcal{B} has advantage at least ϵ in solving the DBDH problem in \mathbb{G} . This concludes the proof.

6 Efficiency

6.1 Comparison to Schemes Secure under the DBDH Assumption

In this section, we compare efficiency among available revocable ABE schemes in which security is based on the DBDH assumption. This is shown in Table 2.

Table Description. Denote by $|\text{sk}|, |\text{ct}|, |\text{uk}|, |\text{pk}|$ the size of user secret key, ciphertext overhead, update key, and public key, respectively. The columns Normal, Direct revoke, and Indirect revoke show the communication cost in the respective modes. (Normal means no user is revoked). Here r is the number of revoked user. n is the number of all users. ℓ is the size of rows in the LSSS matrix, which is equal to the number of attributes presented in the access structure. k is the size of attribute set. m is maximum size allowed for k . d is maximum size of $\text{Cover}(R)$ for all $R \subseteq \mathcal{U}$. All values in the table show the amount of group elements in \mathbb{G} . Symbol ‘-’ denotes unavailability of that mode.

DBDH-based Directly Revocable ABE. As stated in Section §1.2, there is no available directly revocable ABE which is proved secure under the DBDH assumption yet. We briefly show how to construct one here. We use the methodology in [3] for combining broadcast encryption (BE) and ABE. In their paper, they combine BE of [8] and [21] to ABE of [16], yielding two efficient directly revocable ABE systems, albeit they are based on much stronger assumptions (the n -BDHE and q -MEBDH respectively). If we use this methodology to combine BE of [10], in which security is based on DBDH, and ABE of [16], it will yield a DBDH-based directly revocable ABE with parameters in the Table 2 (Variant of [3]).

Comparison. From Table 2, one can see that our HR-ABE scheme has the key size roughly the same as both one-mode revocable ABES (and hence half size of the trivial HR-ABE from their combination). Indeed only $2 \log(n)$ is increased from both one-mode schemes. The ciphertext size in direct mode is the same as the

Table 2. Comparison among available (revocable) ABE schemes based on the DBDH assumption

Scheme	$ \text{sk} $	Normal	Direct revoke mode	Indirect revoke mode		$ \text{pk} $
		$ \text{ct} $	$ \text{ct} $	$ \text{ct} $	$ \text{uk} $	
[16]	2ℓ	$k + 1$	-	-	-	$m + 3$
[6]	$2\ell \log(n) + 1$	$k + 2$	-	$k + 2$	$2(r \log(\frac{n}{r}) + r)$	$m + 3$
[3] (variant)	$2\ell \log(n)$	$k + 2$	$k + r \log(\frac{n}{r}) + r + 1$	-	-	$m + d + 3$
Ours	$2(\ell + 1) \log(n)$	$k + 2$	$k + r \log(\frac{n}{r}) + r + 1$	$k + 2$	$2(r \log(\frac{n}{r}) + r)$	$m + d + 3$

Table 3. Efficiency of available directly revocable ABE based on stronger assumptions

Scheme	$ \text{sk} $	Normal $ \text{ct} $	Direct revoke mode $ \text{ct} $	$ \text{pk} $	Assumption
[3] (scheme 1)	2ℓ	$k + 2$	$k + 2$	$m + 2n + 2$	$n\text{-BDHE}$
[3] (scheme 2)	$2\ell + 2$	$k + 2$	$k + 2r + 1$	$m + 7$	$q\text{-MEBDH}$

variant of [3]. The ciphertext and update key sizes in indirect mode is the same as that of [6]. Hence, we can conclude that among revocable ABE systems that are based on the DBDH assumption, ours is the most flexible since it allows two revocation modes and has efficiency roughly the same as the one-mode schemes.

6.2 Comparisons to Schemes Secure under Stronger Assumptions

In this section, we compare our HR-ABE to the trivially combined schemes between Boldyreva et al. [6] and two directly revocable schemes of Attrapadung-Imai [3], which are based on *much stronger* assumptions, namely the n -BDHE [8] and the q -MEBDH [21] assumptions. Both directly revocable ABE schemes have the private key size about 2ℓ . See Table 3 for their efficiency. Thus, both combined schemes have the private key size about $2\ell \log(n) + 2\ell$. Therefore, our HR-ABE scheme is still more efficient when $\log(n) \leq \ell$. Note also that the first combined scheme ([6] and the first scheme of [3]) is not so efficient in that its public key is large as being linear in n .

7 Concluding Remarks

We presented a formalization and a construction of hybrid revocable attribute-based encryption. An HR-ABE system allows senders to select whether to use either direct or indirect revocation mode when encrypting a message. With direct mode, the sender specifies the list of revoked users directly into the encryption algorithm. With indirect mode, the sender specifies just the encrypt time (besides a usual attribute set), while receivers obtain a key update material at each time slot to update their keys from the authority, albeit in such a way that only non-revoked users are able to update (hence revocation is enforced indirectly). Our HR-ABE scheme requires each receiver to store only one key, which is nonetheless

can be used to decrypt ciphertexts in either mode. The key size in our hybrid scheme is roughly the same as that of the two currently best one-mode revocable schemes. We proved the security for the selective-target and semi-static query model under the DBDH assumption.

As for future direction regarding revocable ABE, it would be interesting to obtain more efficient schemes (*e.g.*, with constant-size keys and/or ciphertexts, update keys). Another direction could be to obtain revocable ABE schemes (one-mode or hybrid) which are secure in the adaptive security model.

References

1. Aiello, W., Lodha, S., Ostrovsky, R.: Fast digital identity revocation (extended abstract). In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 137–152. Springer, Heidelberg (1998)
2. Attrapadung, N., Imai, H.: Dual-policy attribute based encryption. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 168–185. Springer, Heidelberg (2009)
3. Attrapadung, N., Imai, H.: Conjunctive broadcast and attribute-based encryption. In: Boyen, X., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 248–265. Springer, Heidelberg (2009)
4. Libert, B., Vergnaud, D.: Adaptive-ID secure revocable identity-based encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 1–15. Springer, Heidelberg (2009)
5. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy 2007, pp. 321–334 (2007)
6. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: ACM Conference on Computer and Communications Security 2008, pp. 417–426 (2008)
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
8. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
9. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
10. Dodis, Y., Fazio, N.: Public-key broadcast encryption for stateless receivers. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 61–80. Springer, Heidelberg (2003)
11. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
12. Gentry, C.: Certificate-based encryption and the certificate revocation problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
13. Gentry, C., Waters, B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In: Joux, A. (ed.) Eurocrypt 2009. LNCS, vol. 5479, pp. 171–188. Springer, Heidelberg (2009)
14. Gollé, P., Staddon, J., Gagné, M., Rasmussen, P.: A content-driven access control system. In: Symposium on Identity and Trust on the Internet — IDtrust 2008, pp. 26–35 (2008)

15. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute-based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórssón, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
16. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security 2006, pp. 89–98 (2006)
17. Micali, S.: Efficient certificate revocation. Tech. Report MIT/LCS/TM-542b (1996)
18. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
19. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM Conference on Computer and Communications Security 2007, pp. 195–203 (2007)
20. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EU-ROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
21. Sahai, A., Waters, B.: Revocation systems with very small private keys. Cryptology ePrint archive: report 2008/309
22. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. Cryptology ePrint archive: report 2008/290

A Some Further Discussions

A.1 On the Syntax Definition of HR-ABE

In this section, we provide some comparisons on the syntax definition between ours and that of Boldyreva et al. [6].

What to Be Revoked: Key vs Attributes. In a normal ABE system, the private key will be associated with an access structure \mathbb{A} . In general, the private key generation is a randomized algorithm, hence many concrete private keys will correspond to one \mathbb{A} . There are then two approaches for meaningfully defining revocation: (1) to revoke a specific one of many private keys that corresponds to \mathbb{A} or (2) to revoke the access structure \mathbb{A} itself (*i.e.*, to revoke all possible private keys that correspond to \mathbb{A}). Our definition in this paper is of the first type, while the one defined by Boldyreva et al. [6] seems to be of the second type.²

We feel that the type 1 definition is more useful since in practice, the revocation problem is motivated from the scenario where one specific key is leaked to some attacker. Indeed, we do not wish to revoke the access structure \mathbb{A} itself from being able to use ever again in the system.

To meaningfully defining syntax of type 1, we must associate another identity dimension, which we call a serial number (or user index) denoted by id , uniquely to each possible private key for \mathbb{A} . When revoking, a revocator will specify id

² This is only if we guess correctly, since they did not explicitly define the syntax for revocable ABE case in their paper. They did only for revocable IBE case and in this case their formalization is of the second type: to revoke all possible private keys for an identity.

instead of \mathbb{A} itself to distinguish this specific key among all possible keys for \mathbb{A} . More generally, a revocator will specify a set of serial numbers he wants to revoke; any of these may or may not correspond to the same access structures. We capture this type of revocation in our definition in Section §3.1.

On the other hand, Boldyreva et al. [6] provided a definition of the type 2 (for their indirectly revocable IBE but implicitly for the ABE case). In such a system, a revocator will specify the access structure \mathbb{A} he wants to revoke. This kind of definition implies that after being revoked, any private keys for \mathbb{A} (even with different randomness) cannot be used for decryption in the system anymore.

Although the notion of serial number is not included in the syntax definition of [6], it is introduced in their proposed construction and is utilized in a similar way as our explicit use of id in our syntax definition. Hence, the information on the serial number in their scheme is maintained *internally* inside the state of the key authority. Therefore, to be able to include such kind of schemes in the syntax definition, they define the algorithms as stateful type ones. We discuss this issue below.

Algorithms: Stateless vs Stateful. In our main syntax definition of HR-ABE (described in Section §3.1), all algorithms are stateless. We also mentioned a simple augment definition with the stateful KeyGen^s algorithm in order to maintain the assignment of unique id to each key in the real usage.

In contrast, Boldyreva et al. [6] formalized their (indirectly revocable) system as a stateful one in a little bit more complex way. Indeed, their formalization takes into account the notion of time order. In particular, the revoked set R at the present time will be depended on the time t and on all previous possible actions (such as revocation done previously).

We feel that our explicit stateless formalization is more useful in two aspects. First, since the revoked set R is independent from time, an instant or temporary revocation is explicitly possible. Second, it does not take into account the notion of time order, hence t could be any token identity and the scheme will still work.

A.2 On the Security Definition of HR-ABE

We compare our security notion to that of [6] here (for the indirect mode). First, the query model is essentially the same but since in [6] the order of time is important, hence their notion has to check the condition for example $t \leq t^*$ when some queries are asked. In ours, t is an independent attribute, we thus only care about at the target t^* . Second, their notion separates the revocation query and update key query and links them via the internal state. In ours, the revoke set is stated explicitly in the update key query, hence ours is simpler. Third, they only provide an explicit notion for the IBE case but leave the detail for the ABE case to the reader. Finally, ours seems to capture a stronger notion in the sense that the serial number id can be chosen by the adversary when query the \mathcal{SK} oracle. (The last one is due to the difference of the syntax definition types described in Appendix §A.1, though).

Certificate-Free Attribute Authentication

Dalia Khader¹, Liqun Chen², and James H. Davenport¹

¹ University of Bath

{ddk20,jhd}@cs.bath.ac.uk

² Hewlett-Packard Laboratories

liqun.chen@hp.com

Abstract. In this paper we propose the concept of Certificate-free Attribute Authentication (C-AA), which holds a few interesting features, such as a user can demonstrate (1) he owns sufficient attributes to pass an attribute verification without showing his full attribute details, (2) he has been authorized by a number of authorities without revealing his identity, and (3) no certification of the user’s public key is required, and his secret key as a whole is not escrowed by any authority. Although these features have individually been used in various of cryptographic primitives, in this paper we combine them together and demonstrate a Certificate-free Attribute Authentication Scheme (C-AAS) is useful in practice. We provide a formal definition of a C-AAS and four security notions: full anonymity, full traceability, non-frameability and attribute unforgeability. We also construct a concrete C-AAS and prove it is correct and secure under the definition and security notions.

1 Introduction

In lots of applications in practice, people require knowing attributes of their communication partners much more than their communication partners’ identities and other personal information. Let us consider the following Scenario:

Scenario 1. *Bob is a pharmacist. To follow some policies, he only gives a medicine to a patient if s/he has a national health insurance and a valid prescription signed by a doctor, who must be registered with the Ministry of Health (MOH) and works in a known clinic, practice or hospital, etc. Furthermore, Bob is allowed to give a discount for any student, elder, or employee in a health organization. Alice is entitled to obtain the medicine and discount. But she does not feel comfortable to tell Bob her name, age, employment, residential location etc. What they need is a solution to allow Alice to convince Bob that she is eligible to obtain the medicine with the discount, and this solution will not reveal any unnecessary private information about Alice to Bob.*

Our target in this work is to create such a solution.

1.1 Our Contributions

Our proposed solution is the primitive of Certificate-free Attribute Authentication (C-AA), in which a user can demonstrate (1) he owns sufficient attributes

to pass an attribute verification without showing full attribute details, (2) he has been authorized by a number of authorities without revealing his identity, and (3) no certificate of the user's public key is required, and his secret key as a whole is not escrowed by any authority.

For obvious reasons, we assume that Alice in Scenario 1 is not a sole patient in the solution; otherwise maintaining privacy is impossible. In order to make our goal more clear and general, we refer to all potential patients as a group of users \mathcal{U} (or called them signers), and use \mathcal{U}_i to denote an individual user, and we refer to Bob as a verifier \mathcal{V} . We also refer to each condition (e.g., national health insurance, private practice, public practice, student, employee of a health organization, elder and so on) to an attribute denoted by j , refer a set of attributes requested by \mathcal{V} for a particular purpose to an attribute tree Γ , and refer an attribute proof of Γ by \mathcal{U}_i to an attribute signature denoted by $\sigma_i(\Gamma)$. We list all the properties held by a C-AAS as follows:

- Unforgeability of an attribute signature: \mathcal{U}_i cannot provide $\sigma_i(\Gamma)$ if he is not in possession of sufficient attributions required in Γ .
- Coalition Resistant: Two signers, each of which does not own enough attributes to create an attribute signature, cannot jointly provide a valid signature. For instance, in the above scenario, a student without national health insurance and another patient who owns all other attributes but is neither a student, an employee of any health organization nor an elder cannot jointly claim the discount.
- Anonymity of a signer's identity: Given an attribute signature $\sigma_i(\Gamma)$, \mathcal{V} cannot find out who in \mathcal{U} has created it, i.e., the index i cannot be retrieved from $\sigma_i(\Gamma)$ by \mathcal{V} .
- Unlinkability: Given two attribute signatures, \mathcal{V} is not able to find out whether they have been created by the same signer or not.
- Traceability: Given an attribute signature $\sigma_i(\Gamma)$, an authorized entity can retrieve the index i . Traceability is required in order to prevent a legitimate but malicious signer from misusing anonymity.
- Anonymity of Attributes: Given an attribute signature $\sigma_i(\Gamma)$ and Γ , \mathcal{V} cannot find out which set of attributes has been used in computing $\sigma_i(\Gamma)$ if the size of total attributes in Γ is larger than the attribute set required. For instance, in the above scenario, Bob cannot find out whether Alice is an elder, an employee of a health organisation or a student. The details of attribute tree will be described in §4.
- Separability: It is possible for \mathcal{U}_i to obtain different attributes from different authorities, and these authorities do not have to maintain any trust relationship between each other. For instance, in the above scenario, Alice can get a student attribute from her university and get a national health insurance from the government.
- Non-repudiation: \mathcal{U}_i cannot deny a valid attribute signature $\sigma_i(\Gamma)$ has been created by him. In particular, \mathcal{U}_i 's private signing key should not be escrowed by any authority.

These properties have individually been used in various of cryptographic primitives, such as these that will be mentioned in §1.2. In this paper we combine them together to build a C-AA scheme and demonstrate such a scheme could be useful in practice. In the scheme a signing key consists of two parts, one of which is created by a trusted authority and another is created by the signer, and the whole private signing key is not escrowed by any authority. In addition it does not require a traditional key certificate to check validation of the key. These features have some similarity to certificateless cryptography as specified by Al-Riyami and Paterson in [1]. However, the notion of certificate-free used in our C-AAS is not exactly the same as certificateless of Al-Riyami and Paterson defined, since we do not use identity-based key construction. Our key construction is more like an anonymous credential.

We will provide a formal definition and four security notions of C-AAS. The security notions are full anonymity, full traceability, non-frameability and attribute unforgeability. We will also construct a concrete C-AA scheme and prove it is correct and secure under the definition and security notions.

1.2 Related and Prior Work

Researchers have realized the importance of attribute oriented cryptography (e.g. [11,22,17]). There have been a few proposals for attribute based signatures, such as [16,14,20], which do not hold the properties of traceability and certificate-free as we require. In [16], the authors proposed a signature scheme based on mesh signatures [5]. In [14] the authors introduced an attribute based ring signature. Neither of these papers managed to involve multi-authorities for different attributes, and adding multiple attributes to the signing key in these schemes is not straightforward. The authors of [20] proposed a threshold attribute signature scheme where the signer decides their signing identity, given a signing key to a set of attributes from some central authority. Their scheme has limited separation of multiple authorities.

One of the widely studied cryptographic primitive is “anonymous credentials”. Anonymous credential systems were introduced by Chaum [8] and further studied in [6,7,15,21]. A credential in such systems is issued and verified on a user pseudonym, which in turn is bound to the user’s secret identity. Users remain anonymous and unlinkable in such systems. However, attribute anonymity, and traceability are not handled in these systems. Furthermore, most “anonymous credentials” cryptosystems in the literature can not handle multi-credential requirements. More recent proposals are based on ring signatures [14,16], therefore traceability is not an option.

In 2003, Al-Riyami and Paterson proposed the notion of Certificateless Public Key Cryptography (CL-PKC) [1], which was intended to respond to a challenge of key escrow in identity-based cryptography [19]. In CL-PKC, a user has a public key generated by himself and his private key is determined by two pieces of secret information: one secret associated with the user’s identity is issued by a key generation center and the other associated with the public key is generated by the user himself. As a result, the key generation center cannot compute the

private key corresponding to the user's public key. Hence the CL-PKC is key-escrow-free.

Group signatures [2,3] provide somewhat similar features, in allowing people to sign as members of a group, but do not meet our needs since they do not have multiple attribute verification with coalition resistance and attribute anonymity, as described in [13]. These prior works are related to our target in Scenario 1, although none of them alone can meet our needs.

1.3 Paper Organization

The remainder of this paper is organized as follows. In §2, we describe a formal definition of C-AAS and its security model. In §3, we recall the definition of some well-known theorems, security problems and assumptions, which is used to prove security of our C-AA scheme. In §4, we introduce an attribute tree technique, which is a building block of the C-AA scheme. We present the proposed C-AA scheme in §5 and its security analysis in §6. In §7, we discuss the aspect of honestly generated attribute keys. Finally we conclude the paper in §8.

2 Definitions and Security Notions of C-AAS

A C-AA system is shown in Figure 1. In this system, there are six types of entities: a set of Signers (who are also called users \mathcal{U}_i where i is an integer to denote the i -th user of the set), a verifier (\mathcal{V}), an attribute authority (\mathcal{AA}), a central authority (\mathcal{CA}), an opener (\mathcal{O}) and an issuer (\mathcal{I}). In a high level description, a C-AA scheme includes the following nine operation steps (the step numbers match with the ones in Figure 1):

1. \mathcal{CA} creates a general public key known to all other entities, an issuing key for \mathcal{I} and a tracing key for \mathcal{O} . The issuing key is used in creating signing keys. The tracing key is used in tracing signatures. The general public key is used by different entities for different purposes, e.g., \mathcal{AA} uses it to create an attribute public key while \mathcal{V} uses it in the verification process.
2. A join protocol is executed between \mathcal{U}_i and \mathcal{I} . As a result, a private signing key and a registration key are created for \mathcal{U}_i such that nobody other than \mathcal{U}_i will know the full private key, preventing key escrow, but \mathcal{I} holds enough information to help \mathcal{O} to trace the identity of \mathcal{U}_i from his signature.
3. \mathcal{I} sends the registration key to \mathcal{O} , who adds \mathcal{U}_i into the list of possible signers.
4. \mathcal{U}_i sends his registration key to \mathcal{AA} .
5. \mathcal{AA} gives \mathcal{U}_i a set of attribute private keys, each for a particular attribute that \mathcal{U}_i is legitimate to obtain. \mathcal{V} collects all attribute public keys and uses them together with the general public key to generate a verification key.
6. The verification key is made available to all potential signers.
7. Any user with enough attribute private keys can create an attribute-based signature, which can be verified by \mathcal{V} .
8. If \mathcal{V} doubts the signature, he can ask \mathcal{O} to trace it.
9. \mathcal{O} sends a proof back to \mathcal{V} if a legitimate user has signed the signature.



Fig. 1. The Certificate-free Attribute Authentication (C-AA) Scheme

2.1 Formal Definition of a C-AA Scheme

The certificate-free attribute authentication scheme (C-AAS) is a collection of algorithms and protocols that are executed by the entities (signer \mathcal{U}_i , verifier \mathcal{V} , central authority \mathcal{CA} , attribute authority \mathcal{AA} , opener \mathcal{O} and issuer \mathcal{I}). We define the scheme by explaining these algorithms or protocols as follows:

- **KeyGen(k_1)** : This algorithm run by \mathcal{CA} takes the security parameter k_1 as input and outputs three keys: the issuing key isk given to \mathcal{I} , the tracing key tk given to \mathcal{O} , and the general public key gpk known to all. In the description of the remaining algorithms and protocols, we will not repeat that gpk is used as an input.
- **U.KeyGen(k_2)** : This algorithm run by \mathcal{U}_i takes the security parameter k_2 as input and outputs a user public key $upk[i]$ and its corresponding private key $usk[i]$. Note that the security parameters k_1 and k_2 are not logically connected, though it might be wise to choose them together. The choice of k_1 will affect the full anonymity and full traceability games defined later in §2.2. On the other hand, the parameter k_2 should be chosen so that the pair $(upk[i], usk[i])$ can be used in an unforgeable signature scheme.
- **Join($\mathcal{I}(isk)$)** : This protocol is run by \mathcal{I} and \mathcal{U}_i . \mathcal{U}_i uses $(upk[i], usk[i])$ as inputs and \mathcal{I} uses isk as an input. The result of the protocol is a private key $bsk[i]$ known to \mathcal{U}_i only and a registration key A_i saved in a database accessible by \mathcal{O} .
- **A.KeyGen_{pub}(j)** : This algorithm run by \mathcal{AA} takes an attribute j as an input and outputs an attribute public key bpk_j and its corresponding master attribute private key t_j for j .
- **A.KeyGen_{pri}(A_i, j, t_j)** : This algorithm run by \mathcal{AA} takes \mathcal{U}_i 's registration key A_i , an attribute j and its master private key t_j as input, and outputs an attribute private key $T_{i,j}$ for \mathcal{U}_i associated with j . Note that if the size of total attributes owned by \mathcal{U}_i is μ , then the general private key of \mathcal{U}_i is $gsk[i] = (bsk[i], T_{i,1}, \dots, T_{i,\mu})$.

- **Verifign**($\mathcal{U}_i(gsk, M) : \mathcal{V}(M, \overline{B}, gpk)$) : This protocol is run between \mathcal{V} and \mathcal{U}_i . \mathcal{V} uses a set of attribute public keys (i.e. $\overline{B} = \{bpk_1, \dots, bpk_\kappa\}$) as input, creates a tree Γ as described in **TCreate** of **AT** in §4, and sends Γ to \mathcal{U}_i . \mathcal{U}_i uses $gsk[i]$, Γ and a signed message M as input, computes a signature σ , by following the procedure described in **TVerify** of **AT** in §4, and then sends σ to \mathcal{V} . \mathcal{V} checks validity of the signature that proves possession of the attributes represented in Γ . Note that in the security nations and analysis, we use the common notation *Sign* to denote the process of computing σ by \mathcal{U}_i , and *Verify* to denote the process of verifying σ by \mathcal{V} .
- **Open**(σ, tk) : This algorithm run by \mathcal{O} takes tk and σ as inputs and uses the registration key database (updated in the **Join** protocol) to trace the signer’s registration key.
- **Judge**($\mathcal{V}(td) : \mathcal{O}(tk)$) : This protocol is run between \mathcal{V} and \mathcal{O} in order to prove that \mathcal{O} does not frame \mathcal{U}_i and that the **Open** algorithm does in fact trace to this user. The verifier calculates a trapdoor td from the signature σ and sends it to \mathcal{O} , who returns a proof \mathcal{P} that shows \mathcal{U}_i has been traced. Finally the correctness of \mathcal{P} can be verified by using the trapdoor td .

2.2 Security Notions of the C-AA Scheme

We start this section by defining the correctness of a C-AA scheme:

Definition 1. (Correctness of a C-AA scheme): *For all keys gsk generated correctly, every signature generated by a member i verifies as valid unless the member could/did not use sufficient attributes. Every valid signature should trace to a member of the group. In other words,*

$$\begin{aligned} & Verify(\overline{D}_v, gpk, M, Sign(M, \overline{D}_s, gsk)) = \text{valid, where } \overline{D}_s = \overline{D}_v; \\ & \text{if } Verify(\dots) = \text{valid} \text{ then } Open(Sign(M, \overline{D}_s, gsk), tk) = i; \\ & \text{otherwise } Verify(\dots) = \text{not valid}. \end{aligned}$$

In order to capture the properties required in C-AAS specified in Section 1, we define four security notions: full traceability, full anonymity, attribute unforgeability and non-frameability. We argue these four notions cover all the properties, except Separability, but this property can be seen clearly. In particular, if a C-AAS has full traceability and attribute unforgeability then it also has unforgeability, coalition resistance, non-repudiation and traceability, and if the scheme has full-anonymity then it also has anonymity and unlinkability.

For each security notion we propose a game between a hypothetical adversary **A** and challenger **C**. **C** runs certain oracles that **A** can query. We first introduce the following oracles, which will be used in every game.

- **User Secret Key (USK) Query Oracle**: **A** sends **C** an index i , and **C** returns the triple $(bsk[i], A_i, usk[i])$, which are \mathcal{U}_i ’s three parts of secret keys.
- **Signature Oracle**: **A** sends **C** a verification key \overline{D} , a message M and an index i all at **A**’s choice, and **C** returns $\sigma = Sign(M, \overline{D}, gsk)$.
- **Open Oracle**: **A** sends **C** a signature σ , and **C** replies with an index i of the signer. They can execute the *Judge* protocol if it is requested.

- **Corrupt User in Join (CrptJoinUsr) Oracle:** **A** engages with **C** in a join protocol where **A** plays the role of \mathcal{U}_i and **C** is \mathcal{I} .
- **Corrupt Issuer in Join (CrptJoinIss) Oracle:** **A** engages with **C** in a join protocol where **A** plays the role of \mathcal{I} and **C** is \mathcal{U}_i .
- **Attribute Private Key (AttPriKey) Query Oracle:** **A** sends a registration key A_i to **C** together with an index of the attribute j , and **C** returns a private attribute key $T_{i,j}$.
- **Attribute Master Key (AttMasKey) Query Oracle:** **A** sends an index j to **C**, and **C** returns t_j .

In the games of full traceability, full anonymity, and non-frameability, **A** is allowed to choose the master keys of the universal set of attributes $U = \{t_1, \dots, t_m\}$, so all attribute authorities in these games are corrupted. But in the game of attribute unforgeability, **A** is not allowed to do so. We now show these four games as follows.

C-AAS Full Anonymity: We say that a C-AA Scheme is fully anonymous under a specific set of attributes, if no polynomially bounded adversary **A** has a non-negligible advantage against the challenger **C** in the following $A.AAS$ game:

- **A.AAS.Setup:** **C** sets up the system by creating a tracing key tk , an issuer key isk and a general public key gpk . He gives gpk and isk to **A**.
- **A.AAS.Phase 1:** **A** queries the following oracles, USK, Signature oracle, CrptJoinUsr, CrptJoinIss and Open oracle, which are all answered by **C**.
- **A.AAS.Challenge:** **A** provides a message M , two indexes i_0, i_1 , and verification key \bar{D} . **C** responds back with a signature σ_{i_b} , where $b \in_R \{0, 1\}$. Both i_0, i_1 should not have been queried in the CrptJoinUsr oracle, however it can be queried in the Signature oracle.
- **A.AAS.Phase 2:** This stage is similar to Phase 1. The signature σ_b is not queried in the open algorithm.
- **A.AAS.Output:** **A** outputs a guess $\bar{b} \in \{0, 1\}$. If $\bar{b} = b$, he wins the game.

In this game we assume the attribute authority is totally corrupted since the adversary decides the master keys of the attributes (i.e. t_j). We also assume that the issuer is corrupted since the adversary gets the issuer key. The open manager is not corrupted since the tracing key is not disclosed to the adversary. We allow the Signature oracle to be queried for the indexes of the challenge in both phase 1 and 2. This implies that unlinkability is taken into consideration in this game. We assume in the challenge process that CrptJoinUsr is not used to query either of i_0 or i_1 . This is a reasonable assumption. In phase 2, we restrict access to the open oracle such that the challenge signature is not queried in the open oracle. This is another reasonable assumption. Let the advantage $Adv_{A.AAS}(k_1) = Pr[b = \bar{b}] - 1/2$, and then the definition of full anonymity is:

Definition 2. Full Anonymity: A C-AA scheme is fully anonymous if for all polynomial time adversaries, $Adv_{A.AAS}(k_1) < \varepsilon$ and ε is negligible.

C-AAS Full Traceability: We say that our C-AA scheme is traceable if no polynomially bounded adversary **A** has a non-negligible advantage against the challenger **C** in the following *T.AAS* game:

- **T.AAS.Setup:** **C** sets up the system and generates the tracing key tk , issuer key isk and the general public key gpk . He sends **A** tk and gpk . Both **C** and **A** can calculate attribute public keys since they both have access to the master keys of all attributes.
- **T.AAS.Questions:** **C** runs oracles: USK oracle, Signature oracle, and CrptJoinUsr. **A** queries them as described earlier.
- **T.AAS.Output:** **A** sends a message M to **C**, who calculates a new \bar{D} and sends it back. **A** replies with a signature σ . **C** verifies the signature. If it is not valid **C** returns 0. If it is valid, **C** tries tracing it to a signer. If it traces to a signer in which **A** did not query before or if it traces to a nonmember then **A** wins the game. **C** returns 1 if **A** wins else returns 0.

In this game the open manager is totally corrupted by giving the adversary the tracing key. The attribute authorities are also corrupted since everyone knows the master keys of all attributes. The issuing key is kept as a secret from the adversary, otherwise he can easily issue a private key that can not trace to a member of the user group. The adversary does not need to query the open oracle since he has the tracing keys. Refer to the output of this game as *Exp* then the advantage is $Adv_{T.AAS}(k) = Pr[Exp = 1]$ and definition of traceability is as follows:

Definition 3. Full Traceability: A C-AA scheme is fully traceable if for all polynomial time adversaries, $Adv_{T.AAS}(k_1) < \varepsilon$ and ε is negligible.

C-AAS Unforgeability of Attributes: We say that our C-AA scheme is Attribute-Unforgeable if no polynomially bounded adversary **A** has a non-negligible advantage against the challenger **C** in the following U.AAS game:

- **U.AAS.Setup:** **C** sets up the system and plays the role of all attribute authorities in the system. He generates the tracing key tk , the issuer key isk , and the general public key gpk . He creates the universe of attributes by choosing a list of master keys t_1, \dots, t_m . **C** calculates the attribute public keys bpk_1, \dots, bpk_m which he sends together with tk and gpk to **A**. **C** keeps to himself isk and a list of t_j .
- **U.AAS.Phase (1):** **A** queries the oracles USK, Signature, CrptJoinUsr, AttPriKey, and AttMasKey, which are answered by **C**.
- **U.AAS.Challenge:** **A** sends a tree Γ_1 , user l and attribute z which he would like to be challenged on to **C**. **C** replies with \bar{D} for a tree Γ_2 where Γ_2 has two subtrees: the first is Γ_1 and the other is based on t_z . The threshold value of the root in Γ_2 is 2. The challenge condition is that user l has not been queried in AttPriKey for the attribute z . Furthermore the challenged index z should not have been queried in AttMasKey. These two conditions are reasonable as they contradict with the purpose of the game.

- **U.AAS.Phase (2):** This phase is similar to Phase 1 as long as the challenge conditions are not broken.
- **U.AAS.Output:** **A** outputs a signature σ for the user l on the verification key \bar{D} . If signature is valid, **A** wins and **C** outputs 1, else he outputs 0.

We assume the open manager is totally corrupted since the adversary is given the tracing key tk . The adversary can also corrupt attribute authorities of his choice by querying the AttMasKey. The only condition is that the attribute he asks to be challenged upon should not be corrupted as that contradicts with the purpose of the game. The adversary can also corrupt users by querying USK oracle or CrptJoinUsr. Furthermore, the adversary can get attribute private keys for different users and different attributes as long as he does not query the user l for the attribute z . This is a logical assumption because the idea of the game is to create a valid signature that proves that a user has an attribute when he does not really own it. If we refer to the output of this game as Exp then the advantage of the game is notated as $Adv_{AFDAAS}(k_1) = Pr[Exp = 1]$ and definition of attribute unforgeability is as follows.

Definition 4. Unforgeability of Attributes: A C-AA scheme is attribute unforgeable if for all polynomial time adversaries, $Adv_{U.AAS}(k_1) < \varepsilon$ and ε is negligible.

C-AAS Non-frameability: The last security notion we require in C-AAS is non-frameability, which we mean that given the tracing key, the issuing key, the general public key and private keys of a group of all legitimate users, the adversary **A** can not run the *Judge* protocol with the challenger **C** and prove that another user rather than the real signer has signed the given signature. This notions is defined in the following F.AAS game.

- **F.AAS.Setup:** **C** sets up the systems and gives the issuing key *isk*, tracing key tk , and general public key *gpk* to **A**.
- **F.AAS.Questions:** In this stage **A** can query the CrptJoinUsr, CrptJoinIss, Signature and USK oracles.
- **F.AAS.Challenge:** **A** chooses an index i , a signature σ , a verification key \bar{D} , and a message M . **C** verifies the signature. If it is not valid return 0 else trace the signature. If it traces to a user not from the system then return 0. Otherwise **C** accepts the challenge on the registration key A_i and lets **A** know. **A** and **C** then engage in the *Judge* protocol where **A** proves to **C** that user $*$ rather than user i is the signer. The corresponding (A_i, x_i, y_i) and (A^*, x^*, y^*) were not produced in CrpJoinUsr or in a process without the aid of **C**, and there was no USK query for neither of these two users. This implies that **A** does not know the values y_i and y^* and that $y_i \neq y^*$ holds. If (A^*, x^*, y^*) is a valid proof on the challenge, **A** wins the game.

In the non-frameability game, all authorities are corrupted. We do not need an Open oracle since the tracing keys are given to the adversary. The rest of the oracles are used, that allows the adversary to corrupt any legitimate users at his choice apart from the two users in the Challenge phase. If this experiment is referred to

as Exp then the advantage of winning the game is defined as $\text{Adv}_{F,AAS}(n, k) = \Pr[\text{Exp} = 1]$ and the notion of non-frameability is defined as follows.

Definition 5. Non-frameability: A C-AA scheme is non-frameable if for all polynomial time adversaries, $\text{Adv}_{F,AAS}(k_1) < \varepsilon$ and ε is negligible.

3 Preliminaries

This section defines some pairing-based computational problems used in proving our construction traceable. Assume $e : G_1 \times G_2 \rightarrow G_3$ has been created using parameter k^1 . Let G_1, G_2 and G_3 be cyclic groups of prime order p , with a computable isomorphism ψ from G_2 to G_1 or possibly $G_1 = G_2$. Assuming the generators $g_1 \in G_1$, and $g_2 \in G_2$.

Definition 6. (q -Strong Diffie–Hellman Problem (q -SDH) in G_1 and G_2 [4]): Given a $(q+2)$ tuple $(g_1, g_2, g_2^\gamma, g_2^{\gamma^2}, \dots, g_2^{\gamma^q})$ as an input where $\gamma \in \mathbb{Z}_p^*$, output what is called a SDH pair $(g_1^{1/(\gamma+x)}, x)$ for an $x \in \mathbb{Z}_p^*$.

An algorithm A has a negligible advantage in successfully solving q -SDH in (G_1, G_2) if: $\text{Adv}(p) = \left| \Pr[A(g_1, g_2, g_2^\gamma, g_2^{\gamma^2}, \dots, g_2^{\gamma^q}) = (g_1^{1/(\gamma+x)}, x)] - 1/|G| \right| \leq \varepsilon$ where the probability is over a random choice of a generator g_2 (with $g_1 = \psi(g_2)$), and of random bits of A . The advantage $\text{Adv}(p)$ is specific to the prime number p and is equivalent to the probability of the algorithm A calculating the SDH pair from the $(q+2)$ tuple. This problem is believed hard to solve in polynomial time and ε should be negligible [4]. In [9] the author proposes an algorithm that solves the SDH problem in $O(p^{\frac{1}{3}+\varepsilon})$. Throughout this paper we will be using q -SDH where q is roughly an upper bound on the number of users in our proposed C-AA scheme specified in §5. We note that basic Boneh–Boyen signatures are roughly equivalent ($O(p^{2/5+\varepsilon})$) to q -SDH [11]. This proof does not translate directly to our setting, so equivalence is still open.

Definition 7. Bilinear DH Inversion (k -BDHI) assumption [?]: For an integer k , and $x \in_R \mathbb{Z}_p^*$, $g_2 \in G_2$, $g_1 = \psi(g_2)$, $e : G_1 \times G_2 \rightarrow G_3$, given $(g_1, g_2, g_2^x, g_2^{x^2}, \dots, g_2^{x^k})$, computing $e(g_1, g_2)^{1/x}$ is hard.

The following two theorems will be used to prove our proposed C-AA scheme in Section 6.

Theorem 2. (Boneh–Boyen SDH Equivalence [4])

Given a q -SDH instance $(\tilde{g}_1, \tilde{g}_2, \tilde{g}_2^\gamma, \tilde{g}_2^{\gamma^2}, \dots, \tilde{g}_2^{\gamma^q})$, by applying the Boneh and Boyen’s Lemma found in [4] we obtain $g_1 \in G_1, g_2 \in G_2, w = g_2^\gamma$ and $(q-1)$ SDH pairs (A_i, x_i) (such that $e(A_i, wg_2^{x_i}) = e(g_1, g_2)$) for each i . Any SDH pair besides these $(q-1)$ ones can be transformed into a solution to the original q -SDH instance.

¹ $k = k_1$ in §5.

Assume a signature scheme produces the triple $\langle \sigma_0, c, \sigma_1 \rangle$ where σ_0 takes its values randomly from a set, c is the result of hashing the message M with σ_0 , and σ_1 depends only on (σ_0, c, M) . The Forking Lemma is as follows [18]:

Theorem 3. (The Forking Lemma)

Let A be a Probabilistic Polynomial Time Turing machine, given only public data as input. If A can find, with non-negligible probability, a valid signature $(M, \sigma_0, c, \sigma_1)$ then, with non-negligible probability, a replay of this machine, with the same random tape but a different oracle, outputs new valid signatures $(M, \sigma_0, c, \sigma_1)$ and $(M, \sigma_0, \tilde{c}, \tilde{\sigma}_1)$ such that $c \neq \tilde{c}$.

4 Attribute Tree

In our proposed C-AA scheme, an attribute tree is used to present certain attributes requested by a verifier for the purpose of authenticating a user, where the user must demonstrate to hold sufficient attributes in order to pass authentication. This type of trees was first proposed in [11] to implement an attribute based encryption scheme. The tree consists of a set of non-leaf nodes (including a root) that have some child notes, and a set of leaf nodes that do not have any child. Each interior node is a (m, n) threshold gate that represents m out of n children branching from the current node which need to be satisfied for the parent to be considered satisfied. Each leaf node represents an attribute that is associated with a cryptographic key. Satisfaction of a leaf is achieved by proving in possession of some secret information of the key. Only a legitimate user owning the attribute is issued with such a piece of information.

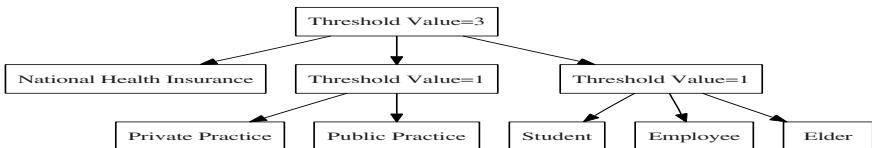


Fig. 2. An Example of the Attribute Tree

For further explanation, we show an example in Figure 2, which demonstrates Scenario 1 specified in §1. Let Γ denote the tree, which is described in a Top-Down-Left-Right manner. So it can be written as $\Gamma = \{(3, 3) : \text{leaf}, (1, 2), (1, 3) : \text{leaf}, \text{leaf}; \text{leaf}, \text{leaf}, \text{leaf}\}$. The first note $(3, 3)$ is a root, the next three notes are the children of the root, and the last five notes are the grandchildren of the root. As mentioned before, (m, n) represents a non-leaf node which has n children and which is a threshold gate with the threshold value m . In the above tree, national health insurance, private practice, public practices, student, employee of health organisations, and elder are all leaves.

Let Υ_i be a set of all attributes owned by a user \mathcal{U}_i and μ be the size of Υ_i . Let κ denote the number of total leaves in the tree Γ . Let \mathfrak{S}_i with the size of τ

be a subset of \mathcal{Y}_i , and \mathfrak{S}_i is used for satisfying Γ . Note that the values of μ , κ and τ do not have to be the same as each other.

When a verifier \mathcal{V} wants the user \mathcal{U}_i to prove possession of a set of attributes, e.g. having national health issuance, holding a doctor's prescription from either of the two practices, and be either a student, an employee of a health organisation or an elder. \mathcal{V} first builds the tree Γ ($\kappa = 6$) and sends it to \mathcal{U}_i . Suppose \mathcal{U}_i is a student and also a part-time employee in a clinic and has national health issuance and a valid prescription from the public practice ($\mu = 4$). \mathcal{U}_i has sufficient attributes to pass every threshold gate of Γ . \mathcal{U}_i decides on a set \mathfrak{S}_i including three of her four attributes, say student, national health issuance and public practice ($\tau = 3$). She makes a proof \mathcal{P} of Γ and sends it back to \mathcal{V} .

Formally we define such an attribute tree **AT** to be a tuple of protocols and algorithms $\mathbf{AT} = (\mathbf{TSetup}, \mathbf{TCreate}, \mathbf{TVerify})$ where

- **TSetup**(k) is a p.p.t. system setup and key generation algorithm. Dependent on applications, this algorithm can be run by a trusted third party \mathcal{T} . It takes as an input a security parameter k , selects a bilinear map $e : G_1 \times G_2 \rightarrow G_3$ defined on the three groups of prime order p , and chooses a generator $w \in G_2$. For each attribute j , \mathcal{T} creates a master private key $t_j \in \mathbb{Z}_p^*$ and computes its public key $bpk_j = w^{t_j}$. To each user \mathcal{U}_i , \mathcal{T} issues a unique base $A_i \in G_1$. If \mathcal{U}_i owns the attribute j , \mathcal{T} issues him a private key $T_{i,j} = A_i^{1/t_j}$.
- **TCreate**($\mathbf{\Gamma}, \alpha, \overline{\mathbf{B}}$) \mathcal{V} takes a set of attribute public keys denoted by $\overline{\mathbf{B}} = \{bpk_1, bpk_2, \dots, bpk_\kappa\}$ as inputs, and creates a tree Γ by performing the following steps:
 - Setup the tree construction, i.e. the parent-child relations of the notes, randomly assign each node (other than the root) an index and add the indexes to Γ . We use the notation $Index(Node)$ to represent the index of a node where $Index(Node) \in \mathbb{Z}_p^*$.
 - Choose a polynomial q_{node} over \mathbb{Z}_p^* for each note, where the polynomial is of degree $d_{node} = k_{node} - 1$, and k_{node} is the threshold value of the node. For example, in the tree represented in Figure 2, $k_{root} = 3$. Pick a secret $\alpha \in_R \mathbb{Z}_p^*$ as $q_{root}(0)$; for the rest of the nodes setup $q_{node}(0) = q_{parent}(Index(Node))$.
 - calculate $D_j = bpk_j^{q_j(0)}$ for all leaves, setup $\overline{D} = \{D_1, \dots, D_\kappa\}$.
- At the end of the protocol, \mathcal{V} sends $\langle \Gamma, \overline{D} \rangle$ to \mathcal{U}_i .
- **TVerify**($\overline{\mathbf{D}}, \mathfrak{S}_i, \overline{\mathbf{T}}$) Given the tree Γ , \mathcal{U}_i first decides \mathfrak{S}_i , a subset of \mathcal{Y}_i , which is sufficient to satisfy Γ , and then computes a proof \mathcal{P} by performing the following steps:
 - Choose $\beta \in_R \mathbb{Z}_p^*$, setup $\bar{T} = \{CT_1, CT_2, \dots, CT_\tau\}$ where $CT_j = T_{i,j}^\beta$.
 - For each leaf note, compute a recursive function $Sign_{Node}$ as follows:

$$Sign_{Node}(leaf) = \begin{cases} \text{If } (j \in \mathfrak{S}_i); \text{return } e(CT_j, D_j) = e(A_i^\beta, w)^{q_j(0)} \\ \text{Otherwise return } \perp \end{cases}$$

- For each non-leaf node ρ , proceed the recursive function $Sign_{Node}$ as follows: For all children z of the node ρ it calls $Sign_{Node}$ and stores output

as F_z . Let \hat{S}_ρ be an arbitrary k_ρ sized set of children nodes z such that $F_z \neq \perp$ and if no such set exists return \perp . Otherwise let

$$\Delta_{\hat{S}_\rho, index(z)} = \prod_{\iota \in \hat{S}_\rho \setminus \{index(z)\}} (-\iota / (index(z) - \iota))$$

and compute

$$\begin{aligned} F_\rho &= \prod_{z \in \hat{S}_\rho} F_z^{q_z(0) \Delta_{\hat{S}_\rho, index(z)}} = \prod_{z \in \hat{S}_\rho} e(T_{i,j}^\beta, D_j)^{q_z(0) \cdot \Delta_{\hat{S}_\rho, index(z)}} = \\ &\prod_{z \in \hat{S}_\rho} e(A_i^\beta, w)^{q_z(0) \cdot \Delta_{\hat{S}_\rho, index(z)}} = \prod_{z \in \hat{S}_\rho} e(A_i, w)^{\beta q_{parent(z)}(index(z)) \cdot \Delta_{\hat{S}_\rho, index(z)}} \end{aligned}$$

As a result, $F_\rho = e(A_i, w)^{q_\rho(0)\beta}$ and $F_{root} = e(A_i^\beta, w)^\alpha$ if tree is satisfied.

- Setup $td = w^\beta$ and $\mathcal{P} = \langle F_{root}, td \rangle$.

\mathcal{U}_i then sends \mathcal{P} to \mathcal{V} , who will accept the proof if $F_{root}^{1/\alpha} = e(A_i, td)$.

Remarks. This implementation of the attribute tree is a modification of the attribute-based encryption scheme in [11]. The major difference is that the tree is not bound with decryption, but with a signature. Each signer \mathcal{U}_i has a unique base A_i . In this implementation, the value A_i is known to \mathcal{V} . But in our C-AA scheme presented in §5, this value is hidden to \mathcal{V} and the signature is anonymous. The value β is introduced here for the purpose of maintaining the randomization in F_{root} . This feature is reserved for the anonymity requirement of the C-AA scheme.

5 The Proposed C-AA Scheme

In this section we present an example of a C-AAS construction. The algorithms defined in §2.1 are specified as follows:

- **KeyGen(k_1)** : Using the security parameter k_1 \mathcal{CA} does the following key and parameter generation: Choose a bilinear map $e : G_1 \times G_2 \rightarrow G_3$ where G_1 , G_2 and G_3 are three groups of sufficiently large prime order p with a computable isomorphism from G_2 to G_1 . Suppose that the DDH problem is hard in G_1 and the q-SDH problem is hard to solve in G_1 and G_2 (See Definition 6). Choose a hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$. Choose two integers ξ_1 and $\xi_2 \in_R \mathbb{Z}_p^*$, two generators g_3 , and $g_4 \in G_1$ and compute $g_1 = g_4^{\xi_1}$ and $g_2 = g_4^{\xi_2} \in G_1$. Choose $h \in_R G_2$ and $\gamma \in \mathbb{Z}_p^*$, and compute $w = h^\gamma \in G_2$. Let the issuing key be $isk = \gamma$, the tracing key be $tk = \{\xi_1, \xi_2\}$, and the general public key be $gpk = \langle e, G_1, G_2, G_3, H, g_1, g_2, g_3, g_4, h, w \rangle$.
- **U.KeyGen(k_2)** : \mathcal{U}_i using security parameter k_2 generates a corresponding public key $upk[i]$ and secret key $usk[i]$ for a digital signature scheme.
- **Join($\mathcal{I}(isk)$)** : $\mathcal{U}_i(upk[i], usk[i])$: The protocol runs as follows in a secure and authentic channel between \mathcal{I} and \mathcal{U}_i :
 1. \mathcal{U}_i picks $y_i \in_R \mathbb{Z}_p^*$, computes and sends $g_1^{y_i}$ to \mathcal{I} .
 2. \mathcal{I} chooses $x_i \in_R \mathbb{Z}_p^*$, computes and sends $A_i = (g_1^{y_i} g_3)^{1/(x_i + \gamma)}$ to \mathcal{U}_i .
 3. \mathcal{U}_i checks $A_i \in G_1$, calculates $S = Sign(A_i, usk[i])$ and sends S along with $upk[i]$ to \mathcal{I} . S is a proof of possession of the private key $usk[i]$.

4. \mathcal{I} checks S with respect to $upk[i]$ and saves $(upk[i], A_i, x_i, S)$ in a database. \mathcal{I} sends x_i to \mathcal{U}_i .
 5. \mathcal{U}_i verifies $A_i^{(x_i+\gamma)} = g_1^{y_i} g_3$ by checking $e(A_i, h^{x_i} w) = e(g_1^{y_i} g_3, h)$. The element A_i will be considered as the registration key. The user's basic secret key is $bsk[i] = \langle A_i, x_i, y_i \rangle$.
- **A.KeyGen_{pub}**(j) : For each attribute j , \mathcal{AA} chooses $t_j \in_R \mathbb{Z}_p^*$ as its master private key and generates an attribute public key $bpk_j = w^{t_j}$.
 - **A.KeyGen_{pri}**(A_i, j) : If \mathcal{U}_i is legitimate to own the attribute j , \mathcal{AA} generates an attribute private key $T_{i,j} = A_i^{1/t_j}$ for the user; and if \mathcal{U}_i owns μ attributions, his entire general secret key is $gsk = \langle bsk[i], T_{i,1}, \dots, T_{i,\mu} \rangle$.
 - **Verifign**($\mathcal{U}_i(gsk, M)$) : The protocol runs as follows in an open channel between \mathcal{U}_i and \mathcal{V} :
 1. Using a set of public attribute keys $\overline{B} = \{bdk_1, \dots, bdk_\kappa\}$, \mathcal{V} chooses $\alpha \in_R \mathbb{Z}_p^*$ and decides on an attribute tree Γ . \mathcal{V} calculates $\overline{D} = \{D_1, \dots, D_\kappa\}$ by following **TCreate**($\Gamma, \alpha, \overline{B}$), and makes \overline{D} available to \mathcal{U}_i .
 2. \mathcal{U}_i computes the signature σ by performing the following steps (this process is call $Sign(M, \overline{D}, gsk)$ in security notions and proofs):
 - Choose $\beta_1, \beta_2 \in_R \mathbb{Z}_p^*$, set $\beta = \beta_1 \beta_2$, and compute $F_{root} = e(A_i, w)^{\alpha\beta}$ by following **TVerify**($\overline{D}, \mathfrak{S}_i, \bar{T}$), where $\bar{T} = \{T_{i,1}^\beta, \dots, T_{i,\kappa}^\beta\}$.
 - Choose $\zeta, \delta, r_\zeta, r_\delta, r_x, r_z \in_R \mathbb{Z}_p^*$, compute $C_1 = g_4^\zeta, C_2 = A_i g_1^\zeta, C_3 = g_4^\delta, C_4 = A_i g_2^\delta, C_5 = e(g_2^\delta A_i^{1-\beta_2}, w)^{\beta_1}$ and $C_6 = w^{\beta_1}$.
 - Compute $R_1 = g_4, R_2 = e(C_2, h)^{r_x} e(g_1, w)^{-r_\zeta} e(g_1, h)^{-r_z}, R_3 = g_4^{r_\delta}$ and $R_4 = g_1^{r_\zeta} g_2^{-r_\delta}$.
 - Compute $c = H(M, C_1, C_2, C_3, C_4, C_5, C_6, R_1, R_2, R_3, R_4)$.
 - Compute $s_\zeta = r_\zeta + c\zeta, s_\delta = r_\delta + c\delta, s_x = r_x + cx_i$ and $s_z = r_z + cz$ where $z = x_i \zeta + y_i$.
 - Send $\sigma = (C_1, C_2, C_3, C_4, C_5, C_6, F_{root}, c, s_\zeta, s_\delta, s_x, s_z)$ to \mathcal{V} .
 3. \mathcal{V} verifies σ as shown below (this process is call $Verify(M, \sigma, \overline{D}, gpk)$ in security notions and proofs):
 - Reject the signature if $F_{root}^{1/\alpha} C_5 = e(C_4, C_6)$ does not hold.
 - Derive $\bar{R}_1 = g_4^{s_\zeta} C_1^{-c}, \bar{R}_2 = e(C_2, h)^{s_x} e(g_1, w)^{-s_\zeta} e(g_1, h)^{-s_z} (\frac{e(C_2, w)}{e(g_3, h)})^c, \bar{R}_3 = g_4^{s_\delta} C_3^{-c}$ and $\bar{R}_4 = g_1^{s_\zeta} g_2^{-s_\delta} / (C_2 C_4^{-1})^c$.
 - Check if $c = H(M, C_1, C_2, C_3, C_4, C_5, C_6, \bar{R}_1, \bar{R}_2, \bar{R}_3, \bar{R}_4)$; if it is not equal then reject the signature.
 - **Open**(σ, α) : \mathcal{O} verifies the signature using α . If the verification passes, he takes the tracing key $tk = \{\xi_1, \xi_2\}$, derive $A_i = (C_2 / (C_1)^{\xi_1}) = (C_4 / (C_3)^{\xi_2})$, and then compares it with his record to find the owner.
 - **Judge**($\mathcal{V}(td, \sigma)$) : To prove that A_i was used in the signature σ a zero knowledge proof is used between \mathcal{O} and \mathcal{V} as follows:
 1. \mathcal{V} picks a random $rnd \in \mathbb{Z}_p^*$ and sends the pair $td = (C_1^{rnd}, C_2^{rnd})$ to \mathcal{O} .
 2. \mathcal{O} calculates $\mathcal{P} = (C_2^{rnd} / C_1^{rnd\xi_1}) = A_i^{rnd}$ and sends it to \mathcal{V} . Note that \mathcal{O} does not know rnd .
 3. \mathcal{V} calculates A_i from \mathcal{P} .

6 Analysis of the C-AA Scheme

This section shows that the the C-AA scheme proposed above is correct and secure according to the definitions in §2. The four security notions will be proved by using the relevant games defined in §2.2. Throughout the proofs, we refer to the adversary as \mathbf{A} , the challenger as \mathbf{C} and a special entity \mathbf{E} plays the role of an adversary when dealing with \mathbf{C} and the role of the challenger when dealing with \mathbf{A} .

6.1 Correctness of the C-AA Scheme

Theorem 4. *The construction in §5 is correct according to Definition 1.*

To prove correctness we first show that $\bar{R}_1 = R_1$, $\bar{R}_2 = R_2$, $\bar{R}_3 = R_3$, and $\bar{R}_4 = R_4$. The equalities hold as shown:

$$\begin{aligned}\bar{R}_1 &= g_4^{s_\zeta} C_1^{-c} = g_4^{r_\zeta + c\zeta} g_4^{-c\zeta} = g_4^{r_\zeta} = R_1, \\ \bar{R}_2 &= e(C_2, h)^{s_x} e(g_1, w)^{-s_\zeta} e(g_1, h)^{-s_z} \left(\frac{e(C_2, w)}{e(g_3, h)} \right)^c \\ &= e(C_2, h)^{r_x + cx_i} e(g_1, w)^{-(r_\zeta + c\zeta)} e(g_1, h)^{-(r_z + cz)} \left(\frac{e(C_2, w)}{e(g_3, h)} \right)^c \\ &= R_2(e(A_i, h)^{cx_i} e(g_1^\zeta, h)^{cx_i} e(g_1, w)^{-c\zeta} e(g_1, h)^{-c(x\zeta + y)} \left(\frac{e(C_2, w)}{e(g_3, h)} \right)^c \\ &= R_2(e(A_i, h)^{cx_i} e(A_i, h)^{cx_i} e(g_1, w)^{-c\zeta} e(g_1, h)^{-cy} \left(\frac{e(C_2, w)}{e(g_3, h)} \right)^c \\ &= R_2 \left(\frac{e(A_i, h)^{x_i} e(C_2, w)}{e(g_1, w)^\zeta e(g_1, h)^y} e(g_3, h) \right)^c = R_2 \left(\frac{e(A_i, h)^{x_i} e(A_i, w)}{e(g_1, h) e(g_3, h)} \right)^c \\ &= R_2 \left(\frac{e(A_i, h)^{x_i + \gamma}}{e(g_1 g_3, h)} \right)^c = R_2, \\ \bar{R}_3 &= g_4^{s_\delta} C_3^{-c} = g_4^{r_\delta + c\delta} g_4^{-c\delta} = g_4^{r_\delta} = R_3, \text{ and} \\ \bar{R}_4 &= g_1^{s_\zeta} g_2^{-s_\delta} / (C_2 C_4^{c-1})^c \\ &= g_1^{(r_\zeta + c\zeta)} g_2^{-(r_\delta + c\delta)} / ((A_i g_1^\zeta)(A_i g_2^\delta)^{-1})^c = g_1^{r_\zeta} g_2^{-r_\delta} = R_4.\end{aligned}$$

The following step is to prove correctness of the attribute verification and the open algorithm. Given that the $F_{root} = e(A_i, w)^{\alpha\beta}$ then

$$F_{root}^{1/\alpha} C_5 = e(A_i, w)^{\beta_1 \beta_2} e(g_2^\delta A_i^{1-\beta_2}, w)^{\beta_1} = e(A_i g_2^\delta, w)^{\beta_1} = e(C_4, C_6).$$

The correctness of the open algorithm follows from $A_i = C_2 / (C_1)^{\xi_1} = C_4 / (C_3)^{\xi_2}$, since

$$\begin{aligned}C_2 / (C_1)^{\xi_1} &= (A_i g_1^\zeta) / g_1^\zeta = A_i, \text{ and} \\ C_4 / (C_3)^{\xi_2} &= (A_i g_2^\delta) / g_2^\delta = A_i.\end{aligned}$$

6.2 Full Anonymity of the C-AA Scheme

Theorem 5. *If the ElGamal Encryption Scheme [10] is IND-CPA secure then the C-AAS construction is fully anonymous under the random oracle model.*

\mathbf{A} is an adversary that attacks the schemes anonymity, \mathbf{E} tries to use \mathbf{A} 's capability in order to break the IND-CPA security of ElGamal encryption scheme. The following is the game:

- **Init:** **A** decides the universal set of attributes $U = \{t_1, \dots, t_m\}$, in which he would like to be challenged upon and gives it to **E**.
- **Setup:** **C** sets up the ElGamal Encryption scheme. The public key is $(g_1, g_4) \in G_1$ where $g_1 = g_4^{\xi_1}$ and $\xi_1 \in \mathbb{Z}_p^*$. ξ_1 is kept secret to the challenger while the rest is public and known to **E**. **E** calculates $g_2 = g_1 g_4^{rnd_1}$ therefore $\xi_2 = \xi_1 + rnd_1 \in \mathbb{Z}_p^*$. **E** chooses a $\gamma \in \mathbb{Z}_p^*$, and $h \in G_2$. **E** can calculate gpk and send it to **A** together with γ .
- **Phase 1:** In this phase **A** queries the oracles: USK, a Signature oracle, CrptJoinUsr, CrptJoinIss, Open and the Hash oracle.

In the Hash oracle **E** responses with a unique but random $c \in \mathbb{Z}_p^*$ every time the query of the tuple $(M, C_1, C_2, C_3, C_4, C_5, C_6, R_1, R_2, R_3, R_4)$ takes place. By unique we mean for the same input response is always the same and by random we mean the response is different and random for other inputs. A list of responses is kept for such purposes.

Replies to the rest of the oracles are straightforward except for the Open oracle. The reason is that **E** has the issuing key γ , the master keys t_1, \dots, t_m and the gpk that are needed in the oracles but she does not have the tracing keys ξ_1 and ξ_2 . Therefore all oracles are run exactly as done in the main scheme except for the open oracle.

To respond to the open oracle **E** will use the list of registration keys she has and rnd_1 . For every element in the list A^* check the following equality and if it holds then $A^* = A_i$

$$\frac{F_{root}^{1/\alpha} C_5}{e(A^*, C_6) e(C_3^{rnd_1}, C_6)} = e\left(\frac{C_4}{C_3^{rnd_1} A^*}, C_6\right)$$

Note that such an equality can not be checked by **A** even if he has the lists of all A^* because the element rnd_1 is used and is only known to the challenger.

- **Challenge:** **A** decides on a message M , two indexes (i_0, i_1) and a verification key \overline{D} in which he would like to be challenged on. **E** sends A_{i_0}, A_{i_1} as two messages to challenge **C** with. **C** encrypts one of them and returns ciphertext $(C_1 = g_4^\zeta, C_2 = A_b g_1^\zeta)$. Note that **E** has to guess b . **E** can simulate a signature by calculating $C_4 = C_2 C_1^{rnd_1} g_2^{rnd_2}$ and $C_3 = C_1 g_4^{rnd_2}$. Given that $\delta = \zeta + rnd_2$, then $C_4 = A_b g_2^\delta$ and $C_3 = g_4^\delta$. **E** chooses randomly $s_\zeta, s_\delta, s_x, s_z$ and c from \mathbb{Z}_p^* . Note that c should have not been a response to a query to the Hash oracle.

She calculates $R_1 = g_4^{s_\zeta} C_1^{-c}$, $R_3 = g_4^{s_\delta} C_3^{-c}$, $R_4 = g_1^{s_\zeta} g_2^{-s_\delta} / (C_2 C_4^{-1})^c$ and $R_2 = e(C_2, h)^{s_x} e(g_1, w)^{-s_\zeta} e(g_1, h)^{-s_z} (\frac{e(C_2, w)}{e(g_3, h)})^c$. Finally **E** creates F_{root} with $T_{i,j} = (C_2 C_1^{rnd_1})^{1/t_j}$ therefore $F_{root} = e(C_2 C_1^{rnd_1}, w^\beta)^\alpha$ for some random $\beta \in \mathbb{Z}_p^*$. $C_6 = w^\beta$ and $C_5 = e(g_2^{rnd_2}, w^\beta)$.

- **Phase 2:** This phase is similar to Phase 1 except that σ_b can not be queried in the open oracle.
- **Output:** **A** outputs a guess $\bar{b} \in \{0, 1\}$.

E can respond to **C** with her guess being \bar{b} .

6.3 Full Traceability of the C-AA Scheme

Theorem 6. *If the q -SDH is hard in group G_1 and G_2 then the C-AAS construction is fully traceable under the random oracle model.*

Similar to the previous two sections and traceability games, the security model will be defined as an interacting framework between **C** and **A** as follows:

- **Init:** **A** decides on the universal set of attributes $U = t_1, \dots, t_m$ from \mathbb{Z}_p^* , where m is the size of the set U . **A** sends this list to **C**.
- **Setup:** **C** is given a bilinear map $e : G_1 \times G_2 \rightarrow G_3$ with generators $g_1 \in G_1$ and $h \in G_2$. He is also given a value $w = h^\gamma$ and n SDH pairs $\langle O_i, x_i \rangle$ for an $1 \leq i \leq n$, which he will be using in creating private key bases. Some of those pairs have $x_i = \star$ which implies that x_i corresponding to O_i is not known; Other pairs are valid SDH pairs (Definition 6). Assume users $i = 1, \dots, n_1$ are the list of honest users that can be corrupted by querying the USK oracle and $i = n_1 + 1, \dots, n_2$ are the ones for dishonest user where the adversary runs the join protocol with **C**. Note that $n = n_1 + n_2$.

C chooses $\xi_1, \xi_2 \in \mathbb{Z}_p^*$ and $g_4, g_2 \in G_1$ such that $g_1 = g_4^{\xi_1}$ and $g_2 = g_4^{\xi_2}$. Finally **C** sets $g_3 = g_1^{rnd_1}$ for some random $rnd_1 \in \mathbb{Z}_p^*$. **C** sends w , and $gpk = \langle G_1, G_2, G_3, e, g_1, g_2, g_3, g_4, h \rangle$. The hash function H is represented as random oracles. Both **C** and **A** can run the $A.KeyGen_{pub}$ to obtain $\langle bpk_1, \dots, bpk_m \rangle$ where $bpk_j = w^{1/t_j}$. **A** is also given ξ_1 , and ξ_2 .

- **Queries:** **A** queries the oracles USK, Signature, CrptJoinUser and Hash as follows:

In the USK Oracle, **A** asks for a certain private key by sending **C** an index $1 \leq i \leq n_1$. If **A** queries an index where $x_i = \star$ abort the game and declare failure; Otherwise he chooses a random $y_i \in \mathbb{Z}_p^*$ and calculates $A_i = O_i^{y_i} O_i^{rnd}$. The private key $\langle A_i, x_i, y_i \rangle$ is sent to the adversary.

The Hash oracle is queried when **A** asks **C** for the hash of $(M, C_1, C_2, C_3, C_4, C_5, C_6, R_1, R_2, R_3, R_4)$, **C** responds with a random element in \mathbb{Z}_p^* and saves the answer just in case the same query is requested again. This represents the hash function H .

The CrptJoinUser presents **A** and **C** engaging in a join protocol where **A** resembles a user i and **C** the issuer manager. If $i = \star$ abort else run the join protocol. The join protocol is similar to the one in the construction with one exception where **C** in the first step sends $O_i^{rnd_2}$. This change helps **C** to generate private keys from the O_i he has rather than γ since he does not know it. He can now obtain $O_i^{y_i}$ and calculate from that the key $\langle A_i, x_i, y_i \rangle$ as in the USK oracle. The rest of the protocol runs normally.

In the Signature oracle, **A** runs the $\overline{D} = TCreate(\Gamma, \alpha, \overline{B})$ for a random $\alpha \in \mathbb{Z}_p^*$ and a set of attribute public keys \overline{B} . He then sends \overline{D} to **C**.

A requests a signature on a message M by the member i . If $x_i \neq \star$ then **C** follows the same signing procedure done in Section 5.

If $x_i = \star$, **C** simulates a signature. He chooses the random elements $s_\zeta, s_\delta, s_x, s_z, \zeta, \delta, \beta_1, \beta_2$ and c , all belong to \mathbb{Z}_p^* . Let $\beta = \beta_1 + \beta_2$. He calculates $C_1 = g_4^\zeta, C_2 = A_i g_1^\zeta, C_3 = g_4^\delta, C_4 = A_i g_2^\delta, C_5 = e(g_2^\delta A_i^{1-\beta_2}, w_1^\beta)$ and finally $C_6 =$

w^{β_1} . **C** also computes $R_1 = g_4^{s_\zeta}$, $R_3 = g_4^{s_\delta} C_3^{-c}$, $R_4 = g_1^{s_\zeta} g_2^{-s_\delta} / (C_2 C_4^{-1})^{-c}$ and finally $R_2 = e(C_2, h)^{s_x} e(g_1, w)^{-s_\delta} e(g_1, h)^{-s_z} (\frac{e(C_2, w)}{e(g_3, h)})^c$. **C** adds c to the list of responses in the hash function.

C can calculate F_{root} as done in the main scheme since he has all master keys needed. Signature is $\sigma = (C_1, C_2, C_3, C_4, C_5, C_6, F_{root}, c, s_\zeta, s_\delta, s_x, s_z)$.

- **Output:** **A** asks to be challenged and sends **C** a message M . **C** responds with a \overline{D} for a certain Γ . If **A** is successful he will output a signature $\sigma = (r, C_1, C_2, C_3, C_4, C_5, C_6, c, s_\xi, s_x, s_\delta, F_{root})$ for a message M . Let A_i^* be the value used in signing the forged signature. For $i = 1, \dots, n$, **C** checks whether $A_i^* = (C_2/(C_1)^{\xi_1}) = (C_4/(C_3)^{\xi_2})$. If the equality holds then this implies that $A_i^* = A_i$. In that case check if $s_{i^*} = \star$ to output σ or otherwise declare failure. If the loop goes through all the (A_i) 's and no equality is identified output σ .

There are two types of forgery. Type-I outputs a signature that can be traced to some identity which is not part of O_{i_0}, \dots, O_{i_n} . Type-II has $A_i^* = O_i$ where $1 \leq i \leq n$ but **A** did not submit a query of i to the USK oracle nor did he participate in the join protocol using it. We prove both forgeries are hard.

Type-I. If we consider Theorem 2 for a $(n+1)$ SDH, we can obtain g_1, g_2 and w . We can also use the n pairs (O_i, x_i) to calculate the private key bases $\langle A_i, x_i, y_i \rangle$. These values are used when interacting with **A**. **A**'s success leads to forgery of Type-I.

Type-II. Using Theorem 2 once again but for a (n) SDH, we can obtain g_1, g_2 and w . Then we use the $n - 1$ pairs (O_i, x_i) to calculate the private key bases $\langle A_i, x_i, y_i \rangle$. In a random index i^* , we choose the missing pair randomly where $O_{i^*} \in G_1$ and set $x_{i^*} = \star$. **A**, in the security model, will fail if he queries the USK oracle with index i^* or use it in the corrupted join protocol. In the Signature oracle (because the hashing oracle is used) it will be hard to distinguish between signatures with a SDH pair and ones without.

A signature will be represented as $\langle M, \sigma_0, c, \sigma_1, \sigma_2 \rangle$, M is the signed message, $\sigma_0 = \langle r, C_1, C_2, C_3, C_4, C_6, R_1, R_2, R_3, R_4 \rangle$, c is the value derived from hashing σ_0 , and $\sigma_1 = \langle s_\zeta, s_\delta, s_x, s_z \rangle$ which are values used to calculate the missing inputs for the hash function. Finally, $\sigma_2 = F_{root}$ the value that depends on the set of attributes in each Signature oracle.

According to the Forking Lemma if we have a replay of this attack with the same random tape but a different response of the random oracle we can obtain a signature $\langle \sigma_0, \tilde{c}, \tilde{\sigma}_1, \sigma_2 \rangle$.

Finally we show how we can extract from $\langle \sigma_0, c, \sigma_1, \sigma_2 \rangle$ and $\langle \sigma_0, \tilde{c}, \tilde{\sigma}_1, \sigma_2 \rangle$ a new SDH tuple. Let $\Delta c = c - \tilde{c}$, and $\Delta s_\zeta = s_\zeta - \tilde{s}_\zeta$, and similarly for $\Delta s_x, \Delta s_\delta, \Delta s_x$ and Δs_z .

Divide two instances of the equations used previously in proving correctness of the scheme. One instance with \tilde{c} and the other with c to obtain the following:

- Dividing $C_1^c/C_1^{\tilde{c}} = g_4^{s_\zeta}/g_4^{\tilde{s}_\zeta}$ we get
 $g_4^{\tilde{\zeta}} = C_1$; where $\tilde{\zeta} = \Delta s_\zeta/\Delta c$
- Dividing $C_2^{s_\delta}/C_2^{\tilde{s}_\delta} = g_4^{s_\delta}/g_4^{\tilde{s}_\delta}$ we get
 $g_4^{\tilde{\delta}} = C_2$; where $\tilde{\delta} = \Delta s_\delta/\Delta c$
- The division of $(C_2 C_4^{-1})^{\Delta c} = g_1^{\Delta s_\zeta} g_4^{\Delta s_\delta}$ implies $(C_2 C_4^{-1}) = g_1^{\tilde{\zeta}} g_4^{\tilde{\delta}}$
- The division of $e(C_2, h)^{\Delta s_x} e(g_1, w)^{-\Delta s_\zeta} e(g_1, h)^{-\Delta s_z} = (\frac{e(g_3, h)}{e(C_2, w)})^{\Delta c}$ leads to
 $e(C_2, h)^{\tilde{x}} e(g_1, w)^{-\tilde{\zeta}} e(g_1, h)^{-\tilde{z}} = (\frac{e(g_3, h)}{e(C_2, w)})^{\Delta c}$ for $\tilde{x} = \Delta s_x/\Delta c$ and $\tilde{z} = \Delta s_z/\Delta c$

If $\tilde{A} = C_2 g_1^{\tilde{\zeta}}$ and $\tilde{y} = \tilde{z} - \tilde{\zeta}\tilde{x}$ then from the last division we get $e(\tilde{A}, h)^{\tilde{x}} e(\tilde{A}, w) = e(g_3, h) e(g_1, h)^{\tilde{y}}$ this implies we have obtained a certificate $(\tilde{A}, \tilde{x}, \tilde{y})$ where $\tilde{A} = (g_3 g_1^{\tilde{y}})^{1/(\tilde{x}+\gamma)}$.

This leads to a SDH pair (\mathcal{A}, χ) and breaking Boneh and Boyen's Lemma (See Theorem 2). Knowing that $\tilde{A} = (g_3 g_1^{\tilde{y}})^{1/(\tilde{x}+\gamma)} = (g_1^{\tilde{y}+rnd_1})^{1/(\tilde{x}+\gamma)}$.

Calculate $(\mathcal{A}, \chi) = (\tilde{A}^{1/(\tilde{y}+rnd_1)}, x_i)$.

6.4 Unforgeability of Attributes in the C-AA Scheme

Theorem 7. *Breaking the Unforgeability of Attributes in the C-AAS construction is as hard as solving the k-BDHI problem under the random oracle model.*

We have a minor issue to prove the above theorem for our C-AAS, so we have to make a condition in the game, where the adversary **A** can only generate the random numbers β_1, β_2 by using a random oracle, which is controlled by the challenger **C**. This obviously makes the model weaker, but we leave a full proof without this condition as an open issue. **C** has an example of the 1-BDHI problem ($p_1 \in G_1, p_2, p_2^x \in G_2$) and his goal is to compute $e(p_1, p_2)^{1/x} \in G_3$. For reasons of limited space, we only give a stretch of the proof for this theorem. The following shows the game model defined in Section 2.

- **U.AAS.Setup:** For setting up the system, **C** takes the three groups G_1, G_2, G_3 from the 1-BDHI problem, sets $g_3 = p_1, g_4 = p_1^r$ for a random r chosen by **C** and $h = p_2$, and generates the remaining domain parameters, the tracing key tk , the issuer key isk , and the general public key gpk by following the C-AA scheme specification properly. **C** plays the role of all attribute authorities in the system. He creates the universal of attributes by choosing a list of master keys t_1, \dots, t_m randomly except one, which is the value x in the 1-BDHI problem and is not known to him. Without loss generality, we call it t_z . **C** calculates the attribute public keys bpk_1, \dots, bpk_m , in which $bpk_z = (p_2^x)^\gamma$ for a random γ chosen by **C**. He sends these attribute public keys together with tk and gpk to **A**. **C** keeps to himself isk and list of t_j .
- **U.AAS.Phase (1):** **A** queries the oracles USK, Signature, CrptJoinUsr, AttPriKey, and AttMasKey which are answered by **C**.
- **U.AAS.Challenge:** **A** sends a tree Γ_1 , user l and attribute z in which he would like to be challenged on. **C** replies with \overline{D} for a tree Γ_2 where Γ_2

has two subtrees: the first is Γ_1 and the other is based on t_z . The threshold value of the root in Γ_2 is 2. The challenge condition is that user l has not been queried in AttPriKey for the attribute z . Furthermore the challenged index z should not have been queried in AttMasKey. These two conditions are reasonable as the contradict with the purpose of the game.

- **U.AAS.Phase (2):** This phase is similar to Phase 1 as long as the challenge conditions are not broken.
- **U.AAS.Output:** **A** outputs a signature σ for the user l on the verification key \overline{D} . If signature is valid then the adversary wins and **C** outputs 1 else **A** loses and **C** outputs 0.

C can reply to most of the oracles without problems since he has most of the private keys needed in creating the outputs such as tk , isk , and the list of t_j except for t_z . So **C** is not able to answer the AttPriKey and AttMasKey queries relevant to t_z .

Let the root polynomial be $q_r(x)$. The subtree Γ_1 has a root with the polynomial $q_1(x)$ and the other child holding attribute t_z has a polynomial $q_2(x)$. Further let $q_r(0) = \alpha$, $q_1(0) = q_r(x_1) = y_1$ and $q_2(0) = q_r(x_2) = y_2$. The root polynomial is of degree 1 since the threshold gate is 2. **C** computes $D_z = p_2^{ty_2}$ for a random t chosen by **C**. Assume at the end of the game **A** creates a valid signature $\sigma = (r, C_1, C_2, C_3, C_4, C_5, C_6, c, s_\xi, s_x, s_\delta, s_z, s_\delta)$ without asking any queries relevant to t_z . The validation of σ implies that the following formula must hold:

$$F_{root} = (e(A_l, w)^{y_1 \cdot x_2 / (x_2 - x_1)} \cdot e(A_l^{1/t_z}, D_z)^{x_1 / (x_1 - x_2)})^\beta,$$

where $A_l = (g_3 \cdot g_1^{y_l})^{1/(\gamma+x_l)}$. As mentioned before, by controlling a random oracle, **C** can retrieve the value β . By using the Forking Lemma as specified in Section 3, **C** can retrieve the value y_l from two rounds of the signature. By following the roles of the authorities in the scheme, **C** knows the values γ , x_l , ξ_1 and ξ_2 . Based on these conditions, **C** can compute $e(p_1, p_2)^{1/x} = e(A_l^{1/t_z}, D_z)^{(\gamma+x_l)/(ty_2(1+r \cdot \xi_1 \cdot y_l))}$ from the above equation. Therefore, the theorem follows.

6.5 Non-frameability of the C-AA Scheme

Finally we discuss Non-frameability. Given all secret keys of authorities **A** cannot show that for a given signature, the signer is someone rather than the real signer.

Theorem 8. *Breaking the Non-frameability of the C-AAS construction is as hard as solving the DDH problem in G_1 .*

Assume that the challenger **C** has an example of the DDH problem in G of order p , where given $g, g^u, g^v, g^c \in G$ and his target is to find whether $g^c = g^{uv}$ or not. In the following game we assume that neither A_i nor A^* have been produced in CrptJoinUsr or in a process without the aid of **C**. This implies **C** knows (A_i, x_i, y_i) and (A^*, x^*, y^*) , **A** does not know the values y_i and y^* , and $y_i \neq y^*$ holds.

- **F.AAS.Setup:** **C** sets the group $G_1 = G$, $g_1 = g$ and $g_3 = g^u$ in the DDH problem, and also sets $g_4 = g^{1/\xi_1}$ for a random ξ_1 chosen by **C**. Then **C** generates the keys isk , tk , and gpk . He gives all of them to **A**. Recall that **A** initiates the game by choosing the universal set of attributes' master keys $U = \{t_1, \dots, t_m\}$. Therefore both **A** and **C** can create attribute keys, create private keys, and trace signatures. **C** maintains a list of all the legitimate user's three-part private keys (A_l, x_l, y_l) .
- **F.AAS.Oracles:** The oracles responses are computed as done in the construction since all keys are known to both **A** and **C**.
- **F.AAS.Output:** **A** chooses an index i , a signature σ , a verification key \overline{D} , and a message M . He sends them to **C**. **C** verifies the signature and if it is not valid he aborts the game returning 0. He then traces it by calculating $A_i = C_2/(C_1)^{\xi_1} = C_4/(C_3)^{\xi_2}$. **C** accepts the challenge on A_i and sends to **A** the values C_2^{rd} and C_1^{rd} . As a result, **A** returns $(A^*)^{rd}$.

Now let us discuss the following two cases (**C** randomly chooses one of these two cases with the probability of 1/2):

Case 1. $A_i = A^*$. **C** chooses a random rd . If he receives $(A^*)^{rd} = (A_l)^{rd}$ for any A_l in the list of (A_l, x_l, y_l) where $l \neq i$, **C** computes the value u as follows. Since $g_1 = g$ and $g_3 = g^u$, $A_l = g^{(u+y_l)/(\gamma+x_l)}$ and $A_i = g^{(u+y_i)/(\gamma+x_i)}$ hold. When $A_l = A_i$, $(u + y_l)/(\gamma + x_l) = (u + y_i)/(\gamma + x_i)$ holds. **C** knows the value γ since it is the issuer's secret key. Therefore **C** can retrieve the value u . In this case, by using **A** as a black box, **C** actually solves the DL problem instead of the DDH problem. Since the DL problem is harder than the DDH problem, so **C** can compute $g^{uv} = (g^v)^u$ and then check whether it is equal to g^c .

Case 2. $A_i \neq A^*$. **C** selects $rd = v$ although he does not know this value. If the signature was created by **C**, he should be able to get the value ζ from his record. If the signature was created by **A**, **C** can retrieve ζ by using the Forking Lemma. He calculates $C_1^{rd} = (g^v)^{\zeta/\xi_1}$ and $C_2^{rd} = (g^v)^{\zeta} \cdot (g^v)^{y_i/(x_i+\gamma)} \cdot (g^c)^{1/(x_i+\gamma)}$.

Note that $C_2^{rd} = (A_i g_1^\zeta)^v$ if $c = uv$ otherwise it will be just random data. *Adversary* returns back $(A^*)^v$ where A^* should belong to the list of possible signers. This can be tested by testing all honestly generated users (i.e. *challenger* will have (A^*, y^*, x^*)). For each element j in the list *challenger* checks whether $(A^*)^v = (g^c \cdot (g^v)^{y_j})^{1/(x_j+\gamma)}$ if it is then check $(A_j, x_j, y_j) \neq (A^*, y^*, x^*)$. If an equality is found the *challenger* outputs 1 for $c = uv$ else he outputs 0.

Note that in both cases, if their exist an adversary that can break the non-frameability then the DDH can be solved in G_1 . Assuming that the adversary can break the non-frameability game and assuming $c = uv$ then the *challenger* should find A^* in the list of possible signers.

7 Honestly Generated Attribute Keys

Previously, in the games of full anonymity, traceability and none-frameability, we assumed the attribute authority is dishonest by giving the adversary the

privilege of creating the master keys in the initialization stage of the games. In the “Unforgeability of Attribute”, the attributes can be corrupted by querying the AttMasKey Oracle. However, in real life, we need the signer to be able to verify that the attribute private keys he obtains are valid. Furthermore, he needs to verify that the attribute public key is also valid. In this section we add two further protocols which will help to achieve that. We start with defining what we mean by “Honestly Generated Attribute Keys”.

Definition 8. (Honestly Generated Attribute Public Keys): *The public attribute key is “generated honestly” if the attribute authority can not produce it or change it without the knowledge of the central authority.*

Definition 9. (Honestly Generated Attribute Private Keys): *The private attribute key is “generated honestly” if the member can verify its correctness with an honestly generated public attribute key and the members’ registration key.*

We add two protocols the APK and the ASK defined as follows:

- **APK($\mathcal{CA} : \mathcal{AA}$)**: This protocol runs between the attribute authority \mathcal{AA} and the central authority \mathcal{CA} . At the end of the protocol the central authority should obtain bpk_j and the attribute authority should obtain t_j .
- **ASK($\mathcal{AA}(t_j) : \mathcal{U}_i(gsk)$)**: This protocol runs between the attribute authority \mathcal{AA} and the user \mathcal{U}_i . The inputs are the master key of the attribute t_j and the users private key gsk . \mathcal{AA} authenticates \mathcal{U}_i and \mathcal{U}_i gets $T_{i,j}$ without revealing the registration key A_i to \mathcal{AA} .

Attribute Public Key Exchange Protocol (APK): This protocol is between the central authority \mathcal{CA} and attribute authority \mathcal{AA} . It authenticates the attribute authority to the central authority. It guarantees that the attribute authorities generate the master key honestly. Therefore we replace the $A.KeyGen_{pub}$ algorithm with a interactive protocol. This protocol is a 6-move key generation protocol adopted from [12] and adjusted as follows:

1. \mathcal{AA} picks a random $a, b \in \mathbb{Z}_p^*$ and $c \in \mathbb{Z}_p^*$. \mathcal{AA} then sends $A = w^a$, $B = w^b$, and $C = w^c$ to \mathcal{CA} .
2. \mathcal{CA} picks $d, e \in \mathbb{Z}_p^*$ and sends $DE = w^dC^e$ to \mathcal{AA} .
3. \mathcal{AA} picks $f \in \mathbb{Z}_p^*$ and sends it to \mathcal{CA} .
4. \mathcal{CA} sends e, d to \mathcal{AA} .
5. \mathcal{AA} checks $DE = w^dC^e$. If the check passes calculate $t_j = a + d + f$. Then send $z = (d + f)a + b \bmod p$ and c to \mathcal{CA} .
6. \mathcal{CA} checks $C = w^c$ and $A^{d+f}B = w^z$ and output $bpk_j = Aw^{d+f}$.

Note that our scheme does not deal with the honesty of the \mathcal{AA} but it guarantees honesty when generating the master key. We will assume some form of standard authentication occurred before the protocol started. Therefore throughout

this protocol our \mathcal{CA} is showing a proof of the master key used by the \mathcal{AA} without revealing the key.

Attribute Private Key Exchange Protocol (ASK): This protocol is executed between the attribute authority \mathcal{AA} and the user \mathcal{U}_i as follows:

1. When \mathcal{AA} wants to verify some information about \mathcal{U}_i before giving him an attribute, \mathcal{U}_i and \mathcal{AA} first run the protocol $Verifign(\mathcal{U}_i(gsk, M), \mathcal{AA}(M, \overline{B}))$.
2. From the $Verifign$ protocol \mathcal{AA} obtains the signature,

$$\sigma = (C_1, C_2, C_3, C_4, C_5, C_6, F_{root}, c, s_\zeta, s_\delta, s_x, s_z).$$
3. \mathcal{AA} verifies the signature and if valid sends $E = C_2^{1/t_j}$ and $F = g_1^{1/t_j}$ back.
4. \mathcal{U}_i calculates his attribute private key as $T_{i,j} = E/F^\delta = A_i^{1/t_j}$.
5. \mathcal{U}_i verifies $T_{i,j}$ by checking $e(T_{i,j}, bpk_j) = e(A_i, w)$.

The protocols APK and ASK are used to prove respectively that the attribute public (or private) key is generated honestly. To generate the attribute public key we used the APK protocol. Security of the APK protocol can be argued under the assumption that the discrete logarithm problem is hard. The reader is referred to [12] for the full proof. In the proof of the private attribute key, the attribute authority can not calculate elements E and F without the knowledge of the master key t_j . The user can verify that the t_j used in creating the $T_{i,j}$ is the same as the one used in the attribute public key. This implies the attribute private key is generated honestly, with the condition that the attribute authority does not have a clue about the value of A_i or $T_{i,j}$.

8 Conclusions

In this paper we have proposed the concept and a concrete scheme of C-AA. The new scheme allows the verifier to choose a set of attributes that he requires the signer to possess. We have defined four security notions: full anonymity, full traceability, attribute unforgeability, and non-frameability.

The scheme includes multiple authority's roles that removes the bottle neck from the central authority, and the scheme is key escrow free. It also implies extra security since in the game models we were able to corrupt one authority while testing the other. One other main contribution is non-frameability where even the open manager can not validly claim that anyone signed a signature unless they actually did.

Future work can include revocation of users, and revocation of attributes. It would also be nice to have the request of the verifier hidden from entities that do not possess enough attributes.

In the concrete scheme proposed, the *number* of messages is constant (and reasonable), and the signature and signing keys are constant in size, while the verification key's length is linear in the number of attributes required by verifier. It would be desirable to reduce the complexity of the computations involved.

References

1. Al-Riyami, S., Paterson, K.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
2. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
3. Bellare, M., Shi, H., Zhang, C.: Foundations of Group Signatures: The Case of Dynamic Groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
4. Boneh, D., Boyen, X.: Short Signatures without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
5. Boyen, X.: Mesh Signatures: How to Leak a Secret with Unwitting and Unwilling Participants. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007)
6. Camenisch, J., Lysyanskaya, A.: Efficient Revocation of Anonymous Group Membership. Cryptology ePrint Archive, Report 2001/113 (2001), <http://eprint.iacr.org/>
7. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
8. Chaum, D.: Security without Identification Transaction Systems to Make Big Brother Obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
9. Cheon, J.H.: Security Analysis of the Strong Diffie-Hellman Problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)
10. 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); IEEE Transactions on Information Theory IT-31, 469–472 (1984)
11. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In: Proceedings of the 13th ACM conference on Computer and communications security, pp. 89–98 (2006)
12. Groth, J.: Fully Anonymous Group Signatures Without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
13. Khader, D.: Attribute Based Authentication Scheme. PhD Thesis, University of Bath (2009)
14. Li, J., Kim, K.: Attribute-Based Ring Signatures. Cryptology ePrint Archive, Report 2008/394 (2008), <http://eprint.iacr.org/>
15. Lysyanskaya, A., Rivest, R., Sahai, A., Wolf, S.: Pseudonym Systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
16. Maji, H., Prabhakaran, M., Rosulek, M.: Attribute-Based Signatures: Achieving Attribute-Privacy and Collusion-Resistance. Cryptology ePrint Archive, Report 2008/328 (2008), <http://eprint.iacr.org/>

17. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based Encryption with Non-Monotonic Access Structures. In: ACM Conference on Computer and Communications Security, pp. 195–203. ACM Press, New York (2007)
18. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptography* 13(3), 361–396 (2000)
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. Shahandashti, S., Naini, R.: Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. *Cryptology ePrint Archive*, Report 2009/126 (2009), <http://eprint.iacr.org/>
21. Verheul, E.: Self-Blindable Credential Certificates from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 533–551. Springer, Heidelberg (2001)
22. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. *Cryptology ePrint Archive*, Report 2008/290 (2008), <http://eprint.iacr.org/>

Comparing with RSA

Julien Cathalo^{1,*}, David Naccache², and Jean-Jacques Quisquater¹

¹ UCL Crypto Group

Place du Levant 3, Louvain-la-Neuve, B-1348, Belgium

julien.cathalo@uclouvain.be, jean-jacques quisquater@uclouvain.be

² École normale supérieure, Département d'informatique

45, rue d'Ulm, F-75230 Paris Cedex 05, France

david.naccache@ens.fr

Abstract. A multi-set (MS) is a set where an element can occur more than once. MS hash functions (MSHFs) map MSS of arbitrary cardinality to fixed-length strings.

This paper introduces a new RSA-based MSHF. The new function is efficient and produces small hashes. We prove that the proposed MSHF is collision-resistant under the assumption of unforgeability of deterministic RSA signatures.

In many practical applications, programmers need to compare two (unordered) sets of integers. A trivial solution consists in sorting both sets ($\mathcal{O}(n \log n)$) and comparing them linearly. We show how MS hash functions can be turned into a quasi-linear-time, quasi-constant-space integer set equality test.

An interesting advantage of the proposed algorithm is its ability to compare MSS without sorting them. This can prove useful when comparing very large files which are read-only or otherwise hard to sort (e.g. on tapes, distributed across web-sites etc).

1 Introduction

A multi-set (MS) is a set where elements can occur more than once. MS hash functions (MSHFs) were introduced by Clarke *et alii* in [5]. While standard hash functions map arbitrary-length strings to fixed-length strings, MSHFs map MSS of arbitrary cardinality to fixed-length strings.

An MSHF \mathcal{H} is *incremental* if $\mathcal{H}(A \cup B)$ can be computed from $\mathcal{H}(A)$ in time proportional to $\#B$.

The **MSet-Mu-Hash** MSHF defined in [5] is MS-collision-resistant, produces small hashes (typically $\cong q$ such that solving discrete logarithms modulo q is hard¹) and is computationally efficient. **MSet-Mu-Hash** is provably secure in the random oracle model under the discrete logarithm assumption.

* Research supported by the Belgian Walloon Region projects MAIS (Programme WIST) and IRMA (Programme Réseaux 2).

¹ e.g. 1024 bits.

In this work we introduce a new MSHF. The proposed function is MS-collision resistant, produces hashes of the size of an RSA modulus and is computationally efficient (comparable to **MSet-Mu-Hash**). However, we prove the new MSHF's security under an assumption different than [5]'s, namely: the unforgeability of deterministic RSA signatures.

Moreover, we show that MSHFs provide a practical solution to the *Set Equality Problem* (SEP). SEP consists in deciding whether two (unordered) sets of n integers are equal. Efficient SEP solutions allow, for instance, to check that two hard drives contain the same files, or that two differently indexed databases contain the same fields. The SEP is related to the *Set Inclusion Problem* (SIP) where one needs to decide whether a set A is a subset of another set B .

In the algebraic computation model where the only allowed operation is comparison, the SEP can be solved in $O(n \log n)$ by sorting both lists; Ben-Or showed that this is optimal [2]. In 2004, Katriel [9] proposed a linear-time set equality test in the algebraic computation model where simple algebraic computations are allowed. Katriel maps the sets into $\mathbb{Z}[X]$ and compares polynomials rather than integers. In essence, [9] shows that SEP is easier when the sets contain integers. However, [9] is impractical as it requires to evaluate the polynomial of a huge value.

Using our MSHF, we propose a *practical* quasi-linear-time, quasi-constant-space integer MS equality test. The algorithm can be used in practice to compare very large MSS and does not yield false negatives. Immunity against false positives is guaranteed if a specific type of RSA signatures is secure.

This non cryptographic application of RSA is quite unusual as, in general, cryptography "borrows" techniques from other fields (such as complexity theory, number theory or statistics) rather than the other way round.

2 The MSet-Mu-Hash Function

Let B be a set. We consider a MS $X = \{x_1, \dots, x_n\} \in B^n$. The **MSet-Mu-Hash** function proposed by Clarke *et alii* [5] is defined as follows: Let q be a large prime and $H : B \rightarrow GF(q)$ be a poly-random function².

$$\text{MSet-Mu-Hash}(X) = \prod_{i=1}^n H(x_i) \bmod q$$

This function is proven to be MS-collision resistant in the random oracle model under the discrete logarithm assumption. It produces small hashes (typically, the size of a prime q such that solving discrete logarithms modulo q is hard) and is computationally efficient.

² H is a poly-random function if no polynomial time (in the logarithm of q) algorithm with oracle access H can distinguish between values of H and true random strings, even when the algorithm is permitted to select the arguments to H . *cf.* to [7].

3 Katriel's Set Equality Test

Let $X = \{x_1, \dots, x_n\} \in \mathbb{Z}^n$ and $Y = \{y_1, \dots, y_n\} \in \mathbb{Z}^n$. Katriel [9] defines the polynomials:

$$p(z) = \prod_{i=1}^n (z - x_i), \quad q(z) = \prod_{i=1}^n (z - y_i) \quad \text{and} \quad d(z) = p(z) - q(z)$$

As $d(z) \equiv 0 \Leftrightarrow X = Y$, the test ascertains that $d \equiv 0$.

A trivial way to do this would be to check that d has $n + 1$ roots. However, this would require $n + 1$ evaluations of d and as an evaluation of d is linear in n , the test will become quadratic in n .

Katriel evaluates $d(z)$ only once, at a point α which is too large to be a root of $d(z)$, unless $d(z) \equiv 0$:

$$\alpha = 1 + 2(mn)^n \quad \text{where } m = \max\{x_1, \dots, x_n, y_1, \dots, y_n\}$$

The test is simply:

$$\text{return}(d(\alpha) \stackrel{?}{=} 0)$$

Complexity: This test requires the computation of:

$$\prod_{i=1}^n (\alpha - x_i) - \prod_{i=1}^n (\alpha - y_i)$$

Which can be done in $2(n - 1)$ multiplications but requires a memory capacity quadratic in n . Indeed:

$$\prod_{i=1}^n (\alpha - x_i) \cong \alpha^n \cong ((mn)^n)^n = (mn)^{n^2}$$

Storing this value requires $n^2 \log_2(mn)$ bits. e.g., to compare lists of 2^{20} two-byte integers ($m = 2^{16}$, $n = 2^{20}$), one needs a 36×2^{40} bit memory, i.e. ~ 4 terabytes. This makes [9] of little practical use.

4 A New RSA-Based MSHF

We now describe a new RSA-based MSHF and a corresponding MS equality test.

4.1 Digital Signatures

The digital signature of a message m is a string that depends on m and a secret known only to the signer. Digital signatures are traditionally (e.g. [8]) defined as follows:

Definition 1 (Signature Scheme). A signature scheme $\{\text{Generate}, \text{Sign}, \text{Verify}\}$ is a collection of three algorithms:

- The key generation algorithm Generate is a probabilistic algorithm that, given 1^k , outputs a pair of matching public and secret keys, $\{\text{pk}, \text{sk}\}$.
- The signing algorithm Sign takes the message m to be signed and the secret key sk and returns a signature $x = \text{Sign}_{\text{sk}}(m)$. Sign may be probabilistic.
- The verification algorithm Verify takes a message m , a candidate signature x' and the public key pk . It returns a bit $\text{Verify}_{\text{pk}}(m, x')$, equal to one if the signature is accepted, and zero otherwise. We require that:

$$\text{Verify}_{\text{pk}}(m, \text{Sign}_{\text{sk}}(m)) = 1$$

The security of signature schemes was formalized in an asymptotic setting by Goldwasser, Micali and Rivest in [8]. Here we use the definitions of [1] that provide a framework for a concrete security analysis of digital signatures and consider resistance against adaptive chosen-message attacks; i.e. a forger \mathcal{F} who dynamically obtains signatures of messages of his choosing and attempts to output a valid forgery.

A valid forgery is a message/signature pair (\tilde{m}, \tilde{x}) such that $\text{Verify}_{\text{pk}}(\tilde{m}, \tilde{x}) = 1$ whilst the signature of \tilde{m} was never requested by \mathcal{F} .

Definition 2. A forger \mathcal{F} is said to $(t, q_{\text{sig}}, \varepsilon)$ -break the signature scheme if after at most $q_{\text{sig}}(k)$ signature queries and $t(k)$ processing time, \mathcal{F} outputs a valid forgery with probability at least $\varepsilon(k)$ for any $k > 0$.

Definition 3. A signature scheme is EUF-CMA $(t, q_{\text{sig}}, \varepsilon)$ -secure if there is no forger capable of $(t, q_{\text{sig}}, \varepsilon)$ -breaking the signature scheme.

Definition 4. A signature scheme is EUF-CMA-secure if for any forger \mathcal{F} that $(t(k), q_{\text{sig}}(k), \varepsilon(k))$ -breaks the scheme, if $t(k)$ and $q_{\text{sig}}(k)$ are polynomial, then $\varepsilon(k)$ is negligible.

4.2 RSA Signatures

RSA [10] is certainly the most famous public-key cryptosystem:

System parameters : Two integers $k, \ell \in \mathbb{N}$ and a function $\mu : \{0, 1\}^\ell \rightarrow \{0, 1\}^k$.

Generate : On input 1^k ,

- Randomly select two distinct $k/2$ -bit primes p and q .
- Compute $N = pq$.
- Pick a random encryption exponent $e \in \mathbb{Z}_{\phi(N)}^*$
- Compute the corresponding decryption exponent $d = e^{-1} \bmod \phi(N)$.

The output of the key generation process is $\{N, e, d\}$; the public key is $\text{pk} = \{N, e\}$ and the private key is $\text{sk} = \{N, d\}$.

Sign : Return $y = \mu(m)^d \bmod N$.

Verify : If $y^e \bmod N = \mu(m)$ then return 1 else return 0.

4.3 Coron-Koeune-Naccache Long-Message RSA Encoding

Signing long messages with RSA is possible using a construction proposed by Coron, Koeune and Naccache (CKN) in [6] (and improved in [4]). CKN split a long message into short blocks, encode each block with μ and multiply all the encoded blocks modulo N . Before encoding a block, CKN's procedure appends to each block a 0 and the block's index i . Then the product of so-formed encodings is appended to 1 and re-encoded again with μ .

System parameters : Two integers $k > 0$ and $a \in [0, k - 1]$ and a function

$$\mu : \{0, 1\}^{k+1} \rightarrow \{0, 1\}^k$$

Generate : As in standard RSA.

Sign : Split the message m into $(k - a)$ -bit blocks such that $m = m[1]||\dots||m[r]$.

Let $\alpha = \prod_{i=1}^r \mu(0||i||m[i]) \bmod N$ where i is an a -bit string representing i .

Let $y = \mu(1||\alpha)$ and return $y^d \bmod N$.

Verify : Let $y = x^e \bmod N$ and recompute $\alpha = \prod_{i=1}^r \mu(0||i||m[i]) \bmod N$.

If $y = \mu(1||\alpha)$ then return 1 else return 0.

4.4 The New MSHF

Let N, e, d be parameters selected as in sub-section 4.3. We additionally require that e is a prime number and $e > n$. Let $\mu : \{0, 1\}^k \rightarrow \{0, 1\}^k$ be an encoding function. We propose the following MSHF:

Apply μ to all the elements of $X = \{x_1, \dots, x_n\}$ and multiply all the encoded integers modulo N :

$$\mathcal{H}(X) = \prod_{i=1}^n \mu(x_i) \bmod N$$

Note that d and e are not used in the function³ and that \mathcal{H} is incremental as it can be updated easily if new elements need to be added to the set.

This construction is very similar to CKN's long-message RSA encoding. The difference is that indices are omitted because the order of the elements is not taken into account. The equality test for two MSS X and Y is simply :

$$\text{return}(\mathcal{H}(X) \stackrel{?}{=} \mathcal{H}(Y))$$

The setup is a slightly modified RSA since we additionally require that e is a prime number and that $e > n$. The latter requirement is not a problem as n is the size of the compared MSS (e.g. $n < 2^{30}$).

³ e is only needed in the security reduction, so strictly speaking the only requirement for the definition of our MSHF is that there is some prime between N and n .

5 MS Collision-Resistance Proof

We now prove the MS collision-resistance of our MSHF. We show that computing a collision in time t with probability ε implies forging μ -encoded RSA signatures in polynomially-related t' and ε' .

Let $\text{Generate}'(k, n)$ be an algorithm that, given two positive integers k and n , returns (N, e, d, μ) such that (N, e, d) are RSA parameters, $|N| = k$ and e is a prime number such that $e > n$. Moreover, $\mu : \{0, 1\}^k \rightarrow \{0, 1\}^k$ is such that the deterministic-padding RSA scheme obtained using μ and (N, e, d) is EUF-CMA secure.

Given k and n and an output of $\text{Generate}'(k, n)$, we consider two different games.

In the first game Game_1 , a forger \mathcal{F} is given $\{N, e, \mu\}$. \mathcal{F} has access to a signing oracle \mathcal{S} that, when given $m_i \in \{0, 1\}^k$, answers $\mu(m_i)^d \bmod N$. After $q_1(k, n)$ signature requests to \mathcal{S} and $t_1(k, n)$ computing time, \mathcal{F} outputs with probability $\varepsilon_1(k, n)$ a forgery (m, s) such that $s = \mu(m)^d \bmod N$ and m was never signed by \mathcal{S} . When $n = n(k)$ is a polynomial in k , the security of the μ -based RSA deterministic-encoding signatures implies that, for any \mathcal{F} , if $t_1(k)$ and $q_1(k)$ are polynomial then $\varepsilon_1(k)$ is negligible.

In the second game Game_2 , an adversary \mathcal{A} is given $\{N, e, \mu, n\}$. This adversary's goal is to produce a collision. It wins if it can find two sets $X = \{x_1, \dots, x_{n'}\}$ and $Y = \{y_1, \dots, y_{n''}\}$ where $x_i, y_i \in [0, 2^k]$ such that $X \neq Y$, $n' \leq n$, $n'' \leq n$ and

$$\prod_{i=1}^{n'} \mu(x_i) \equiv \prod_{i=1}^{n''} \mu(y_i) \pmod{N}$$

\mathcal{A} runs in time $t_2(k, n)$ and succeeds with probability $\varepsilon_2(k, n)$.

Theorem 1. *If there exists an adversary \mathcal{A} that finds a collision in time $t_2(k, n)$ with probability $\varepsilon_2(k, n)$, then there exists a forger \mathcal{F} that finds a forgery after $q_1(k, n) < 2n$ queries to \mathcal{S} and $t_1(k, n) = t_2(k, n) + \mathcal{O}(n^2) + \mathcal{O}(nk^2)$ computing time, with probability $\varepsilon_1(k, n) = \varepsilon_2(k, n)$.*

Proof. Let \mathcal{A} be an adversary that finds a collision in time t_2 with probability ε_2 . We construct a forger \mathcal{F} as follows.

\mathcal{F} first uses \mathcal{A} to try to obtain a collision. If, after t_2 time units, \mathcal{A} does not succeed, \mathcal{F} stops. This happens with probability $1 - \varepsilon_2$.

Otherwise, \mathcal{A} returns a collision. This happens with probability ε_2 and in this case \mathcal{F} learns $X = \{x_1, \dots, x_{n'}\}$ and $Y = \{y_1, \dots, y_{n''}\}$ such that $X \neq Y$ and:

$$\prod_{i=1}^{n'} \mu(x_i) \equiv \prod_{i=1}^{n''} \mu(y_i) \pmod{N}$$

We denote by $\#X(x)$ number of occurrences of an element x in a MS X .

Since $X \neq Y$, there exists x_{i_0} such that $\#X(x_{i_0}) \neq \#Y(x_{i_0})$ and without loss of generality, we assume that $\#X(x_{i_0}) > \#Y(x_{i_0})$.

Let $a = \#X(x_{i_0}) - \#Y(x_{i_0})$ (note that $1 \leq a \leq n'$).

The forger \mathcal{F} finds x_{i_0} and a by sorting and comparing X and Y . We have:

$$\prod_{i \in V} \mu(x_i) \times \mu(x_{i_0})^{\#X(x_{i_0})} \equiv \prod_{i \in W} \mu(y_i) \times \mu(x_{i_0})^{\#Y(x_{i_0})} \pmod{N}$$

where V is the subset of $\{1, \dots, n'\}$ corresponding to the indices of the x_i not equal to x_{i_0} and W is the subset of $\{1, \dots, n''\}$ corresponding to the indices of the y_i not equal to x_{i_0} .

We get:

$$\mu(x_{i_0})^a \equiv \prod_{i \in V} \mu(x_i)^{-1} \times \prod_{i \in W} \mu(y_i) \pmod{N} \quad (1)$$

The integer x_{i_0} does not appear on the right side of this equation.

\mathcal{F} computes u and λ such that $au = \lambda e + 1$ (since e is prime and $0 < a \leq n' \leq n < e$, a is invertible modulo e). \mathcal{F} obtains from the signing oracle \mathcal{S} the signatures $s_i = \mu(x_i)^d \pmod{N}$ of all x_i such that $i \in V$ and the signatures $s'_i = \mu(y_i)^d \pmod{N}$ of all y_i for $i = 1, \dots, n''$. Then \mathcal{F} computes:

$$s = \left(\left(\prod_{i \in V} s_i \right)^{-1} \times \prod_{i \in W} s'_i \right)^u \times \mu(x_{i_0})^{-\lambda} \pmod{N}$$

One can show that $s = \mu(x_{i_0})^d \pmod{N}$ using equation (1):

$$\mu(x_{i_0})^{au} \equiv \prod_{i \in V} \mu(x_i)^{-u} \times \prod_{i \in W} \mu(y_i)^u \pmod{N}$$

$$\mu(x_{i_0})^{\lambda e + 1} \equiv \prod_{i \in V} \mu(x_i)^{-u} \times \prod_{i \in W} \mu(y_i)^u \pmod{N}$$

$$\mu(x_{i_0}) \equiv \prod_{i \in V} \mu(x_i)^{-u} \times \prod_{i \in W} \mu(y_i)^u \times \mu(x_{i_0})^{-\lambda e} \pmod{N}$$

$$\mu(x_{i_0})^d \equiv \prod_{i \in V} \mu(x_i)^{-ud} \times \prod_{i \in W} \mu(y_i)^{ud} \times \mu(x_{i_0})^{-\lambda} \pmod{N}$$

$$\mu(x_{i_0})^d \equiv s \pmod{N}$$

This implies that (x_{i_0}, s) is a valid (message, signature) pair. Since the message x_{i_0} was never sent to \mathcal{S} , \mathcal{F} succeeds in finding a forgery. The number of queries to \mathcal{S} is $\#V + n'' = (n' - a) + n'' < 2n$.

We now evaluate \mathcal{F} 's running time. First, \mathcal{F} runs \mathcal{A} in time $t_2(k, n)$. Finding x_{i_0} and a by sorting and comparing X and Y takes $\mathcal{O}(n \log n)$ time⁴. Computing u and λ takes $\mathcal{O}(n^2)$ time⁵; s can be computed in $2n$ modular multiplications (i.e. $\mathcal{O}(nk^2)$ time), n modular inversions (still $\mathcal{O}(nk^2)$ time), two modular

⁴ Dominated by $\mathcal{O}(n^2)$.

⁵ Extended Euclidean algorithm.

exponentiations ($\mathcal{O}(k^2 \log n)$ dominated by $\mathcal{O}(nk^2)$) and an evaluation of μ , that we omit. All in all, the total running time of \mathcal{F} is:

$$t_1(k, n) = t_2(k, n) + \mathcal{O}(n^2) + \mathcal{O}(nk^2)$$

□

Using Theorem 1 we infer that no algorithm can efficiently find collisions:

Theorem 2. *If $n = n(k)$ is polynomial, the success probability of any adversary that running in polynomial time $t_2(k)$ is negligible.*

Proof. Assume that $n(k)$ is polynomial and that \mathcal{A} finds a collision in polynomial time $t_2(k)$. We want to show that $\varepsilon_2(k)$ is negligible.

By virtue of Theorem 1, there exists an \mathcal{F} that finds a forgery after $q_1(k)$ signature queries in time $t_1(k)$. The number of queries $q_1(k) < 2n(k)$ is polynomial. As we have seen that $t_1(k, n) = t_2(k, n) + \mathcal{O}(n^2) + \mathcal{O}(nk^2)$ is also polynomial. Therefore the attacker's success probability $\varepsilon_1(k)$ is negligible. By virtue of Theorem 1, $\varepsilon_1(k) = \varepsilon_2(k)$ and hence $\varepsilon_2(k)$ is negligible. □

Theorem 2 validates the MSHF's asymptotic behavior.

In practical terms the above means that if a modulus N and a deterministic encoding function μ can be safely used to produce RSA signatures, they can also be used to compute MS hashes. The proposed hash function is comparable in term of efficiency and security to MSet-Mu-Hash while having the merit of relying on a weaker assumption.

6 Caveat Lector: Complexity Estimates

In this section we would like to clarify the expressions "quasi-linear time" and "quasi-linear space" used throughout this paper.

Formally, time complexity is at least $\mathcal{O}(n \log N)$ and space complexity is $\mathcal{O}(\log N)$. We need N to be bigger than the largest integer in the MS and, to make the security reduction work, we require that $N > e > n$. Hence, asymptotically N is not constant, and time complexity is at least $\mathcal{O}(n \log N)$.

However, if N is chosen to provide adequate security guarantees (say 2048 bits) then this will suffice to hash MSS larger than the known universe (e.g. n up to 2^{256}) and a random e will present no problems for the security reduction. In other words, this is a case where we have in any "practical" terms constant space, even though not asymptotically.

For a fixed N , the algorithm then takes time linear in n , as long as the integers are bounded by N , which might be a more serious constraint. Also, with a fixed upper bound on the size of integers, there exist linear-time sorting algorithms (such as radix sort). This suggests again that time complexity is not better than sorting and comparing (though as mentioned above there may well be cases where sorting before comparing is not feasible.)

Finally, the construction depends on a function μ mapping k -bit strings to k -bit strings where $k = \log N$. A careful choice of μ is necessary: N and k are variables hence the complexity of μ must also be factored into the overall complexity of the algorithm.

7 Conclusion and Further Research

In this paper, we proposed a new MSHF whose collision-resistance is directly linked to the security of deterministic-encoding RSA signatures.

The function allows to test if two integer sets are equal using moderate memory and computational resources. The test does not yield false negatives and with carefully chosen parameters, it does not yield false positives either since we prove that a single false positive would imply the insecurity of deterministic RSA signature encoding.

While this gives a practical answer to a theoretical question asked by Katriel, we still do not know how to generalize the proposed construction to solve the SIP i.e. test if a set A is a subset of a set B .

Another open question is whether a similar construction based on an aggregate signature scheme (e.g. [3]) could also be used to provide MSHF functions. The idea would be to multiply hashes⁶ of set elements and prove the resulting MSHF's collision-resistance under the assumption that the aggregate signature scheme is secure.

Acknowledgements

The authors thank Sylvie Baudine, Fabien Laguillaumie, Mark Manulis, David Pointcheval and the anonymous ICALP and IMACC referees for their helpful comments and suggestions.

References

1. Bellare, M., Rogaway, P.: The exact security of digital signatures - How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
2. Ben-Or, M.: Lower bounds for algebraic computation trees. In: STOC 1983 - Proceedings of the fifteenth annual ACM symposium on the Theory of Computing, pp. 80–86. ACM, New York (1983)
3. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. *Journal of Cryptology* 17, 297–319 (2004)
4. Cathalo, J., Coron, J.-S., Naccache, D.: From fixed-length to arbitrary-length RSA encoding schemes revisited. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 234–243. Springer, Heidelberg (2005)
5. Clarke, D., Devadas, S., van Dijk, M., Gassend, B., Suh, G.: Incremental multiset hash functions and their application to memory integrity checking. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 188–207. Springer, Heidelberg (2003)
6. Coron, J.-S., Koeune, F., Naccache, D.: From fixed-length to arbitrary-length RSA padding schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 90–96. Springer, Heidelberg (2000)

⁶ Obtained using the hash function of the aggregate signature scheme.

7. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. *Journal of the ACM* 33(4), 210–217 (1986)
8. Goldwasser, S., Micali, S., Rivest, R.: A Digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of computing* 17, 281–308 (1988)
9. Katriel, I.: On the algebraic complexity of set equality and inclusion. *Information Processing Letters* 92, 175–178 (2004)
10. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21(2), 120–126 (1978)

Double-Exponentiation in Factor-4 Groups and Its Applications

Koray Karabina

Department of Combinatorics & Optimization, University of Waterloo,
Waterloo, Ontario, N2L 3G1, Canada
kkarabin@uwaterloo.ca

Abstract. In previous work we showed how to compress certain prime-order subgroups of the cyclotomic subgroups of orders $2^{2m} + 1$ of the multiplicative groups of $\mathbb{F}_{2^{4m}}^*$ by a factor of 4. We also showed that single-exponentiation can be efficiently performed using compressed representations. In this paper we show that double-exponentiation can be efficiently performed using factor-4 compressed representation of elements. In addition to giving a considerable speed up to the previously known fastest single-exponentiation algorithm for general bases, double-exponentiation can be used to adapt our compression technique to ElGamal type signature schemes.

1 Introduction

The Diffie-Hellman key agreement protocol [3] can be used by two parties A and B to establish a shared secret by communicating over an unsecured channel. Let $G = \langle g \rangle$ be a prime-order subgroup of the multiplicative group \mathbb{F}_q^* of a finite field \mathbb{F}_q . Party A selects a private key a and sends g^a to B . Similarly, B selects a private key b and sends g^b to A . Both parties can then compute the shared secret g^{ab} . Security of the protocol depends on the intractability of the problem of computing a from g^a ; this is called the discrete logarithm problem in G . If q is prime (say $q = p$), then the fastest algorithms known for solving the discrete logarithm problem in G are Pollard's rho method [17] and the number field sieve [9]. To achieve a 128-bit security level against these attacks, one needs to select $\#G \approx 2^{256}$ and $p \approx 2^{3072}$ [6, Section 4.2]. Note that even though the order of G is approximately 2^{256} , the natural representation of elements of G , namely as integers modulo p , are approximately 3072 bits in length. If q is a power of 2 or 3, then one needs to select $\#G \approx 2^{256}$ and $q \approx 2^{4800}$ to achieve 128-bit security level against Pollard's rho method and Coppersmith's index-calculus attack [2,12]. This brings an overhead both to the efficiency of the protocol and to the number of bits that need to be stored or transmitted.

In recent years, there have been several proposals for compressing the elements of certain subgroups of certain finite fields [21,8,1,13,18,5,4,20,11]. The compression methods in these works fall into two categories. They either use a rational parametrization of an algebraic torus [18,5,4], or use the trace representation of elements [21,8,1,13,20,11]. Even though there is a close relation between these

two methods (see [18]), the trace representation of elements seems to give more efficient single-exponentiation algorithms. Single-exponentiation in the context of compressed representations is the operation of computing the compressed representation of g^a given an integer a and the compressed representation of g . Double-exponentiation in the context of compressed representations is the operation of computing the compressed representation of g^{ak+bl} given integers a, b , and the compressed representations of g^k, g^l . It is not clear how to perform double-exponentiation *efficiently* when one uses the trace representation of elements because the trace function is not multiplicative. On the other hand, the use of a rational parametrization of a torus enjoys the full functionality of the group structure, and one can perform double-exponentiation with similar efficiency to that of a single-exponentiation.

Having efficient double-exponentiation is crucial in cryptographic applications. For example, in ElGamal type signature schemes the verifier should perform a double-exponentiation to verify the signature on the received message. Moreover, double-exponentiation can be used to speed up single-exponentiation by representing the exponent $\tau = ak+b$ where k is some fixed integer and a, b are half the bitlength of τ . Then g^k can be precomputed and given τ , one can compute $g^\tau = (g^k)^a \cdot g^b$ using simultaneous exponentiation (*Straus-Shamir's trick*; see Algorithm 14.88 in [14]) much more efficiently than direct exponentiation by τ . As mentioned in the previous paragraph, it is not clear if one can favourably exploit this idea when the trace representation of elements is used.

Lenstra and Stam [22] show that in the case of factor-2 and factor-3 compression in large-prime characteristic fields one can perform double-exponentiation very efficiently and they discuss some related applications such as speeding up the single-exponentiation algorithm for compressed elements.

In this paper, we show that double-exponentiation that works directly with factor-4 compressed elements can be performed very efficiently in the case of characteristic two fields, and describe two particular cryptographic applications. As a first application, we show how to use double-exponentiation to speed up single-exponentiation thereby obtaining an estimated 20% acceleration over the previously known fastest single-exponentiation algorithm when the base is general. Speeding up the single-exponentiation is important as it speeds up some cryptographic protocols using the factor-4 compression technique. For example, as observed in [11], the factor-4 compression technique can be applied to the image of the symmetric bilinear pairing derived from an embedding degree $k = 4$ supersingular elliptic curve defined over a characteristic two field. If this pairing is used to implement the identity-based key agreement protocol of Scott [19], then the messages exchanged can be compressed by a factor of 4; moreover, the single-exponentiation in the protocol can be performed using the compressed representation of elements. As a second application, we give details on deploying factor-4 compressed representation of elements in the Nyberg-Rueppel signature scheme [16]; our method also reduces the size of public keys.

The remainder of the paper is organized as follows. Section 2 introduces some terminology and sets the notation that we will use throughout the paper.

In Section 3 we review the previous work on factor-4 compression. Section 4 presents our double-exponentiation algorithm and an analysis of the algorithm. Two cryptographic applications of our double-exponentiation algorithm are given in Section 5. We make some concluding remarks in Section 6.

2 Preliminaries and Notation

Let q be a prime power and \mathbb{F}_q denote a finite field with q elements. Let n be a prime such that $\gcd(n, q) = 1$, and let k be the smallest positive integer such that $q^k \equiv 1 \pmod{n}$. Then $\mathbb{F}_{q^k}^*$ has a multiplicative subgroup of order n which cannot be embedded in the multiplicative group of any extension field \mathbb{F}_{q^i} for $1 \leq i < k$. For such a triple (q, k, n) we denote the multiplicative group of order n by μ_n and call k the *embedding degree* of μ_n over \mathbb{F}_q . In this setting, we let h be a positive integer and define $t_h = q + 1 - h \cdot n$ to be the *trace* of μ_n over \mathbb{F}_q with respect to the *cofactor* h . Throughout the rest of this paper we will assume that the cofactor h is fixed and we simply denote the trace of μ_n by t , instead of t_h .

Let $g \in \mathbb{F}_{q^k}$ and let s be a positive divisor of k . We assume that g is not contained in any proper subfield of \mathbb{F}_{q^k} . The *conjugates* of g over \mathbb{F}_{q^s} are $g_i = g^{q^{is}}$ for $0 \leq i < k/s$. The *trace* of g over \mathbb{F}_{q^s} is the sum of the conjugates of g over \mathbb{F}_{q^s} , i.e.,

$$\text{Tr}_s(g) = \sum_{i=0}^{\frac{k}{s}-1} g_i \in \mathbb{F}_{q^s}. \quad (1)$$

The *minimal polynomial* of g over \mathbb{F}_{q^s} is the monic polynomial

$$f_{g,s}(x) = \prod_{i=0}^{\frac{k}{s}-1} (x - g_i). \quad (2)$$

Note that $f_{g,s}(x) \in \mathbb{F}_{q^s}[x]$. When $s = 1$ we simply use $\text{Tr}(g)$ and $f_g(x)$ by abuse of notation. Also, we will assume that the conjugates of g over \mathbb{F}_{q^s} are well defined for any integer i by setting $g_i = g_i \bmod k/s$.

We fix some notation for finite field operations that will be used in the remainder of the paper. We will denote by $A_i, a_i, C_i, F_i, I_i, M_i, m_i$, and S_i the operations of addition, addition by 1 or 2, cubing, exponentiation by a power of the characteristic of the field, inversion, multiplication, multiplication by 2, and squaring in \mathbb{F}_{q^i} for $i \in \{1, 2\}$. $SR_{i,j}$ will denote the cost of finding a root of a degree i irreducible polynomial over \mathbb{F}_{q^j} . We use soft- O notation $\tilde{O}(\cdot)$ as follows: $a = \tilde{O}(b)$ if and only if $a = O(b(\log_2 b)^c)$ for some constant c .

3 Review of Factor-4 Compression

We recall some of the facts on factor-4 compression and also review the exponentiation algorithms presented in [11].

Let r be a positive integer, and let $q = 2^{2r+1}$, $t = \pm 2^{r+1}$, $T = |t|$. The values of r for which $q+1-t = hn$ and n is prime lead to a multiplicative subgroup μ_n of $\mathbb{F}_{q^4}^*$ of prime order n with embedding degree 4 and trace t . We fix h, n, q, t, T and $\mu_n = \langle g \rangle$ in this fashion, and also write $c_u = \text{Tr}(g^u)$. Note that $c_0 = 0$ and $c_{uT} = c_u^T$.

The following theorem shows that $g^u \in \mu_n$ can be uniquely represented (up to conjugation over \mathbb{F}_q) by its trace, thus providing compression by a factor 4. From now on, we will refer to this group $\mu_n \subset \mathbb{F}_{q^4}^*$ as the *factor-4 subgroup* of $\mathbb{F}_{q^4}^*$ with trace t .

Theorem 1. [11, Corollary 4.5] *Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t . Let $f_{g^u}(x)$ be the minimal polynomial of $g^u \in \mu_n$ over \mathbb{F}_q . Then*

$$f_{g^u}(x) = x^4 + c_u x^3 + c_u^T x^2 + c_u x + 1.$$

Next, we recall some facts from [11] that we will use in Section 4.

Lemma 1. *Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t . Then for all integers u and v we have*

- (i) $c_{u+v} = (c_u + c_{u-2v})c_v + c_{u-v}c_v^T + c_{u-3v}$ [11, Corollary 4.4(i)].
- (ii) $c_u = c_{-u}$ [11, Lemma 4.2(i)].
- (iii) $c_{2u} = c_u^2$ [11, Corollary 4.4(ii)].
- (iv) $c_u c_v = c_{u+v} + c_{u-v} + c_{u+v(t-1)} + c_{v+u(t-1)}$ [11, Lemma 4.2(ii)].

Remark 1. Throughout the remainder of this paper, we will assume without loss of generality that the trace t is positive. If t is negative then one can replace the expressions of the form $c_u^{p(t)}$, where p is some polynomial, by $c_u^{p(T)}$ without changing the validity of the results in this paper.

Let a, b be integers with $0 < a, b < n$. Single-exponentiation to the base a in μ_n is the operation of computing c_{ab} given c_a and b . Five single-exponentiation algorithms were presented in [11]. We concentrate on Algorithm 1 that works directly with c_a , and is the fastest exponentiation algorithm for general bases. For completeness we present Algorithm 1 and its running time.

Table 1. Cost of Algorithm 1 for factor-4 compression. The exponent is an ℓ -bit integer.

Algorithm	Precomputation	Main Loop
Algorithm 1 [11]	$1I_1 + 1M_1$	$(4M_1 + 4S_1)(\ell - 1)$

Next, we give a generalization of Theorem 4.7 in [11] that will be used in Section 4 to describe a double-exponentiation algorithm in μ_n .

Theorem 2. *Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t . Let $c_u = \text{Tr}(g^u)$,*

$$A = \begin{pmatrix} c_v & c_v & 0 & 0 \\ 0 & c_v & c_v & 0 \\ 0 & 1 & c_v^t & 1 \\ 1 & c_v^t & 1 & 0 \end{pmatrix}, \quad X = \begin{pmatrix} c_{2u-3v} \\ c_{2u-v} \\ c_{2u+v} \\ c_{2u+3v} \end{pmatrix}, \quad Y = \begin{pmatrix} (c_u + c_{u-2v})^2 + c_{u-v}^2 c_v^t \\ (c_{u+v} + c_{u-v})^2 + c_u^2 c_v^t \\ (c_u + c_{u+v})^2 c_v \\ (c_{u-v} + c_u)^2 c_v \end{pmatrix}.$$

Algorithm 1. Single-exponentiationInput: c_a and b Output: c_{ab}

```

1: Write  $b = \sum_{i=0}^{\ell-1} b_i 2^i$  where  $b_i \in \{0, 1\}$  and  $b_{\ell-1} = 1$ 
2:  $s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}] \leftarrow [c_a, 0, c_a, c_a^2]$ 
3:  $m_1 \leftarrow 1/c_a^{t+1}$  and  $m_2 \leftarrow 1/c_a$ 
4: for  $i$  from  $\ell - 2$  down to 0 do
5:    $c_{2u-1} \leftarrow m_1 ((c_{u+1} + c_u + c_{u-1} + c_{u-2})^2 + (c_u + c_{u-1})^2(c_a^t + c_a^2))$ 
6:    $c_{2u} \leftarrow c_u^2$ 
7:    $c_{2u+1} \leftarrow c_{2u-1} + m_2 ((c_{u+1} + c_{u-1})^2 + c_u^2 c_a^t)$ 
8:   if  $b_i = 1$  then
9:      $c_{2u+2} \leftarrow c_{u+1}^2$ 
10:     $s_u \leftarrow [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}]$ 
11:   else
12:      $c_{2u-2} \leftarrow c_{u-1}^2$ 
13:      $s_u \leftarrow [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}]$ 
14:   end if
15: end for
16: Return  $(c_u)$ 

```

Then

- (i) A is invertible and $AX = Y$.
- (ii) If c_v is given then A and A^{-1} can be efficiently computed.
- (iii) $c_{2u-v} = \frac{1}{c_v^{t+1}} ((c_{u+v} + c_u + c_{u-v} + c_{u-2v})^2 + (c_u + c_{u-v})^2(c_v^t + c_v^2))$.

Proof. The proof is analogous to the proof of Theorem 4.7 in [11]. \square

Corollary 1. Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t . Let c_v, c_{2u-v} and $s_{u,v} = [c_{u-2v}, c_{u-v}, c_u, c_{u+v}]$ be given. Then

- (i) $c_{u+2v} = (c_{u+v} + c_{u-v})c_v + c_u c_v^t + c_{u-2v}$, and can be computed at a cost of $1F_1 + 2M_1$.
- (ii) $c_{2u+v} = (c_{u+v} + c_{u-v})c_u + c_v c_u^t + c_{2u-v}$, and can be computed at a cost of $1F_1 + 2M_1$.
- (iii) $c_{u-3v} = (c_u + c_{u-2v})c_v + c_{u-v} c_v^t + c_{u+v}$, and can be computed at a cost of $1F_1 + 2M_1$.
- (iv) $c_{3u-v} = (c_v + c_{2u-v})c_u + c_{u-v} c_u^t + c_{u+v}$, and can be computed at a cost of $1F_1 + 2M_1$.
- (v) $c_{u-4v} = c_{u+2v} + (c_u + c_{u-2v})(c_v^2 + c_v^t + 1) + c_{u-v} c_v^{t+1}$, and (c_{u+2v}, c_{u-4v}) can be computed at a cost of $1F_1 + 5M_1 + 1S_1$.
- (vi) $c_{4u-v} = c_{2u+v} + (c_v + c_{2u-v})(c_u^2 + c_u^t + 1) + c_{u-v} c_u^{t+1}$, and (c_{2u+v}, c_{4u-v}) can be computed at a cost of $1F_1 + 5M_1 + 1S_1$.
- (vii) $c_{3u+v} = c_{3u-v} + (c_{u+v} + c_{u-v})(c_u^2 + c_u^t + 1) + c_v c_u^{t+1}$, and (c_{3u-v}, c_{3u+v}) can be computed at a cost of $1F_1 + 5M_1 + 1S_1$.

Proof. (i)–(iv) follow from Lemma 1(i) and (ii).

(v) Using Lemma 1(i) and (ii), we first write

$$c_{u+2v} = (c_{u+v} + c_{u-v})c_v + c_u c_v^t + c_{u-2v} \quad (3)$$

$$c_{u-4v} = (c_{u-v} + c_{u-3v})c_v + c_{u-2v} c_v^t + c_u. \quad (4)$$

Now, adding (3) and (4) together and replacing $c_{u-3v} = (c_u + c_{u-2v})c_v + c_{u-v}c_v^t + c_{u+v}$ gives $c_{u-4v} = c_{u+2v} + (c_u + c_{u-2v})(c_v^2 + c_v^t + 1) + c_{u-v}c_v^{t+1}$. Finally, once c_{u+2v} is computed as in (i) at a cost of $1F_1 + 2M_1$, it is clear that c_{u-4v} can be computed at a cost of $3M_1 + 1S_1$, which completes the proof.

- (vi) The proof follows as in (v) after interchanging u and v .
- (vii) The proof follows as in (v) after replacing (u, v) by $(u - v, u)$. \square

4 Double-Exponentiation in Factor-4 Groups

Definition 1. Let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t , and as usual let $c_u = \text{Tr}(g^u)$. Let a, b, k, l be integers with $0 < a, b, k, l < n$. Double-exponentiation to the base (k, l) in μ_n is the operation of computing c_{ak+bl} given a, b, c_l and $s_{k,l} = [c_{k-2l}, c_{k-l}, c_k, c_{k+l}]$.

We assume that a, b, k, l are strictly positive as otherwise double-exponentiation just becomes a single-exponentiation. Note that k and l are not necessarily known.

A double-exponentiation algorithm was presented by Stam and Lenstra [22] for second-degree and third-degree recursive sequences. Their algorithm is an adaptation of Montgomery's method [15] to compute second-degree recursive sequences. In this section, we adapt the techniques used in [22,15] and present a double-exponentiation algorithm for fourth-degree recursive sequences.

Algorithm 2 starts with $u = k, v = l, d = a > 0, e = b > 0$, from which it follows that $ud + ve = ak + bl = \tau$, c_v and $s_{u,v}$ are known, and c_{2u-v} can be computed at a cost of $1F_1 + 1I_1 + 3M_1 + 3S_1$ by Theorem 2. The reason we introduce the term c_{2u-v} in step 2 of Algorithm 2 is to avoid the repeated computation of c_{2u-v} during the updates. In the main part of the algorithm, u, v, d, e will be updated so that $ud + ve = ak + bl = \tau$ holds, $d, e > 0$, and $(d + e)$ decreases until $d = e$. Also, c_v, c_{2u-v} and $s_{u,v}$ are updated according to the new values of u and v . When $d = e$, we will have $\tau = d(u + v)$ and $s_{u,v} = [c_{u-2v}, c_{u-v}, c_u, c_{u+v}]$. Finally, we compute $c_\tau = c_{d(u+v)}$ using a single-exponentiation algorithm. Table 2 gives the update rules for $u, v, d, e, c_v, c_{2u-v}$ and $s_{u,v}$. Table 3 gives the cost of each update operation and the factor by which each update reduces $(d+e)$. The cost analysis as summarized in the third column of Table 3 follows from Table 2 and Corollary 1.

Correctness. Let $m = \gcd(a, b)$, $a = ma'$, and $b = mb'$. It is easy to see that if $m = 2^r m'$, $r \geq 0$, and m' is odd then after step 6 in Algorithm 2 we will have $f = 2^r$, $d = m'a'$, $e = m'b'$. Moreover, after step 9, we will have $d = e = m'$ and

Table 2. Update rules for double-exponentiation in factor-4 groups

Rule	Condition	d	e	u	v	c_v	c_{2u-v}	$s_{u,v} = [c_{u-2v}, c_{u-v}, c_u, c_{u+v}]$
if $d \geq e$								
R1	$d \leq 4e$	$d - e$	e	u	$u + v$	c_{u+v}	c_{u-v}	$[c_{u+2v}, c_v, c_u, c_{2u+v}]$
R2	$d \equiv e \pmod{2}$	$(d - e)/2$	e	$2u$	$u + v$	c_{u+v}	c_{3u-v}	$[c_u^2, c_{u-v}, c_u^2, c_{3u+v}]$
R3	$d \equiv 0 \pmod{2}$	$d/2$	e	$2u$	v	c_v	c_{4u-v}	$[c_{u-v}^2, c_{2u-v}, c_u^2, c_{2u+v}]$
R4	$e \equiv 0 \pmod{2}$	d	$e/2$	u	$2v$	c_v^2	c_{u-v}^2	$[c_{u-4v}, c_{u-2v}, c_u, c_{u+2v}]$
else								
S	$e > d$	e	d	v	u	c_u	c_{u-2v}	$[c_{2u-v}, c_{u-v}, c_v, c_{u+v}]$

Table 3. Analysis of update rules for double-exponentiation in factor-4 groups

Rule	Condition	Cost	Reduction factor for $(d + e)$
if $d \geq e$			
R1	$d \leq 4e$	$2F_1 + 4M_1$	$\geq 5/4, < 2$
R2	$d \equiv e \pmod{2}$	$1F_1 + 5M_1 + 2S_1$	2
R3	$d \equiv 0 \pmod{2}$	$1F_1 + 5M_1 + 2S_1$	$\geq 5/3, < 2$
R4	$e \equiv 0 \pmod{2}$	$1F_1 + 5M_1 + 2S_1$	$> 1, < 10/9$
else			
S	$e > d$	0	1

Algorithm 2. Double-exponentiationInput: $a > 0, b > 0, c_l$ and $s_{k,l} = [c_{k-2l}, c_{k-l}, c_k, c_{k+l}]$ Output: c_{ak+bl}

-
- 1: $u \leftarrow k, v \leftarrow l, d \leftarrow a, e \leftarrow b, s_{u,v} \leftarrow s_{k,l}$
 - 2: $c_{2u-v} \leftarrow \frac{1}{c_v^{t+1}} ((c_{u+v} + c_u + c_{u-v} + c_{u-2v})^2 + (c_u + c_{u-v})^2(c_v^t + c_v^2))$
 - 3: $f \leftarrow 1$
 - 4: **while** d and e are both even **do**
 - 5: $d \leftarrow d/2, e \leftarrow e/2, f \leftarrow 2f$
 - 6: **end while**
 - 7: **while** $d \neq e$ **do**
 - 8: Execute the first applicable rule in Table 2
 - 9: **end while**
 - 10: Compute $c_{d(u+v)}$ using Algorithm 1 with input c_{u+v} and d
 - 11: Return $c_{d(u+v)}^f$
-

$d(u+v) = m'a'k + m'b'l$ since $ud + ve$ is kept invariant while applying the rules in Table 2. Hence, $ak + bl = 2^r m'a'k + 2^r m'b'l = f \cdot d(u+v), c_{ak+bl} = c_{f \cdot d(u+v)}$, and the correctness of the algorithm follows from Lemma 1(iii), as $c_{f \cdot d(u+v)} = c_{d(u+v)}^f$.

Analysis. The running time analysis of Algorithm 2 is similar to that in [15]. We will ignore the costs F_1 and S_1 . If R4 is never required during the execution

of Algorithm 2, then it is clear from Table 3 that the cost of the second while loop in Algorithm 2 never exceeds

$$\max(4 \log_{5/4}(a' + b'), 5 \log_2(a' + b'), 5 \log_{5/3}(a' + b')) M_1 \approx 12.4 \log_2(a' + b') M_1.$$

Now, suppose that R4 is used $i > 0$ times successively, with the starting (d, e) value (d_1, e_1) . That is, $d_1 > 4e_1$, $d_1 \equiv 1 \pmod{2}$, and $e_1 \equiv 0 \pmod{2}$. Let (d_2, e_2) be the updated value of (d, e) after i applications of R4. Clearly, $d_1 = d_2$ and $e_1 = e_2 2^i$. Now, only the rule R2 is applicable, and suppose we apply R2 and R3 (possibly after R2) j times until R1 qualifies (i.e. $d \leq 4e$) or $j \leq i$; and suppose that the (d, e) value is updated to (d_3, e_3) . Clearly, $e_2 = e_3$ and $d_3 \leq d_2/2^i$. If $d_3 \leq 4e_3$ then

$$\frac{(d_1 + e_1)}{(d_3 + e_3)} \geq \frac{5e_1}{5e_3} = 2^i.$$

If $j = i$ then

$$\frac{(d_1 + e_1)}{(d_3 + e_3)} \geq \frac{d_2 + e_2 2^i}{d_2/2^j + e_2} = 2^i.$$

In both cases, the value $(d + e)$ value is reduced at least by a factor of 2^i at a cost of $5(i + j)M_1 \leq 10iM_1$. Hence, the total cost of the second while loop in Algorithm 2 never exceeds $12.4 \log_2(a' + b')M_1$. Using Algorithm 1 for step 10 in Algorithm 2, we can conclude that the total cost of Algorithm 2 never exceeds $12.4 \log_2(a + b)M_1$.

In our experiments we observed that the performance of Algorithm 2 in practice is remarkably better than the upper bound $12.4 \log_2(a + b)M_1$. Moreover, the behavior of Algorithm 2 becomes very stable as $(a + b)$ gets larger. We tested the performance of Algorithm 2 with 2^{20} randomly chosen pairs (a, b) such that $a \in [1, 2^{609}], b \in [1, 2^{612}]$ and $a, b \in [1, 2^{1221}]$. The intervals $([1, 2^{609}), [1, 2^{612})$ are relevant for obtaining a faster single-exponentiation algorithm at the 128-bit security level (see Section 5.1), and the interval $[1, 2^{1221})$ is relevant for deploying our double-exponentiation algorithm in the Nyberg-Rueppel signature scheme at the 128-bit security level (see Section 5.2). Our experimental evidence suggests that the main loop in steps 7–9 of Algorithm 2 is executed approximately $1.45 \log_2(A + B)$ times on average (where $A = a/\gcd(a, b)$, $B = b/\gcd(a, b)$), and that the average number of multiplications per iteration is 4.39. Moreover, R1 is used in around 61% of the total number of iterations. In our experiments, as one might expect, $\gcd(a, b)$ is very small and the cost of step 10 is $2.32M_1$ on average (see Table 4). Note that step 11 can be performed at a negligible cost. Hence, we may conjecture that the expected running time of Algorithm 2 is $(1.45 \cdot (0.61 \cdot 4 + 0.39 \cdot 5)) \log_2(a + b)M_1 \approx 6.37 \log_2(a + b)M_1$.

5 Applications of Double-Exponentiation in Factor-4 Groups

In this section we discuss two applications of double-exponentiation of compressed elements in factor-4 groups. We show in Section 5.1 that double-exponentiation

Table 4. Practical behavior of Algorithm 2 at the 128-bit security level. 2^{20} pairs (a, b) were randomly chosen such that $a \in [1, 2^{609}), b \in [1, 2^{612})$ and $a, b \in [1, 2^{1221})$. $A = a/\gcd(a, b)$ and $B = b/\gcd(a, b)$.

	Average		Standard deviation	
	$a \in [1, 2^{609}), b \in [1, 2^{612})$	$(a, b) \in [1, 2^{1221})$	$a \in [1, 2^{609}), b \in [1, 2^{612})$	$(a, b) \in [1, 2^{1221})$
$\log_2(a+b)$	611.4	1221	1.058	0.816
$\log_2(A+B)$	611	1221	1.251	1.053
# of iterations (of steps 7–9)	$1.451 \cdot \log_2(A+B)$	$1.453 \cdot \log_2(a+b)$	0.018	0.012
# of multiplications per iteration (during steps 7–9)	4.390	4.390	0.027	0.019
# of multiplications in step 10	2.320	2.315	5.414	5.412
Rule	Average usage		Standard deviation	
R1	0.610	0.610	0.027	0.019
R2	0.175	0.175	0.013	0.009
R3	0.129	0.129	0.012	0.008
R4	0.085	0.085	0.013	0.009

can be used to obtain faster single-exponentiation of compressed elements in factor-4 groups. Next, we show in Section 5.2 that double-exponentiation allows us to use compression techniques in ElGamal type signature schemes and furthermore obtain shorter public keys. As an illustrative example, we provide details for the Nyberg-Rueppel signature scheme.

Throughout this section we let $\mu_n = \langle g \rangle$ be the factor-4 subgroup of $\mathbb{F}_{q^4}^*$ with trace t , and let $T = |t|$.

5.1 Speeding Up Single-Exponentiation

Algorithm 2 can be turned into a single-exponentiation algorithm as follows. Suppose we want to compute c_{rs} given c_r and $0 \leq s < n$. Clearly, if $s = 0$ then $c_{rs} = 0$. Suppose $s \neq 0$, and write $s = aT + b$ where $0 \leq a \leq T$ and $0 \leq b < T$ (this can be done as $T = \sqrt{2q}$ and $n = q + 1 - t$). If $a = 0$ then $c_{rs} = c_{rb}$ can be computed using Algorithm 1. If $b = 0$ then $c_{rs} = c_{raT} = c_{ra}^T$ (recall that T is a power of 2) can be computed by first computing c_{ra} from c_r and a using Algorithm 1, and then raising the result to the T 'th power. Now, suppose $a, b \neq 0$. Then $c_{rs} = c_{arT+br} = c_{ak+bl}$, where $k = rT$ and $l = r$. In other words, c_{rs} can be computed from $a, b, c_l = c_r$ and $s_{k,l} = [c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ using Algorithm 2. Note that c_l is already given, and $s_{k,l}$ is the last s_u value obtained from running Algorithm 1 with input c_r and T . Our discussion yields Algorithm 3 for single exponentiation in μ_n .

Let $C_1(i)$ denote the cost of Algorithm 1 when the input exponent b is approximately an $\lfloor i \rfloor$ -bit integer. Similarly, let $C_2(i)$ denote the cost of Algorithm 2 when the sum of the two input exponents is approximately an $\lfloor i \rfloor$ -bit integer. If $s \approx n$ and if one uses Algorithm 1 to compute $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ in step 12 of Algorithm 3 then the running time of Algorithm 3 is approximately $C_1((\log_2 n)/2) + C_2((\log_2 n)/2)$ since $T \approx \sqrt{n}$. Therefore, we can conclude from Table 1, and from the running time analysis of Algorithm 2 at the end of Section 4, that the running time of Algorithm 3 will not exceed

Algorithm 3. Single-exponentiationInput: c_r and s Output: c_{rs}

```

1: if  $s = 0$  then
2:    $c_{rs} \leftarrow 0$ 
3: else
4:   Write  $s = aT + b$ , where  $T = |t|$ ,  $0 \leq a \leq T$ ,  $0 \leq b < T$ 
5:   if  $a = 0$  then
6:     Use Algorithm 1 to compute  $c_{rb}$  from  $c_r$  and  $b$ 
7:      $c_{rs} \leftarrow c_{rb}$ 
8:   else if  $b = 0$  then
9:     Use Algorithm 1 to compute  $c_{ra}$  from  $c_r$  and  $a$ 
10:     $c_{rs} \leftarrow c_{ra}^T$ 
11:  else
12:    Compute  $s_u = [c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$  from  $c_r$  and  $T$ 
13:     $s_{k,l} \leftarrow s_u$ ,  $c_l \leftarrow c_r$ 
14:    Use Algorithm 2 to compute  $c_{ak+bl}$  from  $a, b, c_l$  and  $s_{k,l}$ 
15:     $c_{rs} \leftarrow c_{ak+bl}$ 
16:  end if
17: end if
18: Return  $c_{rs}$ 

```

$(4\log_2 n + 12.4\log_2 n)M_1/2 = 8.2(\log_2 n)M_1$ and the running time of Algorithm 3 is expected to be $(4\log_2 n + 6.37\log_2 n)M_1/2 \approx 5.19(\log_2 n)M_1$. Moreover, in the case that the base c_r is fixed, $s_{k,l}$ in Algorithm 3 can be precomputed, and the running time of Algorithm 3 is expected to be $6.38(\log_2 n)M_1/2 = 3.19(\log_2 n)M_1$. Assuming the base c_r is fixed, we may conclude that Algorithm 3 is faster than Algorithm 1. However, for general bases, Algorithm 1 remains the fastest single exponentiation algorithm in factor-4 groups, unless we can find a more efficient method for computing $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ given c_r and T .

We now argue that $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ can be computed at a negligible cost. This will refine Algorithm 3 and give a faster single-exponentiation algorithm than Algorithm 1 for general bases. We first need the following theorem.

Theorem 3. Let $\mu_n = \langle g \rangle \subset \mathbb{F}_{q^4}^*$ be a factor-4 group with trace t , and let $c_r = \text{Tr}(g^r)$. Then

- (i) $c_{rt} = c_r^t$.
- (ii) $c_{r(t-1)} = c_r$.
- (iii) $c_{r(t-2)} = c_r^t$.
- (iv) $c_{r(t+1)}$ is a root of $F(x) = x^2 + c_r^{t+1}x + (c_r^{t+2} + c_r^{3t} + c_r^2 + c_r^4)$.

Proof. (i) follows because t is a power of 2 and $\text{char}(\mathbb{F}_q) = 2$.

(ii) $c_{r(t-1)} = c_{rq} = c_r^q = c_r$ since $q \equiv t - 1 \pmod{n}$.

(iii) $c_{r(t-2)} = c_{rqt} = c_r^t$ since $qt \equiv q^2 + q \equiv q - 1 \equiv t - 2 \pmod{n}$.

(iv) First note that

$$\begin{aligned} c_{r(2t-1)} = & \frac{1}{c_r^{t+1}} ((c_{r(t+1)} + c_{rt} + c_{r(t-1)} + c_{r(t-2)})^2) \\ & + \frac{1}{c_r^{t+1}} ((c_{rt} + c_{r(t-1)})^2(c_r^t + c_{rt}^2)), \end{aligned} \quad (5)$$

by using Theorem 2(iii) with $u = rt$ and $v = r$. Moreover, using Lemma 1(iv) with $u = rt$ and $v = r$ together with the fact that $c_{r(t-1)} = c_r$ from part (ii), we can show that

$$c_{r(2t-1)} = c_r^{t+1} + c_{r(t+1)}. \quad (6)$$

Now, combining (5) and (6) we can write

$$c_{r(t+1)}^2 + c_r^{t+1}c_{r(t+1)} + (c_r^{t+2} + c_r^{3t} + c_r^2 + c_r^4) = 0,$$

which proves the result. \square

Suppose now that c_r is known and that $t > 0$, whence $T = t$ (we can similarly argue when $t < 0$). By Theorem 3(i), (ii), and (iii), we can compute $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}]$ at a cost of $2F_1$. We also know that the roots of $F(x) = x^2 + Bx + C$, $B, C \in \mathbb{F}_q$, $B \neq 0$, are given by $\{r_1, r_2\} = \{B \cdot R(C/B^2), B \cdot (R(C/B^2) + 1)\}$, where $R(C/B^2)$ is a root of $x^2 + x + (C/B^2)$, and if $r_1, r_2 \in \mathbb{F}_q$ then they can be computed at a negligible cost (see [10, Section 3.6.2]). By Theorem 3(iv), the roots of $F(x) = x^2 + c_r^{t+1}x + (c_r^{t+2} + c_r^{3t} + c_r^2 + c_r^4)$ are in \mathbb{F}_q , and so can be computed efficiently. Hence, we can determine $[c_{r(T-2)}, c_{r(T-1)}, c_{rT}, c_{r(T+1)}]$ at a negligible cost, up to the ambiguity that we cannot differentiate between the roots $c_{r(T+1)}$ and $c_{r(T+1)} + c_r^{T+1}$ of $F(x)$ (see Theorem 3). This ambiguity problem can be resolved if the sender of c_r is also required to compute $c_{r(T+1)}$ and send one extra bit $b \in \{0, 1\}$ to help the receiver distinguish $c_{r(T+1)}$ from $c_{r(T+1)} + c_r^{T+1}$ (see Section 5.2 for a similar discussion on how b can be determined).

Hence, assuming that we have a general base c_r and that the distinguisher bit b is known, the running time of Algorithm 3 does not exceed $(12.4 \log_2 n)M_1/2 = 6.2(\log_2 n)M_1$. Based on our experiments (see Table 4), the running time of Algorithm 3 is expected to be $(6.37 \log_2 n)M_1/2 \approx 3.19(\log_2 n)M_1$; this is 20% faster than Algorithm 1 which requires a negligible precomputation and has running time $4(\log_2 n)M_1$ (see Table 1). Note that the Diffie-Hellman key exchange protocol is an example of the scenario where c_r and b can be computed from c_1 by one of the communicating parties and sent to the other party.

To be more concrete, we compare the expected running time of our new single-exponentiation algorithm (Algorithm 3) with the previously known fastest single-exponentiation algorithm (Algorithm 1) at the 128-bit security level when general bases are employed. We let $q = 2^{1223}$ and $t = 2^{612}$. Then $q + 1 - t = 5n$ where n is a 1221-bit prime, and $\mathbb{F}_{q^4}^*$ has a factor-4 subgroup μ_n of order n . For any exponent $s \in [1, n - 1]$, we can write $s = a \cdot 2^{612} + b$ where a has bitlength at

most 609 and b has bitlength at most 612, and compute c_{rs} using Algorithm 3 with input c_r , s and these a, b values. Based on our experiments, the average cost of Algorithm 3 is $3895M_1$ (see Table 4). Note that this is 20% less than the cost of Algorithm 1, which is $4880M_1$.

5.2 Nyberg-Rueppel Signature Scheme in Factor-4 Groups

The Nyberg-Rueppel signature scheme [16] can be modified to be used with compressed representations of elements in μ_n as follows. We suppose that q , n and c_1 are system-wide parameters; alternately, they may be included in a party's public key. Alice chooses a random integer $k \in [1, n - 1]$ and computes $s_{k,1} = [c_{k-2}, c_{k-1}, c_k, c_{k+1}]$. Alice's public key is $s_{k,1}$ and her private key is k . To sign a message M , Alice chooses a random integer $a \in [1, n - 1]$ and computes c_a . Alice extracts a session key $K = \text{Ext}(c_a)$ from c_a , and uses a symmetric-key encryption function E to encipher M under K : $e = E_K(M)$. Moreover, she computes the hash $h = H(e)$ of the encrypted text and sets $s = k \cdot h + a \bmod n$. Alice's signature on M is (e, s) .

If Bob wants to verify Alice's signature (e, s) on M , he first computes $h = H(e)$, replaces h by $-h \bmod n$, and computes c_{hk+s} from $s_{k,1}$ and c_1 . Note that this is a double-exponentiation to the base $(k, 1)$. Bob extracts the session key $K' = \text{Ext}(c_{hk+s})$ from c_{hk+s} and computes $e' = E_{K'}(M)$. Bob accepts the signature if and only if $e' = e$.

One advantage of using the compressed representation of elements in μ_n with the Nyberg-Rueppel signature scheme is that c_1 is a system parameter instead of (the longer) g . We further show that it is possible to have a shorter public key at the expense of some negligible precomputation. In particular, we show that c_{k+1} is a root of a certain quadratic polynomial P_k whose coefficients are determined by c_1 , c_{k-1} and c_k . That is, Alice can omit c_{k+1} from her public key and instead specify one bit to help Bob distinguish c_{k+1} from the other root of P_k .

Consider the matrix

$$M_u = \begin{pmatrix} c_u & c_{u+1} & c_u & c_{u+1} \\ c_{u+1} & c_{u+2} & c_{u+3} & c_{u+4} \\ c_{u+2} & c_{u+3} & c_{u+4} & c_{u+5} \\ c_{u+3} & c_{u+4} & c_{u+5} & c_{u+6} \end{pmatrix}.$$

Giuliani and Gong [7] showed that the characteristic polynomial of the matrix $M_u^{-1} \cdot M_{u+k}$ is equal to the characteristic polynomial of g^k over \mathbb{F}_q , namely $f_{g^k}(x) = x^4 + c_k x^3 + c_k^t x^2 + 1$. In particular, using Lemma 1(i), we can compute the characteristic polynomial P_k of $M_{-2}^{-1} \cdot M_{k-2}$ and find that the coefficient $C_2(P_k)$ of x^2 in $P_k(x)$ satisfies

$$\begin{aligned} (c_1 c_t)^2 C_2(P_k) &= (c_1^2 + c_t^2) c_{k+1}^2 + (c_1 c_t ((c_{k-2} + c_k) + c_1 c_{k-1} + c_k c_t)) c_{k+1} \\ &\quad + (c_1 (c_{k-2} + c_k))^2 + c_1^2 c_t (c_{k-2} c_k + (c_{k-1} + c_k)^2) \\ &\quad + c_1 c_{k-1} c_t (c_1^2 c_k + c_{k-2} + c_k) + (c_{k-1} (c_1 + c_1^2 + c_t))^2 \\ &\quad + c_k^2 (c_1^4 + c_t^3). \end{aligned}$$

We should note that $c_1 \neq 0$ as otherwise f_g would not be irreducible over \mathbb{F}_q . Since $P_k = f_{g^k}$, we must have $C_2(P_k) = c_k^t$ yielding the following result.

Theorem 4. c_{k+1} is a root of the polynomial $P_k(x) = Ax^2 + Bx + C$ where

$$\begin{aligned} A &= c_1^2 + c_t^2 \\ B &= c_1 c_t ((c_{k-2} + c_k) + c_1 c_{k-1} + c_k c_t) \\ C &= (c_1(c_{k-2} + c_k))^2 + c_1^2 c_t (c_{k-2} c_k + (c_{k-1} + c_k)^2 + c_k^t c_t) \\ &\quad + c_1 c_{k-1} c_t (c_1^2 c_k + c_{k-2} + c_k) + (c_{k-1}(c_1 + c_1^2 + c_t))^2 + c_k^2 (c_1^4 + c_t^3). \end{aligned}$$

First note that $A = 0$ if and only if $c_1 = c_t$, that is, if and only if g and g^t are conjugates over \mathbb{F}_q . Using $q^2 \equiv -1 \pmod{n}$ and $q \equiv t-1 \pmod{n}$, we can show that this is never the case.

Now, the roots of $P_k(x) = Ax^2 + Bx + C$, $B, C \in \mathbb{F}_q$, $A, B \neq 0$, are given by $\{r_1, r_2\} = \{(B/A) \cdot R(AC/B^2), (B/A) \cdot (R(AC/B^2) + 1)\}$, where $R(AC/B^2)$ is a root of $x^2 + x + (AC/B^2)$. Furthermore, if $r_1, r_2 \in \mathbb{F}_q$ then they can be computed at a negligible cost (see [10, Section 3.6.2]).

If $A, B \neq 0$, then Alice's public key can be $([c_{k-2}, c_{k-1}, c_k], b)$ where $b \in \{0, 1\}$ is determined by Alice to help Bob to distinguish c_{k+1} from the other root of P_k . For example, Alice can do this as follows. She first computes c_{k+1} (c_{k+1} is obtained for free while computing c_k from c_1 using a single-exponentiation algorithm) and determines the roots r_1, r_2 of P_k together with integers corresponding to the bit representations of r_1 and r_2 , say i_1 and i_2 , respectively. We may assume without loss of generality that $i_1 < i_2$. If $c_{k+1} = r_1$ then Alice sets $b = 0$, otherwise she sets $b = 1$. Given Alice's public key, Bob can easily recover $s_{k,1}$ at a negligible cost and verify Alice's signatures.

If $A \neq 0$ and $B = 0$ in $P_k(x)$, then c_{k+1} can be uniquely computed by taking the square root of C/A in \mathbb{F}_q at a negligible cost.

To be more concrete, we compare the length of public keys when compression is deployed with the length of public keys in the original scheme at the 128-bit security level. As in Section 5.1, we let $q = 2^{1223}$ and $t = 2^{612}$, whence $q+1-t = 5n$ where n is a 1221-bit prime and $\mathbb{F}_{q^4}^*$ has a factor-4 subgroup μ_n of order n . We find that the length of a public key when compression is deployed is 3669 bits, while the public key length without compression is 4892 bits.

6 Concluding Remarks

We showed that double-exponentiation can be efficiently performed using factor-4 compressed representation of elements in factor-4 groups. This allowed us to speed up single-exponentiation and also to use compression techniques in ElGamal type signature schemes.

A future research goal is to further speed up single-exponentiation and double-exponentiation algorithms that use compressed representation of elements. One possibility is that Algorithm 2 can be optimized by taking into account update rules different from those given in Table 2 (see for example [15] and [22]).

Another possibility is to search for better parameters. For example, at the 128-bit security level, generating parameters q, n such that $\mu_n \subset \mathbb{F}_{q^4}^*$ is a factor-4 group, $q = 2^m \approx 2^{1200}$, and n is a 256-bit prime, would result in shorter exponents and therefore a speed up by a factor of more than 4. Such parameters can be found by searching for a suitable prime factor n of $N_1 = \gcd(\Phi_{4m}(2), q + 1 - t)$ or $N_2 = \gcd(\Phi_{4m}(2), q + 1 + t)$, where Φ_{4m} is the $(4m)$ th-cyclotomic polynomial of degree $\varphi(4m)$, and φ is Euler's totient function. Note that when $\varphi(m)$ is significantly smaller than m , then factoring N_i is expected to be easier than factoring $q + 1 \pm t$. For example, when $m = 1209$, N_2 is a 718-bit integer and has a 271-bit prime factor n .

We expect that our double-exponentiation algorithm can be adapted to the factor-6 groups that arise as multiplicative subgroups of characteristic three finite fields [20,11]. However, further analysis would be needed to estimate its efficiency, and to judge how the resulting single-exponentiation method compares to previously known algorithms.

Acknowledgments

The author thanks Alfred Menezes for reading earlier drafts of this paper and for his valuable remarks and suggestions.

References

1. Brouwer, A., Pellikaan, R., Verheul, E.: Doing more with fewer bits. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 321–332. Springer, Heidelberg (1999)
2. Coppersmith, D.: Fast evaluation of logarithms in fields of characteristic two. IEEE Transactions on Information Theory 30, 587–594 (1984)
3. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–665 (1976)
4. Dijk, M., Granger, R., Page, D., Rubin, K., Silverberg, A., Stam, M., Woodruff, D.: Practical cryptography in high dimensional tori. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 234–250. Springer, Heidelberg (2005)
5. Van Dijk, M., Woodruff, D.: Asymptotically optimal communication for torus-based cryptography. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 157–178. Springer, Heidelberg (2004)
6. FIPS 186-3, Digital signature standard (DSS), Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology (2009)
7. Giuliani, K., Gong, G.: New LFSR-based cryptosystems and the trace discrete log problem (Trace-DLP). In: Helleseth, T., Sarwate, D., Song, H.-Y., Yang, K. (eds.) SETA 2004. LNCS, vol. 3486, pp. 298–312. Springer, Heidelberg (2005)
8. Gong, G., Harn, L.: Public-key cryptosystems based on cubic finite field extensions. IEEE Transactions on Information Theory 45, 2601–2605 (1999)
9. Gordon, D.: Discrete logarithms in $GF(p)$ using the number field sieve. SIAM Journal on Discrete Mathematics 6, 124–138 (1993)

10. Hankerson, D., Menezes, A., Vanstone, S.: Guide to elliptic curve cryptography. Springer, New York (2004)
11. Karabina, K.: Factor-4 and 6 compression of cyclotomic subgroups of $\mathbb{F}_{2^{4m}}^*$ and $\mathbb{F}_{3^{6m}}^*$. Journal of Mathematical Cryptology (to appear), <http://eprint.iacr.org/2009/304>
12. Lenstra, A.: Unbelievable security matching AES security using public key systems. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 67–86. Springer, Heidelberg (2001)
13. Lenstra, A., Verheul, E.: The XTR public key system. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 1–19. Springer, Heidelberg (2000)
14. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of applied cryptography. CRC Press, New York (1997)
15. Montgomery, P.: Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains, December 13 (1983); Revised (March 1991) and (January 1992), www.cwi.nl/ftp/pmontgom/Lucas.ps.gz
16. Nyberg, K., Rueppel, A.: Message recovery for signature schemes based on the discrete logarithm problem. Designs, Codes and Cryptography 7, 61–81 (1996)
17. Pollard, J.: Monte Carlo methods for index computation mod p . Mathematics of Computation 32, 918–924 (1978)
18. Rubin, K., Silverberg, A.: Compression in finite fields and torus-based cryptography. SIAM Journal on Computing 37, 1401–1428 (2008)
19. Scott, M.: Authenticated ID-based key exchange and remote log-in with simple token and PIN number, Cryptology ePrint Archive, Report 2002/164 (2002), <http://eprint.iacr.org/2002/164>
20. Shirase, M., Han, D., Hibin, Y., Kim, H., Takagi, T.: A more compact representation of XTR cryptosystem. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E91-A, 2843–2850 (2008)
21. Smith, P., Skinner, C.: A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) ASIACRYPT 1994. LNCS, vol. 917, pp. 357–364. Springer, Heidelberg (1995)
22. Stam, M., Lenstra, A.: Speeding up XTR. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 125–143. Springer, Heidelberg (2001)

Oracle-Assisted Static Diffie-Hellman Is Easier Than Discrete Logarithms*

Antoine Joux¹, Reynald Lercier², David Naccache³, and Emmanuel Thomé⁴

¹ DGA and Université de Versailles, UVSQ PRISM 45 avenue des États-Unis
F-78035 Versailles CEDEX, France
antoine.joux@m4x.org

² DGA/CELAR, La Roche Marguerite, F-35174 Bruz, France and
IRMAR, Université de Rennes 1, Campus de Beaulieu, F-35042 Rennes, France
reynald.lercier@m4x.org

³ École normale supérieure, Équipe de cryptographie, 45 rue d'Ulm
F-75230 Paris CEDEX 05, France
david.naccache@ens.fr

⁴ INRIA Lorraine, LORIA, CACAO – bâtiment A, 615 rue du Jardin botanique
F-54602 Villiers-lès-Nancy CEDEX, France
emmanuel.thome@normalesup.org

Abstract. This paper extends Joux-Naccache-Thomé's e -th root algorithm to the static Diffie-Hellman problem (SDHP).

The new algorithm can be adapted to diverse finite fields by customizing it with an NFS-like core or an FFS-like core.

In both cases, after a number of non-adaptive SDHP oracle queries, the attacker builds-up the ability to solve new SDHP instances *unknown before the query phase*.

While sub-exponential, the algorithm is still significantly faster than all currently known DLP and SDHP resolution methods.

We explore the applicability of the technique to various cryptosystems.

The attacks were implemented in $\mathbb{F}_{2^{1025}}$ and also in \mathbb{F}_p , for a 516-bit p .

Keywords: DLP, SDHP, FFS, NFS.

1 Introduction

In [11], Joux, Naccache and Thomé showed that carefully interacting with an e -th root oracle improves one's ability to compute $\sqrt[e]{t} \bmod n$ for arbitrary t *after* the communication with the oracle.

In Joux *et alii*'s game, the attacker can only query the oracle during a learning phase that takes place *before* t is known. As this learning phase ends, the attacker receives the target t and outputs its e -th root. In other words, [11] establishes that the release of *certain modular roots* leaks information that eases the calculation of all *future* roots.

* Work partially supported by DGA research grant 05.34.058.

This is interesting because, while sub-exponential and oracle-assisted, [11] is still faster than GNFS-factoring.

A recent ePrint publication ([12]) extended [11] to discrete logarithms (DLS). While complexities and details vary, [12]'s overall scenario remains essentially the same, replacing root extractions by DLP calculations.

This work tackles the static Diffie-Hellman problem (SDHP) on which many cryptosystems rely (e.g., [2,4,5] – to mention only a few). Namely, we show how after several carefully crafted non-adaptive SDHP oracle queries, an attacker can improve his ability to solve arbitrary SDHP instances, *unknown before the last oracle query*. The attacker's workload is significantly lower than [12]¹ and all currently known DLP and SDHP resolution algorithms.

The method can be adapted to diverse finite fields by customizing it either with an NFS-like core or an FFS-like core.

We assume that the reader is familiar with [1,11,12], whose notations, introductory descriptions and terminology we extensively borrow and recycle.

Results. Let $Q = q^n$ where q is a prime power² and n an integer.

We present an algorithm \mathcal{A} performing, during a learning phase, L non adaptive SDHP-oracle queries $\{x_1, \dots, x_L\} \in \mathbb{F}_Q^L$ to which the oracle responds with $\{x_1^d, \dots, x_L^d\} \in \mathbb{F}_Q^L$ for some secret $d \in [0, Q - 1]$.

After the learning phase, \mathcal{A} is given a *previously unseen challenge* z and outputs $z^d \in \mathbb{F}_Q$.

The $L_Q(\frac{1}{3}, \sqrt[3]{x})$ -complexities of our algorithms are expressed in the following table:

variant	oracle accesses	learning phase time	post-learning phase time
FFS	$\frac{4}{9}$	-	$\frac{4}{9}$
NFS-HD	$\frac{48}{91}$	$\frac{384}{91}$	$\frac{384}{91}$
NFS	$\frac{4}{9}$	$\frac{32}{9}$	3

Here, NFS-HD stands for high-degree NFS to emphasize the difference between this middle case and the regular NFS.

Let us recall that x values of the FFS, NFS-HD and NFS are respectively $\frac{32}{9}$, $\frac{128}{9}$ and $\frac{64}{9}$.

The following sections will detail \mathcal{A} 's inner mechanics³, complexity and the relation between L and $\{q, n\}$.

The attacks was implemented in $\mathbb{F}_{2^{1025}}$ and also in \mathbb{F}_p , for a 516-bit p (details in the Appendix).

¹ Note, however, that [12] addresses a different problem than ours.

² For simplicity, we assume that q is prime although for some composite degree extensions it is sometime more efficient to choose a composite subfield as a base field.

³ Note that as the learning phase takes place before z is disclosed, $\{x_1, \dots, x_L\}$ and L only depend on Q .

2 Conventions

Complexity. We express complexities using the following notations⁴:

$$L_Q(\nu, \lambda) = \exp\left(\lambda(1+o(1))(\log Q)^\nu (\log \log Q)^{1-\nu}\right),$$

$$\ell_Q(\nu) = (\log Q)^\nu (\log \log Q)^{1-\nu} \quad \text{and} \quad D_Q(\nu, \lambda) = \lfloor \log_q(L_Q(\nu, \lambda)) \rfloor.$$

As $L_Q(\nu, \lambda)$ includes an $o(1)$, $D_Q(\nu, \lambda)$ inherently stands for $D_Q(\nu, \lambda + o(1))$.

Following [9,10], we guarantee that the complexity of our algorithms is $L(\frac{1}{3}, O(1))$ by distinguishing the following three cases:

FFS. $\log q \in o(\ell_Q(\frac{1}{3}))$

- ▶ q is asymptotically smaller than all functions in $L_Q(\frac{1}{3}, \gamma)$.

NFS-HD. $\log q \in [\omega(\ell_Q(\frac{1}{3})), o(\ell_Q(\frac{2}{3}))]$

- ▶ q is asymptotically between all functions in $L_Q(\frac{1}{3}, \gamma)$ and $L_Q(\frac{2}{3}, \gamma)$.

NFS. $\log q \in \omega(\ell_Q(\frac{2}{3}))$

- ▶ q is asymptotically greater than all functions in $L_Q(\frac{2}{3}, \gamma)$.

Assumptions. We rely on the usual heuristic assumptions [3,14,13]:

- The probability that an integer $q \sim L_Q(\nu_q, \lambda_q + o(1))$ is a product of prime factors of magnitude at most $L_Q(\nu_p, \lambda_p + o(1))$ is:

$$L_Q\left(\nu_q - \nu_p, -\frac{\lambda_q}{\lambda_p}(\nu_q - \nu_p) + o(1)\right)$$

despite the fact that integers q might follow non-uniform distributions.

- The probability that a polynomial $q \in \mathbb{F}_q[X]$ of degree $D_Q(\nu_q, \lambda_q) \geq 1$ is a product of irreducible polynomials of degree at most $D_Q(\nu_p, \lambda_p) \geq 1$ is also:

$$L_Q\left(\nu_q - \nu_p, -\frac{\lambda_q}{\lambda_p}(\nu_q - \nu_p) + o(1)\right)$$

3 Function Field Sieve Oracle-Assisted SDH Resolution

3.1 The Algorithm

In this section, we denote by ε the constant $\varepsilon = \sqrt[3]{\frac{4}{9}}$ and by $\vartheta_{n+1} = \frac{1}{3}(1 + \frac{1}{2^n})$.

Polynomial Construction. Let $D = D_Q(\vartheta_{n+1}, \varepsilon)$, D tends to infinity with Q because $\log q \in o(\ell_Q(\frac{1}{3}))$ and thus $D \geq 1$.

Let $d_1 \cong \sqrt{nD}$ and $d_2 \cong \sqrt{\frac{n}{D}}$ with $d_1 d_2 \geq n$.

Choose two univariate polynomials $f_1, f_2 \in \mathbb{F}_q[x]$ of respective degrees d_1, d_2 , such that the resultant of $F_1(x, y) = y - f_1(x)$ and $F_2(x, y) = x - f_2(y)$ with respect to variable y has an irreducible factor of degree n over \mathbb{F}_q .

⁴ Throughout this paper log denotes the natural logarithm.

Let $f(x)$ denote this irreducible factor of $\text{Res}_y(F_1, F_2) = x - f_2(f_1(x))$, then clearly $\mathbb{F}_{q^n} \simeq \mathbb{F}_q[x]/(f(x))$. Let α be a root of $f(x)$ in \mathbb{F}_{q^n} and let $\beta = f_1(\alpha)$. Then, by construction, we have $\alpha = f_2(\beta)$ too.

Learning Phase. We let \mathcal{F}_D be the set of all monic irreducible polynomials over \mathbb{F}_q of degree up to D and define $\{x_i\} = \{\{\mathfrak{p}(\alpha), \mathfrak{p}(\beta)\} \mid \mathfrak{p}(x) \in \mathcal{F}_D\}$. After about $2q^D \simeq L_Q(\frac{1}{3}, \varepsilon + o(1))$ SDHP-oracle queries, we get $\{\mathfrak{p}(\alpha)^d, \mathfrak{p}(\beta)^d\}_{\mathfrak{p} \in \mathcal{F}_D}$.

Descent. The descent phase consists in finding a multiplicative relation between the target z and the elements of $\mathcal{F}_\alpha \cup \mathcal{F}_\beta$. To do so, we use a special-q sieving subroutine (hereafter **SqSS**) as a basic building block. **SqSS** is called iteratively to steadily decrease the degree of intermediate polynomials.

The **SqSS** works as follows:

Let \mathfrak{q} be an irreducible polynomial in x (respectively y) of degree $d_{\mathfrak{q}}$.

To ease **SqSS**'s analysis we write $d_{\mathfrak{q}}$ as:

$$d_{\mathfrak{q}} = D_Q(\nu_{\mathfrak{q}}, \lambda_{\mathfrak{q}}) = (\lambda_{\mathfrak{q}} + o(1)) n^{\nu_{\mathfrak{q}}} \log_q^{1-\nu_{\mathfrak{q}}} n$$

for two constants $\nu_{\mathfrak{q}}, \lambda_{\mathfrak{q}}$ such that $\sqrt[3]{n} \leq d_{\mathfrak{q}} \leq n$. Note that the values $\nu_{\mathfrak{q}}, \lambda_{\mathfrak{q}}$ evolve simultaneously – as will be seen later in further detail.

SqSS's purpose is to write an element $\mathfrak{q}(\alpha)$ or $\mathfrak{q}(\beta)$ as a product of elements of the form $\mathfrak{p}(\alpha)$ and $\mathfrak{p}(\beta)$, where the \mathfrak{p} -polynomials are of degrees smaller than the degree of \mathfrak{q} .

Let $d_\varepsilon = D_Q(\frac{1}{3}, \varepsilon)$, we consider $q^{d_\varepsilon} \simeq L_Q(\frac{1}{3}, \varepsilon)$ polynomials:

$$H(x, y) = \sum_{\substack{0 \leq i \leq d_x \\ 0 \leq j \leq d_y}} h_{i,j} x^i y^j \in \mathbb{F}_q[x, y]$$

$$\text{where } d_y = \left\lceil \max(\sqrt{(d_{\mathfrak{q}} + d_\varepsilon)/D} - 1, 1) \right\rceil \text{ and } d_x = \left\lceil \frac{d_{\mathfrak{q}} + d_\varepsilon + 1}{d_y + 1} \right\rceil - 1$$

We only consider polynomials $H(x, y)$ such that \mathfrak{q} divides $H_x = \sum_{i,j} h_{i,j} x^i f_1(x)^j$ (respectively $H_y = \sum_{i,j} h_{i,j} f_2(y)^i y^j$).

The complexity analysis (*cf. infra*) shows that there exists, amongst these polynomials, polynomials H such that H_x and H_y both admit a factorization of the form:

$$\prod_{\substack{\mathfrak{p} \in \mathcal{F}_{D_Q}(\nu_{\mathfrak{p}}, \lambda_{\mathfrak{p}}) \\ \text{for } \mathfrak{p} \neq \mathfrak{q}}} \mathfrak{p} \quad \text{with} \quad \begin{cases} \nu_{\mathfrak{p}} = \frac{1}{6} + \frac{\nu_{\mathfrak{q}}}{2} & \text{and } \lambda_{\mathfrak{p}} = \frac{2\sqrt{\lambda_{\mathfrak{q}} + \varepsilon}}{3\varepsilon} \text{ if } \nu_{\mathfrak{q}} > \vartheta_{n+1} \\ \nu_{\mathfrak{p}} = \nu_{\mathfrak{q}} & \text{and } \lambda_{\mathfrak{p}} = \frac{3\varepsilon + \lambda_{\mathfrak{q}}}{6\varepsilon\sqrt{\varepsilon}} \text{ otherwise.} \end{cases}$$

By proceeding so, we can express an element $\mathfrak{q}(\alpha)$ or $\mathfrak{q}(\beta)$ as a product of polynomials \mathfrak{p} of degrees at most $D_Q(\nu_{\mathfrak{p}}, \lambda_{\mathfrak{p}})$ evaluated at α or β .

Descending Using SqSS. Define the sequences:

$$\nu_{k+1} = \frac{1}{6} + \frac{\nu_k}{2} \quad \text{and} \quad \lambda_{k+1} = \frac{2\sqrt{\lambda_k + \varepsilon}}{3\varepsilon} \quad \text{with} \quad \nu_0 = \lambda_0 = 1$$

We note that, indeed, $\nu_k = \vartheta_k$ and that the limit at infinity of the sequence λ_k is equal to $\frac{1+\sqrt{5}}{3\sqrt{18}} \cong 1.234$. As our procedures perform a finite number of steps, these *ad infinitum* convergence targets are only approached. However, the limit at infinity guarantees that during the phase of the descent where ν decreases, λ remains bounded (smaller than the above limit). After that point ν remains constant and λ quickly tends to ε .

The descent starts with one polynomial $q(x)$ of maximal degree $D_Q(\nu_0, \lambda_0)$ with $z = q(\alpha)$. After a first invocation of SqSS we get polynomials p in x and y of degree at most $D_Q(\nu_1, \lambda_1)$ where $\nu_1 = \frac{2}{3}$ and $\lambda_1 = \sqrt{\varepsilon(1+\varepsilon)}$.

Again, we hand over each intermediate polynomial p to SqSS to obtain new polynomials p in x and y of degree at most $D_Q(\nu_2, \lambda_2)$ etc.

After $N = O(\log n)$ iterations involving intermediate polynomials of degrees $D_Q(\nu_k, \lambda_k)$, we eventually reach polynomials of degree at most $D_Q(\vartheta_{n+1}, \lambda_N)$.

Consider the sequence $\lambda'_{k+1} = \frac{3\varepsilon + \lambda'_k}{6\varepsilon\sqrt{\varepsilon}}$ starting at $\lambda'_0 = \lambda_N = \frac{1+\sqrt{5}}{3\sqrt{18}} + o(1)$.

We iterate SqSS $O(\log n)$ additional times; thereby generating polynomials of degree $D_Q(\vartheta_{n+1}, \lambda'_k)$. Eventually, we reach polynomials of degree at most D , i.e. $D_Q(\vartheta_{n+1}, \varepsilon)$.

It remains to combine these relations to write z as a fraction of products of elements of the form $p(\alpha)$ or $p(\beta)$ with the degrees of p bounded by D . Finally, we use the oracle's answer-list $\{p(\alpha)^d, p(\beta)^d\}_{p \in \mathcal{F}_D}$, to retrieve z^d .

3.2 Complexity

We analyze the complexity of each step:

SqSS For $\nu_q > \vartheta_{n+1}$. The sum of the degrees of H_x and H_y is:

$$(d_x + 1)\sqrt{(d_q + d_\varepsilon)/D} + (d_y + 1)\sqrt{(d_q + d_\varepsilon)D} - 2 \simeq 2\sqrt{(d_q + d_\varepsilon)n}$$

As, in addition, q divides one of these two polynomials, our smoothness probability is identical to that of d_p -smooth polynomials of degree $2\sqrt{(d_q + d_\varepsilon)n} - d_q$ i.e.:

$$2\sqrt{(d_q + d_\varepsilon)n} - d_q \leq 2(\sqrt{\lambda_q + \varepsilon} + o(1))n^{\frac{1+\nu_q}{2}} \log_q^{\frac{1-\nu_q}{2}} n \leq D_Q\left(\frac{1+\nu_q}{2}, 2\sqrt{\lambda_q + \varepsilon}\right)$$

And the probability that polynomials of this degree are $D_Q(\frac{1}{6} + \frac{\nu_q}{2}, \frac{2\sqrt{\lambda_q + \varepsilon}}{3\varepsilon})$ -smooth is $L_Q(\frac{1}{3}, -\varepsilon + o(1))$ (cf. to section 2).

SqSS For $\nu_q \leq \vartheta_{n+1}$. The sum of the degrees of H_x and H_y is:

$$\sqrt{nD} + \frac{d_q + d_\varepsilon}{2} + \frac{d_q + d_\varepsilon}{2}\sqrt{\frac{n}{D}} + 1 \simeq \sqrt{nD} + \frac{d_q + d_\varepsilon}{2}\sqrt{\frac{n}{D}}$$

Then,

$$\begin{aligned} \sqrt{nD} + \frac{d_{\mathfrak{q}} + d_{\varepsilon}}{2} \sqrt{\frac{n}{D}} - d_{\mathfrak{q}} &\leqslant \left(\sqrt{\varepsilon} + \frac{\lambda_{\mathfrak{q}}}{2\sqrt{\varepsilon}} + \frac{\sqrt{\varepsilon}}{2} + o(1) \right) n^{\frac{2}{3} + \frac{1}{3 \cdot 2^n}} \log_q^{\frac{2}{3} - \frac{1}{3 \cdot 2^n}} n \\ &\leqslant D_Q \left(\frac{2}{3} + \frac{1}{3 \cdot 2^n}, \frac{3\varepsilon + \lambda_{\mathfrak{q}}}{2\sqrt{\varepsilon}} \right) \end{aligned}$$

Again, the odds that polynomials of this degree are $D_Q(\vartheta_{n+1}, \frac{3\varepsilon + \lambda_{\mathfrak{q}}}{6\varepsilon\sqrt{\varepsilon}})$ -smooth is $L_Q(\frac{1}{3}, -\varepsilon + o(1))$.

In both cases, after enumerating $L_Q(\frac{1}{3}, \varepsilon + o(1))$ polynomials, we expect to find a good polynomial H .

Descent. SqSS calls involve medium-sized \mathfrak{p} -polynomials of degree larger than D . The number of these polynomials is thus bounded by $O(\frac{n}{D})$, i.e. $O(\sqrt[3]{n^2 \log_q^2 n})$.

As the whole process involves $O(\log n)$ SqSS calls, the total number of medium-size polynomials \mathfrak{p} to be considered is:

$$O \left(\frac{n}{\log_q n} \right)^{\frac{2}{3} O(\log n)} = e^{O(\log^2 n)}$$

Running SqSS on each of these polynomials, costs $L_Q(\frac{1}{3}, \varepsilon + o(1))$, which is far larger than $\exp(O(\log^2 n))$; we thus conclude that the descent's complexity is $L_Q(\frac{1}{3}, \varepsilon + o(1))$, having pre-computed a table of $L_Q(\vartheta_{n+1}, \varepsilon + o(1))$ elements $\{\mathfrak{p}_i(\alpha)^d\}_i \cup \{\mathfrak{p}_i(\beta)^d\}_i$.

Since $\log^{\pm 2^{-n}} q^n = 1 + o(1)$

$$\begin{aligned} L_Q(\vartheta_{n+1}, \varepsilon + o(1)) &\simeq \exp \left((\varepsilon + o(1)) (\log^{\frac{1}{3} + \frac{1}{3 \cdot 2^n}} q^n) (\log \log^{\frac{2}{3} - \frac{1}{3 \cdot 2^n}} q^n) \right) \\ &\simeq L_Q(\frac{1}{3}, \varepsilon + o(1)) \end{aligned}$$

Total Complexity. The attack's total complexity is hence $L_Q(\frac{1}{3}, \varepsilon + o(1)) = L_Q(\frac{1}{3}, \sqrt[3]{\frac{4}{9}} + o(1))$, in time, space and oracle accesses.

4 Number Field Sieve Oracle-Assisted SDH Resolution

4.1 The Algorithm

Due to lack of space, we only sketch the two NFS algorithms covering large and medium base-field cardinality-values q . As our algorithms preserve, *mutatis mutandis*, most FFS features, we will mainly focus on the specificities proper to these two cases. We also refer the reader to [11], whose arbitrary e -th roots procedure has to be slightly modified to fit the case under consideration.

Algebraic Side. In the FFS case, thanks to the existence of an *ad-hoc* construction, both sides are rational, i.e. on both side we only encounter polynomials. With the NFS matters get more complicated. Indeed, all known polynomial constructions for NFS procedures involve at least one non-trivial number field.

For example over \mathbb{F}_p the base- m construction yields two polynomials $f_1(x) = x - m$ and $f_2(x)$, such that $p \mid f_2(m)$. Since f_1 is a linear polynomial, it generates a trivial number field: the corresponding side is rational and can be addressed using integers. However, f_2 defines a number field $\mathbb{Q}(\alpha)$ with non trivial unit and class groups.

Consequently, to deal with f_2 's side⁵, we need to work with factorizations into ideals: the *algebraic factor base* consists of ideals of small norm. This change truly complicates the approach. As the ideals in the factor base are usually non-principal, we do not know how to construct an oracle query corresponding to a given ideal.

Instead, we need to find smooth $a - b\alpha$ values which factor into small ideals and “translate” them to integers $a - bm$ for the oracle.

Linear Algebra. To relate these factorizations back to multiplicative relations in \mathbb{F}_Q we must resort to linear algebra, which can be done in two ways:

- Repeating the descent of [11], it is possible (using the oracle's answers) to relate the challenge $z \in \mathbb{F}_Q$ to some $a - bm \in \mathbb{Z}$ such that $a - b\alpha$ factors into small ideals. By solving a linear system over $\mathbb{Z}/(Q - 1)\mathbb{Z}$, this exact factorization can be mapped to a multiplicative combination of oracle answers.

Note that to transform the equality of ideal factorizations into an equality that makes sense in \mathbb{F}_Q , one must include additive characters in $\mathbb{Z}/(Q - 1)\mathbb{Z}$, à la Schirokauer [15]. Since this methodology performs a linear algebra step that depends on the challenge, it therefore yields an algorithm where the linear algebra step has to be redone for each challenge.

- There is also a second method – very specific to our setting. It is possible to use linear algebra to compute virtual SDH values that correspond to each ideal⁶. From a theoretical standpoint, this can be justified as in [8,16]. Note that this second method also requires the inclusion of additive characters à la Schirokauer.

Note that in some cases (NFS-HD), there are two algebraic sides to consider.

We start from a large system of equations, where each equation expresses a value in \mathbb{F}_Q – the equation's generator – as a product of ideals in the number field. Moreover, we are also given the power of this value returned by the oracle. Our goal is to associate to each ideal I a number P_I such that in each equation, the substitution of I by P_I will yield a product whose value matches the oracle's answer for this equation.

⁵ Called the *algebraic side*.

⁶ We assume here that the class number of $\mathbb{Q}(\alpha)$ is co-prime with the relevant prime factors of $Q - 1$. This assumption is clearly expected to hold.

If we can achieve this goal, P_I can be looked upon as an oracle output for I , and we are back in a FFS-like scenario. However, this is not completely straightforward:

A first (costly) option would be to use linear algebra over the integers and express each ideal as a product of generators (possibly with rational exponents) but one may note that the vector of generators and the vector⁷ of ideals are related by a matrix multiplication in the exponent. More precisely, if ideals are indexed from 1 to B , and denoting by G_i the i -th equation generator ($1 \leq i \leq B$), there exists a $B \times B$ matrix \mathbf{A} , such that:

$$\begin{pmatrix} G_1 \\ G_2 \\ \vdots \\ G_B \end{pmatrix} = \mathbf{A} \star \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_B \end{pmatrix} \quad (1)$$

where \star denotes matrix multiplication in the exponent, i.e. where for all i :

$$G_i = \prod_{j=1}^B I_j^{\mathbf{A}_{i,j}} \quad (2)$$

Moreover this matrix multiplication in the exponent is defined modulo $Q - 1$, the cardinality of \mathbb{F}_Q^* . Having observed that, we can use Wiedemann's algorithm [17] to obtain the P_I values. e.g. if \mathbf{A} is invertible, Wiedemann's algorithm finds an expression $\mathbf{A}^{-1} = F(\mathbf{A})$ where F is a polynomial.

From this expression, we obtain:

$$\begin{pmatrix} P_{I_1} \\ P_{I_2} \\ \vdots \\ P_{I_B} \end{pmatrix} = F(\mathbf{A}) \star \begin{pmatrix} \mathcal{O}(G_1) \\ \mathcal{O}(G_2) \\ \vdots \\ \mathcal{O}(G_B) \end{pmatrix} \quad (3)$$

where $\mathcal{O}(G_i)$ denotes the oracle's output for the generator G_i .

This linear algebra step based on [17] has essentially the same complexity as usual, with the small caveat that additions are replaced by multiplications and multiplications by constants are replaced by exponentiations.

4.2 Complexity

The $L_Q(\frac{1}{3}, \sqrt[3]{x})$ -complexities of the NFS and NFS-HD oracle-assisted attacks are expressed in the following table:

variant	oracle accesses	learning phase time	post-learning phase time
NFS-HD	$\frac{48}{91}$	$\frac{384}{91}$	$\frac{384}{91}$
NFS	$\frac{4}{9}$	$\frac{32}{9}$	3

These figures stem from the technical analysis found in the Appendix.

⁷ In this vector, there are also a few additional components corresponding to the additive characters.

5 Cryptanalytic Applications

This section follows very closely Brown and Gallant's [1]. Variable were renamed to ease the identification of on parameters playing specific roles in our algorithms (e.g. secret d , oracle queries x_i etc).

It is clear that any cryptographic protocol where the secret-owner plays the role of a SDHP-oracle will succumb to the new attacks. The technique can apply to primitives as diverse as public-key encryption, digital signature and key retrieval.

5.1 Textbook El-Gamal Encryption

As observed in [1], textbook El-Gamal encryption makes an SDHP oracle available. El-Gamal encryption of message m to a receiver with public-key g^d results in a ciphertext $c = \{c_1, c_2\} = \{g^r, m \cdot g^{dr}\}$.

In a chosen ciphertext attack against an encryption scheme, an adversary \mathcal{A} can select any ciphertext and obtain its decryption. For El-Gamal encryption, if \mathcal{A} chooses $c = \{x, c_2\}$, then he obtains $m = c_2/x^d$. \mathcal{A} can thus obtain $x^d = c_2/m$, thereby transforming the receiver into an SDHP-oracle for the challenge x .

Textbook El-Gamal is already known to be vulnerable to chosen ciphertext attacks. The known attacks mainly allow information to be learned about previously encrypted messages but do not leak anything about d . In the scenario described here, an attacker use the receiver as an oracle to learn how to decrypt future traffic sub-exponentially, but still significantly faster than all previously known SDHP-solving algorithms.

5.2 Ford-Kaliski Key Retrieval

In Ford-Kaliski's key retrieval protocol [5], a client picks a random exponent b and computes $x = g^b$.

x is sent to the server who, replies with $s = x^d$. The client computes $t = \sqrt[b]{s}$ and retrieves the hardened password $t = \sqrt[b]{s} = g^d$.

The protocol's purpose is to transform a low-entropy secret password g into a hardened password g^d involving a longer secret known only to the server. This prevents dictionary attacks on the client's password.

Therefore, the server plays exactly the role of the SDHP oracle considered in this paper.

Variants of this protocol are currently under discussion in two standardization groups [6,7].

5.3 Chaum- Van Antwerpen's Undeniable Signatures

Another protocol in which an SDHP oracle is available is Chaum and van Antwerpen's undeniable signature scheme [2]. The protocol uses a prime modulus $p = 2q + 1$ such that q is prime.

Let $g \in \mathbb{Z}_p^*$ be an element of order q and $y = g^d$ where d is the signer's private key (all other parameters are public). To sign a message x , Alice produces the signature $s = x^d \bmod p$. Bob verifies s interactively:

Bob picks $\{e_1, e_2\}$ at random in \mathbb{Z}_q , and sends $c = s^{e_1}y^{e_2}$ to Alice.

Alice computes $h = \sqrt[d]{c} \bmod p$ and sends it to Bob. Bob accepts s as a valid signature if $h = x^{e_1}g^{e_2} \bmod p$.

Note that [2], provides SDHP oracles for both d and $\frac{1}{d} \bmod q$.

References

1. Brown, D., Gallant, R.: The static Diffie-Hellman problem, Cryptology ePrint Archive, Report 2004/306 (2004), <http://eprint.iacr.org>
2. Chaum, D., van Antwerpen, H.: Undeniable signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
3. Dickman, K.: On the frequency of numbers containing prime factors of a certain relative magnitude. *Ark. Mat. Astr. Fys.* 22, 1–14 (1930)
4. 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)
5. Ford, W., Kaliski, B.: Server-assisted generation of a strong secret from a password. In: Ninth international workshop on enabling technologies - WET ICE 2000. IEEE Press, Los Alamitos (2000)
6. IEEE p1363.2/d23, Draft standard for specifications for password-based public key cryptographic techniques, p. 24 (March 2006)
7. iso 11770-4, Information technology - security techniques - key management - part 4: Mechanisms based on weak secrets, iso (November 2004)
8. Joux, A., Lercier, R.: Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the gaussian integer method, *Mathematics of computation* 242(72), 953–967 (2003)
9. Joux, A., Lercier, R.: The function field sieve in the medium prime case. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 254–270. Springer, Heidelberg (2006)
10. Joux, A., Lercier, R., Smart, N., Vercauteren, F.: The number field sieve in the medium prime case. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 326–344. Springer, Heidelberg (2006)
11. Joux, A., Naccache, D., Thomé, E.: When e -th Roots Become Easier Than Factoring. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 13–28. Springer, Heidelberg (2007)
12. Koblitz, N., Menezes, A.: Another look at non-standard discrete log and Diffie-Hellman problems, Cryptology ePrint Archive, Report 2007/442 (2007), eprint.iacr.org
13. Lenstra, A., Lenstra, H., Manasse, M., Pollard, J.: The number field sieve. In: Lenstra, A., Lenstra, H. (eds.) The development of the number field sieve. LNM, vol. 1554, pp. 11–42. Springer, Heidelberg (1993)
14. Panario, D., Gourdon, X., Flajolet, P.: An analytic approach to smooth polynomials over finite fields. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 226–236. Springer, Heidelberg (1998)
15. Schirokauer, O.: Discrete logarithms and local units. *Philos. Trans. Roy. Soc. London Ser. A* 345(1676), 409–423 (1993)
16. Schirokauer, O.: Virtual logarithms. *J. Algorithms* 57(2), 140–147 (2005)
17. Wiedemann, D.: Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory* IT-32, 54–62 (1986)

A Implementation Details

As our algorithms are sub-exponential, the assessment of their experimental behavior is essential. We hence implemented the FFS and NFS attacks as described in sections 3.1 and 4.

A.1 Function Field Sieve Experiments

The FFS-based attack acquired the capability to compute the d -th power of an arbitrary target in $\mathbb{F}_{2^{1025}}$ after a learning phase of only 76,916,368 SDHP-oracle calls (these requests correspond to irreducible polynomials in α and β of degree up to 29).

We selected:

$$f_1(x) = x^{171} + x^4 + x^3 + x^2 + 1 \quad \text{and} \quad f_2(y) = y^6 + y + 1$$

so that an irreducible polynomial of degree 1025 divides the resultant $f(x)$ of $y - f_1(x)$ and $x - f_2(y)$ with respect to the variable y .

Let σ be the mapping sending an integer p to a polynomial $\sigma(p)$ over \mathbb{F}_2 such that the evaluation at 2 of this polynomial yields p . For instance, $\sigma(b) = x^3 + x + 1$ where b is the hexadecimal integer 0x0b.

Denoting by $\pi = 3.14159\dots$, the attacked instance is:

$$\bar{\pi}(\alpha) = \sigma(\lfloor 2^{1023}\pi \rfloor)(\alpha) = \alpha^{1024} + \alpha^{1023} + \alpha^{1020} + \dots + \alpha^5 + \alpha^4 + \alpha^2$$

As the factors of $x^{18}\bar{\pi}(x) \bmod f(x)$ have degrees 1, 4, 6, 11, 32, 73, 217, 339 and 341, listing here the entire descent tree would require too much space. Instead, we opt to zoom on the path of length 12 starting with a special-q equal to the factor of degree 341 and following recursively, at each node, the highest degree factor. This yields table 1, where the $a_{i,j}$ stand for the following hexadecimal values:

$a_{1,0} = 1de914aa624143ee2880268$	$a_{1,1} = 5368318d2e945f69775022f$
$a_{1,2} = 6b625fb7825342aecdabd80$	$a_{1,3} = 1055c12e550f64c6d8bd2e6$
$a_{2,0} = 22ac2088c59$	$a_{2,1} = 114043dab72$
	$a_{2,2} = 3e6c922af2e$
$a_{3,0} = 3f536e224dd$	$a_{3,1} = 1077f087dba$
$a_{5,0} = 0006a24a2be$	$a_{5,1} = 000164b63c6$
$a_{7,0} = 00026dc2d0b$	$a_{7,1} = 0000b748064$
$a_{9,0} = 000138bc9c0$	$a_{9,1} = 000122f0d95$
$a_{11,0} = 00003972dfb$	$a_{11,1} = 00069fc51c5$
	$a_{12,0} = 0006faf0133$
	$a_{12,1} = 0000d86d785$

The special-q degree in x or y is underlined in the corresponding column and smoothness probabilities are expressed as $2^{-\psi}$.

We wrote our software chain in C, relying upon the computer algebra systems PARI-GP and MAGMA for a handful of specific tasks. The whole computation was run on single Intel Core-2 at 3.6 GHz, with a 16 GB memory. Each SqSS call claimed roughly five minutes and the entire attack took less than a week.

Table 1. An example path in an attack tree for $\mathbb{F}_{2^{1025}}$

$H(x, y)$		ψ
factor degrees of $H(x, f_1(x))$		factor degrees of $H(f_2(y), y)$
$\sigma(a_{1,0}) + \sigma(a_{1,1})y + \sigma(a_{1,2})y^2 + \sigma(a_{1,3})y^3$ 3, 4, 29, 30, 46, 73, 75, <u>341</u>	1, 5, 20, 20, 21, 47, 64, 68, 91, 100, 103	3
$\sigma(a_{2,0}) + \sigma(a_{2,1})y + \sigma(a_{2,2})y^2$ 1, 12, 20, 23, 25, 31, 33, 54, 59, 62, 63	1, 13, 14, 21, 43, 53, <u>103</u>	16
$\sigma(a_{3,0}) + \sigma(a_{3,1})y$ 2, 12, 13, 15, 27, 33, 46, <u>63</u>	1, 12, 15, 16, 17, 23, 27, 43, 43, 44	18
$\sigma(a_{4,0}) + \sigma(a_{4,1})y$ 2, 3, 12, 17, 18, 33, 35, 35, <u>46</u>	2, 6, 10, 11, 12, 13, 29, 35, 38, 39	19
$\sigma(a_{5,0}) + \sigma(a_{5,1})y$ 1, 1, 2, 3, 5, 15, 18, 18, 22, 24, 24, 27, 35	1, 5, 6, 8, 14, 18, 28, 29, 32, <u>39</u>	21
$\sigma(a_{6,0}) + \sigma(a_{6,1})y$ 5, 12, 12, 18, 19, 31, 32, 34, <u>35</u>	1, 2, 4, 9, 14, 18, 19, 20, 23, 27, 27	22
$\sigma(a_{7,0}) + \sigma(a_{7,1})y$ 1, 5, 8, 10, 18, 26, 30, 33, 33, <u>34</u>	1, 2, 5, 6, 6, 9, 10, 16, 17, 22, 23, 25, 30	24
$\sigma(a_{8,0}) + \sigma(a_{8,1})y$ 1, 1, 11, 20, 20, 24, 27, 29, 32, <u>33</u>	1, 6, 7, 9, 17, 23, 24, 25, 25, 30, 30	26
$\sigma(a_{9,0}) + \sigma(a_{9,1})y$ 2, 9, 13, 18, 19, 20, 26, 27, 31, <u>32</u>	18, 22, 24, 24, 26, 26, 29	27
$\sigma(a_{10,0}) + \sigma(a_{10,1})y$ 1, 1, 3, 14, 20, 20, 24, 27, 27, 28, <u>31</u>	1, 4, 7, 16, 21, 25, 27, 28, 30	28
$\sigma(a_{11,0}) + \sigma(a_{11,1})y$ 1, 4, 10, 15, 15, 23, 24, 24, 26, 29, 30	1, 2, 4, 7, 8, 12, 12, 13, 18, 20, 24, 29, <u>30</u>	28
$\sigma(a_{12,0}) + \sigma(a_{12,1})y$ 1, 1, 14, 16, 18, 18, 18, 25, 28, 29, <u>30</u>	1, 2, 3, 10, 16, 18, 22, 23, 25, 27, 28	30

A.2 Number Field Side Experiments

We also experimented the NFS variant in the finite field \mathbb{F}_p , where p is the 516-bit prime $\lfloor 10^{155}\pi + 88896 \rfloor$ (p is the smallest prime above $10^{155}\pi$ such that $\frac{p-1}{2}$ is also prime).

We computed the secret d -th power of an element in \mathbb{F}_Q after a learning phase of 140 million SDHP-oracle queries.

We selected the following polynomials for defining respectively the rational and algebraic sides:

$$\begin{aligned}
f_1 &= x - 0x5bd59a8c1dfa4580bbd2cee \\
f_2 &= 0xf6841ca54cc1267c \cdot x^5 + 0x423b1fc0c94938f26 \cdot x^4 \\
&\quad - 0x2495d40c188755399c7c \cdot x^3 + 0x3b7ed50dd0329dda55051 \cdot x^2 \\
&\quad - 0x23e3eeb7641a7d13c4b182 \cdot x + 0x1a29246950783fb6b6b7b7c7
\end{aligned}$$

Sieving was done using a modified version of J. Franke and T. Kleinjung's lattice sieving code as included in the `ggnfs` software suite. We selected a factor

base comprising the 43,321 prime ideals of norm smaller than 2^{19} and obtained generators factoring over this set of ideals after four sieving minutes using 128 Intel Core-2 CPU cores clocked at 1.6 GHz.

We then used the factor base extension step detailed in [11] and sieved again for 24 minutes, using the same set of processors. This allowed us to relate the vast majority of ideals of norm smaller than 2^{32} to smaller-norm ideals using 140 million oracle queries.

We postponed linear algebra computations after the descent (*cf.* to the two options explained in section 4).

The descent was coded in MAGMA, where the **SqSS** was performed by a modified version of J. Franke and T. Kleinjung’s lattice sieving code, including several modifications pertaining to the descent step (most notably co-factorization of norms using the **gmp-ecm** program).

We started from the challenge value $\tau = 2^{\lfloor \log_2 N \rfloor - \lfloor \log_3 N \rfloor}$. The first decomposition step consisted in finding a related finite field element (e.g. $2^k\tau$ for some k , assuming 2 was an oracle query) which factored suitably over the integers.

We searched during a few minutes for a suitable integer k . The corresponding factorization involved seven primes unlinked to the factor base, the largest of which having 27 digits. Afterwards, using 164 intermediate descent steps (139 on the algebraic side and 25 on the rational side), we managed to link our initial challenge to the factor base.

The final link between our challenge and the oracle queries was obtained by solving a linear system involving the matrix formed by the first set of relations obtained by sieving; the right hand side formed by the ideal factorization stemming from the descent. After adding character maps, we solved this 43,378 rows and columns system in eight hours on four Intel Core-2 CPU cores clocked at 2.4 GHz.

B Complexity Analysis – Regular NFS

Here, we work in \mathbb{F}_Q , with $Q = p^n$ and $p > L_Q(\frac{2}{3})$.

The polynomial construction is done through lattice reduction. First, choose an irreducible polynomial f_1 of degree n and find another polynomial of degree D , multiple of f_1 modulo p . The coefficients of f_2 have size $Q^{1/(D+1)}$, the coefficients of f_1 are essentially of the same size, just slightly larger.

The main parameter is $D = \delta \sqrt[3]{\frac{\log Q}{\log \log Q}}$. This implies that the coefficients of f_1 and f_2 have size $L_Q(\frac{2}{3}, \frac{1}{\delta})$.

B.1 Sieving and Linear Algebra

There is a small and a large side. The complexity is clearly dominated by the large side, corresponding to f_2 . We sieve over elements $a + b\alpha$, map them into the number field and compute their norm $b^D f_2(-\frac{a}{b})$. If a and b are bounded in absolute value by $B = L_Q(\frac{1}{3}, \beta)$, the norm is $L_Q(\frac{2}{3}, \beta\delta + \frac{1}{\delta})$.

As usual, the smoothness probability over a smoothness basis of ideals of prime norm up to $L_Q(\frac{1}{3}, \beta)$ is $L_Q(\frac{1}{3}, -(\beta\delta + \frac{1}{\delta})\frac{1}{3\beta})$. To ensure that we get enough relations, we need:

$$\beta = -\left(\beta\delta + \frac{1}{\delta}\right)\frac{1}{3\beta} \Rightarrow 3\beta^2 - \beta\delta - \frac{1}{\delta} = 0$$

$$\text{which has the positive root: } \beta = \frac{\delta + \sqrt{\delta^2 + \frac{12}{\delta}}}{6}$$

This is minimized for $\delta = \sqrt[3]{\frac{3}{2}}$ i.e. $\beta = \sqrt[3]{\frac{4}{9}}$.

The complexity of the sieving step is thus $L_Q(\frac{1}{3}, \sqrt[3]{\frac{32}{9}})$.

B.2 Descent Initialization: The Descent's Dominating Step

In the initial descent step, we first need to go from norms of size $L_Q(1)$ down to factors of size $L_Q(\frac{2}{3})$. Indeed, by opposition to the FFS setting, here smoothness testing can only be done using ECM factorization. Consequently, this step costs $L_{\text{factor}}(\frac{1}{2}, \sqrt{2})$ of the bound on the factor size.

With factors of size $L_Q(\frac{2}{3})$, this becomes $L_Q(\frac{1}{3})$ and we cannot afford more than that. We use the same strategy as described in [11] and simply randomize the input challenge until it becomes smooth as an integer. As the complexity of pulling-out a factor F with ECM costs $L_F(\frac{1}{2}, \sqrt{2})$, we can analyze the descent from arbitrary numbers with norm $L_Q(1, 1)$ down to factors of size $L_Q(\frac{2}{3}, \theta)$.

Smoothness probability is $L_Q(\frac{1}{3}, \frac{\theta}{3})$ and ECM's cost is $L_Q(\frac{1}{3}, \sqrt{\frac{4\theta}{3}})$. Thus, the total work-factor per smooth value is $L_Q(\frac{1}{3}, \frac{\theta}{3} + \sqrt{\frac{4\theta}{3}})$.

This is minimized for $\theta^3 = \frac{1}{3}$ and costs $L_Q(\frac{1}{3}, \sqrt[3]{3})$.

B.3 Continuing the Descent

In the middle of the descent, we consider a special- q of size $L_Q(\lambda, \mu)$ with $\frac{1}{3} < \lambda < \frac{2}{3}$. We sieve over polynomials of degree:

$$T = t \left(\frac{\log Q}{\log \log Q} \right)^{\frac{\lambda - \frac{1}{3}}{2}}$$

and with coefficients of size $S = L_Q((\lambda + \frac{1}{3})\frac{1}{2}, s)$. The size of the sieving space is $L_Q(\lambda, st - \mu)$. Since $L_Q(\frac{1}{3})$ freedom should suffice, we can make st tends to μ from upward.

The f_1 norm is $L_Q(\frac{1}{2} + \frac{\lambda}{2}, \frac{t}{\delta})$ and the f_2 norm is $L_Q(\frac{1}{2} + \frac{\lambda}{2}, \frac{t}{\delta} + s\delta)$.

Thus the total norm after dividing by the special- q , which can be ignored is, $L_Q(\frac{1}{2} + \frac{\lambda}{2}, \frac{2t}{\delta} + s\delta)$. Minimizing this under the constraint $st = \mu$ yields $t = \delta \sqrt{\frac{\mu}{2}}$. This step is clearly not the dominating task in the descent. In fact, we can adapt the descent's speed to the specific problem at hand. The only restriction is that the descent should be fast enough to only get a relatively small number of values at the final step of the descent.

B.4 Concluding the Descent

In the final descent step, we need to express special- q values of size immediately above $L_Q(\frac{1}{3}, \beta)$ in the smoothness bases up to $L_Q(\frac{1}{3}, \beta)$. We sieve over polynomials $a + bx$ with a and b bounded by $L_Q(\frac{1}{3}, \frac{\beta+\epsilon}{2})$, with a total sieving space size equal to $L_Q(\frac{1}{3}, \epsilon)$.

At the f_2 side, the norm's size is computed as above and yields $L_Q(\frac{2}{3}, (\beta + \epsilon)\frac{\delta}{2} + \frac{1}{\delta})$.

At the f_1 side, only the constants matter and the norm has size $L_Q(\frac{2}{3}, \frac{1}{\delta})$. Thus the total norm is $L_Q(\frac{2}{3}, (\beta + \epsilon)\frac{\delta}{2} + \frac{2}{\delta})$ and is left unchanged when taking into account the special- q that can be removed. The smoothness probability in basis $L_Q(\frac{1}{3}, \beta)$ is:

$$L_Q\left(\frac{1}{3}, \frac{1}{3}\left(\frac{(\beta + \epsilon)\delta}{2\beta} + \frac{2}{\beta\delta}\right)\right)$$

This is constrained by the fact that we need enough sieving space, i.e. by the relation:

$$3\epsilon = \frac{(\beta + \epsilon)\delta}{2\beta} + \frac{2}{\beta\delta},$$

which yields $\epsilon \simeq 1.27$ as in [11], which does not dominate the overall asymptotic complexity of the descent step.

C Complexity Analysis – High Degree NFS

Here, we work in \mathbb{F}_Q , with $Q = p^n$ and $L_Q(\frac{2}{3}) > p > L_Q(\frac{1}{3})$. We write $p = L_Q(\lambda, 1)$.

The polynomial construction is as follows. First, choose a parameter $0 < \alpha < \frac{1}{2}$, which value will be determined later, on and an integer A close to p^α . Using Euclidean division write $p = BA + r_p$ with $B \simeq p^{1-\alpha}$ and $0 \leq r_p < A$.

Then, find an irreducible polynomial f_1 of degree n , which coefficients are either small numbers or a small multiples of A . All coefficients of f_1 have a size bounded by p^α . Finally, let $f_2 = Bf_1 \bmod p$, all coefficients of f_2 are either small multiples of B or small multiples of r_p . Hence, their size is at most of order $p^{1-\alpha}$.

C.1 Sieving and Linear Algebra

Again, complexity is bounded by the large side, given by f_2 .

We sieve over polynomials of degree D defined by:

$$D = \delta \left(\frac{\log Q}{\log \log Q} \right)^{\frac{2}{3}-\lambda}$$

with coefficients of size $L_Q(\lambda - \frac{1}{3}, \mu)$. The total sieving space size is $L_Q(\frac{1}{3}, \delta\mu)$.

At the f_2 side, the norm is $L_Q(\frac{2}{3}, \delta(1 - \alpha) + \mu)$. For a given sieving space size, this is minimized when $\delta(1 - \alpha) = \mu$. With a smoothness basis bounded by

$L_Q(\frac{1}{3}, \beta)$, the smoothness probability is $L_Q(\frac{1}{3}, -\frac{2\mu}{3\beta})$. To balance the complexities of the sieving and the linear algebra we have:

$$\beta = \frac{2\mu}{3\beta} \quad \text{i.e.} \quad 3\beta^2 = 2\mu.$$

In addition, to ensure that we get enough relations, we need:

$$\beta + \frac{2\mu}{3\beta} = (1 - \alpha)\mu^2 \quad \text{i.e.} \quad 2\beta = (1 - \alpha)\mu^2.$$

Put together, this yields $\beta = \sqrt[3]{\frac{8}{9}(1 - \alpha)}$. And, the complexity of the sieving step is $L_Q(\frac{1}{3}, \sqrt[3]{\frac{64}{9}(1 - \alpha)})$.

C.2 Descent Initialization

In the initial descent step, as in the regular NFS case, we first need to go from norms of size $L_Q(1)$ down to factors of size $L_Q(\frac{2}{3})$. However, since f_1 's coefficients are of order p^α , the norm of a generic element, with degree n and coefficient modulo p is larger than Q . This norm is in fact, of order $L_Q(1, 1 + \alpha)$.

Using the ECM to go to factors of size $L_Q(\frac{2}{3}, \theta)$ costs $L_Q(\frac{1}{3}, \frac{1+\alpha}{3\theta} + \sqrt{\frac{4\theta}{3}})$.

This is minimized for $\theta^3 = (1 + \alpha)^{\frac{2}{3}}$ and costs $L_Q(\frac{1}{3}, \sqrt[3]{3(1 + \alpha)})$.

Equilibrating the complexities of the sieving and of the descent's initialization, we get:

$$3(1 + \alpha) = \frac{64(1 - \alpha)}{9} \quad \text{i.e.} \quad \alpha = \frac{37}{91}.$$

For this α the complexities of the sieving and of the descent's initialization become:

$$L_Q(\frac{1}{3}, \sqrt[3]{\frac{384}{91}}) \simeq L_Q(\frac{1}{3}, 1.62)$$

C.3 Continuing the Descent

In the middle of the descent, we encounter a special- q of size $L_Q(\lambda', \mu')$ with $\frac{1}{3} < \lambda' < \frac{2}{3}$. We sieve over polynomials of degree:

$$T = t \left(\frac{\log Q}{\log \log Q} \right)^{\lambda_t}$$

and with coefficients of size $S = L_Q(\lambda_s, s)$. The two values λ_s and λ_t are related and sum to λ' . More precisely, we choose:

$$\begin{aligned} \lambda_t &= \frac{\lambda' + 1}{2} - \lambda \\ \lambda_s &= \frac{\lambda' - 1}{2} + \lambda \end{aligned}$$

The sieving space size is $L_Q(\lambda', st - \mu')$. Since $L_Q(\frac{1}{3})$ freedom should suffice, we can make st tends to μ' from upward. Again, there is enough freedom here and this step is not limiting.

C.4 Concluding the Descent

In the final descent step, we need to express special- q values of size immediately above $L(\frac{1}{3}, \beta)$ in the smoothness bases up to $L(\frac{1}{3}, \beta)$. We sieve over polynomials of degree D defined by:

$$D = \delta \left(\frac{\log Q}{\log \log Q} \right)^{\frac{2}{3} - \lambda}$$

with coefficients of size $L_Q(\lambda - \frac{1}{3}, \mu)$. The total sieving space has size $L_Q(\frac{1}{3}, \delta\mu - \beta)$. Here the total norm does not depend on α and is $L_Q(\frac{2}{3}, \delta + 2\mu)$. Moreover, the norm's expression is left unchanged when removing the special- q contribution.

The smoothness probability in basis $L_Q(\frac{1}{3}, \beta)$ is

$$L_Q \left(\frac{1}{3}, \frac{\delta + 2\mu}{3\beta} \right)$$

Again, this is constrained by the fact that we need enough sieving space, i.e. by the relation:

$$\frac{\delta + 2\mu}{3\beta} = \delta\mu - \beta$$

Choosing $\delta = 2\mu$, we get:

$$2\mu^2 - \frac{4\mu}{3\beta} - \beta = 0.$$

which yields $\mu \simeq 1.13$. The total complexity $L_Q(\frac{1}{3}, 2\mu^2 - \beta) \simeq L_Q(\frac{1}{3}, 0.93)$ is, again, not dominating.

An Improvement to the Gaudry-Schost Algorithm for Multidimensional Discrete Logarithm Problems

Steven Galbraith^{1,*} and Raminder S. Ruprai^{2,**}

¹ Mathematics Department, Auckland University, Auckland, New Zealand
s.galbraith@math.auckland.ac.nz

² Mathematics Department, Royal Holloway University of London,
Egham, Surrey, UK
r.s.ruprai@rhul.ac.uk

Abstract. Gaudry and Schost gave a low-memory algorithm for solving the 2-dimensional discrete logarithm problem. We present an improvement to their algorithm and extend this improvement to the general multidimensional DLP. An important component of the algorithm is a multidimensional pseudorandom walk which we analyse thoroughly in the 1 and 2 dimensional cases as well as giving some discussion for higher dimensions.

Keywords: discrete logarithm problem (DLP).

1 Introduction

The discrete logarithm problem (DLP) is: Given a cyclic group G (in additive notation) of order r and $P, Q \in G$ where P is a generator of the group, find a positive integer n such that $Q = [n]P$. This is an important computational problem due to applications in public key cryptography. In a generic group there are standard algorithms for solving the DLP such as Baby-Step Giant-Step (BSGS) [18] (see, for example, [19]) and Pollard rho [14]. Van Oorschot and Wiener [21,22] give a version of Pollard Rho, suitable for distributed computing, based on the idea of distinguished points. For further works on Pollard rho see [3,20]. BSGS solves the DLP in $O(\sqrt{r})$ group operations but also requires $O(\sqrt{r})$ group elements of storage. Pollard rho and van Oorschot-Wiener also solve the DLP in heuristic *expected* $O(\sqrt{r})$ group operations but require only constant storage. We also refer to [10] for a rigorous analysis of the Pollard rho algorithm.

A higher dimensional version of the DLP arises in a number of applications (see Section 6). We now give a precise definition of it.

* This work supported by EPSRC grant EP/D069904/1.

** The work described in this report has in part been supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Definition 1 (Multidimensional DLP). Let G be an abelian group and let $P_1, P_2, \dots, P_d, Q \in G$ and $N_1, \dots, N_d \in \mathbb{N}$ be given. The d -dimensional discrete logarithm problem is to find (if they exist) integers $n_i \in [-N_i, N_i]$, for $1 \leq i \leq d$, such that

$$Q = [n_1]P_1 + [n_2]P_2 + \dots + [n_d]P_d. \quad (1)$$

We call (n_1, \dots, n_d) the exponent vector of Q . We write $\tilde{N}_i = 2N_i + 1$ and

$$N = \prod_{i=1}^d \tilde{N}_i. \quad (2)$$

Note that G may or may not be cyclic. In the case of a cyclic group of prime order and $\tilde{N}_i = \#G$ this computational problem is the representation problem introduced by Brands [2] in 1993.

It is easy to adapt the BSGS algorithm to the multidimensional case, as shown by Matsuo et. al. [12], and this algorithm requires $O(\sqrt{N})$ group operations and $O(\sqrt{N})$ group elements of storage. Pollard describes his kangaroo method in [14,15] to solve the 1-dimensional case (i.e., $d = 1$) also in expected $O(\sqrt{N})$ group operations but requiring only constant storage. Van Oorschot and Wiener [21,22] also give a version of the kangaroo method which is suitable for distributed computing. It seems unlikely that the kangaroo algorithm can be adapted to the case of dimension $d \geq 2$.

Gaudry and Schost [9] present an algorithm to solve the 2-dimensional case using random walks and distinguished points. Unlike the kangaroo method, their algorithm is analysed using a variant of the Birthday Paradox. They also analyse their algorithm in the 1-dimensional case. We give brief details of their algorithm in Section 2 as well as extending their approach to any dimension. Theorem 2 generalises the result of Gaudry and Schost. Note that the analysis of the Gaudry-Schost algorithm is only heuristic; to be able to easily state theorems we consider an idealised model.

In Section 3 we present a simple idea which improves the Gaudry-Schost algorithm in all dimensions. In Section 4 we give more details about how to put the improved algorithm into practice by correctly choosing parameters. In Section 5 we discuss the implementation issues in higher dimensions and then look at some applications of solving the multidimensional DLP in Section 6.

We thank John Pollard, Pierrick Gaudry and the referees for good advice.

2 The Gaudry-Schost Algorithm

We first recall the Pollard kangaroo algorithm for the 1-dimensional DLP. In other words, we have $Q = [n_1]P_1$ with $-N_1 \leq n \leq N_1$ (and $N = 2N_1 + 1 \approx 2N_1$). The algorithm finds two integers $a, b \in \mathbb{N}$ such that $[a]P_1 = Q + [b]P_1$ and hence solves the DLP.

The crucial idea is to use a “deterministic pseudorandom walk”. More precisely, a selection function $S : G \rightarrow \{1, \dots, n_S\}$ is chosen, which can be

interpreted as partitioning the group into n_S sets of roughly equal size. Suitable integers z_1, \dots, z_{n_S} are chosen (e.g., uniformly chosen from $[0, c_1\sqrt{N}]$ for some constant c_1) and, given $x_1 \in G$, the deterministic pseudorandom walk $x_1, x_2, \dots \in G$ proceeds as

$$x_{i+1} = x_i + [z_{S(x_i)}]P_1.$$

Pollard uses a *tame walk* which starts at $x_1 = [N_1]P$ and takes $n = O(\sqrt{N})$ steps before stopping at x_n . Pollard also uses a *wild walk* which starts at $y_1 = Q$ and also takes $O(\sqrt{N})$ steps. If the wild walk lands on a footprint of the tame walk then the wild walk follows the path of the tame walk and eventually we find $y_m = x_n$. By storing not just the values x_n and y_m but also the corresponding integers a and b such that $x_n = [a]P_1$ and $y_m = Q + [b]P_1$ we can solve for the DLP.

Van Oorschot and Wiener [21,22] give a version of this algorithm which is not only distributed but has better expected running time. The idea is to use distinguished points, so instead of just storing a single end-point (x_n, a) one stores a moderate number of intermediate values. Indeed, van Oorschot and Wiener showed that the expected running time is heuristically $2\sqrt{N} + 1/\theta$ group operations for storage requirement approximately $2\theta\sqrt{N}$ group elements (throughout the paper, θ denotes the probability that a group element is a distinguished point).

The Gaudry and Schost algorithm is motivated by the above ideas but the analysis is very different. Before we describe the Gaudry-Schost algorithm we must first define the following sets of exponents.

Definition 2. Let notation be as in Definition 1 and suppose $Q = [n_1]P_1 + \dots + [n_d]P_d$ for some integers $n_i \in [-N_i, N_i]$. Define the Tame set T and the Wild set W , which are subsets of \mathbb{Z}^d , by

$$T = \{(a_1, a_2, \dots, a_d) \in \mathbb{Z}^d : a_i \in [-N_i, N_i] \text{ for all } 1 \leq i \leq d\},$$

$$W = (n_1, n_2, \dots, n_d) + T = \{(n_1 + a_1, \dots, n_d + a_d) : (a_1, \dots, a_d) \in T\}.$$

The exponent vector of Q lies somewhere in the tame set T . The wild set W is a translation of T which is centred on the exponent vector of Q . The sets T and W are orthotopes in \mathbb{Z}^d (i.e., d -dimensional products of intervals).

The basic idea of the Gaudry-Schost algorithm [9] is the same as the kangaroo algorithm of Pollard in the van Oorschot and Wiener formulation. Let the multidimensional DLP problem instance of dimension d be given as in Definition 1. We run a large number of pseudorandom walks (possibly distributed over a large number of processors). Half the walks are “tame walks”, which means that every element in the walk is of the form $[a_1]P_1 + [a_2]P_2 + \dots + [a_d]P_d$ where the integer tuple $(a_1, a_2, \dots, a_d) \in T$ (though note that with very small probability some walks will go outside T). The other half are “wild walks”, which means that every element is of the form $Q + [b_1]P_1 + [b_2]P_2 + \dots + [b_d]P_d$ where the integer tuple $(b_1, b_2, \dots, b_d) \in T$. Each walk proceeds until a distinguished point is hit. This distinguished point is then stored in an easily searched structure (e.g., a binary

tree), together with the corresponding d -tuple of exponents. We maintain two such structures: one to store the points found by tame walks and one for the wild walks. When the same distinguished point is visited by two different types of walk we have the collision $[a_1]P_1 + [a_2]P_2 + \dots + [a_d]P_d = Q + [b_1]P_1 + [b_2]P_2 + \dots + [b_d]P_d$ and one solves the multidimensional DLP as follows

$$Q = [a_1 - b_1]P_1 + [a_2 - b_2]P_2 + \dots + [a_d - b_d]P_d.$$

A significant difference between the Gaudry-Schost algorithm and the kangaroo algorithm is that when a distinguished point is hit, Gaudry and Schost restart the walk from a random starting point in the appropriate set, whereas the kangaroos keep on running. The theoretical analysis is different too: Gaudry and Schost use a variant of the birthday paradox whereas Pollard and van Oorschot and Wiener use a different probabilistic argument (based on the mean step size).

We now present the theoretical analysis of the Gaudry-Schost algorithm. Our main result in this section is Theorem 2 which generalises the results of Section 3.1 and 4.1 of [9]. We first recall a tool from probability theory which we will need (this result was also used by Gaudry and Schost) and which we call the *Tame-Wild Birthday Paradox*.

Theorem 1. *When sampling uniformly at random from a set of size N , with replacement, and alternately recording the selected elements in two different lists, then the expected number of selections that need to be made in total before we have a coincidence between the lists is $\sqrt{\pi N} + O(1)$.*

Proof. See Nishimura and Sibuya [13] or [17]. □

Since tame walks lie in T (with high probability) and wild walks lie in W , a collision between tame and wild walks can only occur in $T \cap W$. We call this set the *overlap* and denote its size by M . We will therefore apply Theorem 1 in $T \cap W$ only. To get a rough idea of the running time of the Gaudry-Schost algorithm we consider an idealised version of it which includes two simplifying assumptions. First, we assume that the points in T and W are chosen uniformly at random, rather than using a pseudorandom walk. Second, we assume that all points are stored (in other words, all points are distinguished) so that a tame-wild collision is detected immediately.

Theorem 2. *Given the multidimensional discrete logarithm problem as described in Definition 1 with N the cardinality of the search space as given by equation (2). Then the expected number of group operations (in the idealised model) in the worst case of the original Gaudry-Schost algorithm is*

$$2^{d/2} \sqrt{\pi N}, \tag{3}$$

and the average case expected number of group operations is

$$((4 - 2\sqrt{2})^d + o(1))\sqrt{\pi N}. \tag{4}$$

When $d = 1$ the worst and average case running times are approximately $2.51\sqrt{N}$ and $2.08\sqrt{N}$ group operations. When $d = 2$ the worst and average case running times are approximately $3.54\sqrt{N}$ and $2.43\sqrt{N}$.

Proof. In the worst case Q lies in a ‘corner’ of the search space and so the overlap between the original tame and wild sets has cardinality $M = \frac{N}{2^d}$. Therefore we expect only about $1/2^d$ of the points sampled to be in $T \cap W$. The running time is therefore 2^d times the expected number of elements sampled in $T \cap W$ to get a collision. Using the Tame-Wild Birthday Paradox given in Theorem 1, the expected number of group operations in the worst case is therefore given by

$$2^d \sqrt{\pi M} = 2^{d/2} \sqrt{\pi N}$$

which proves the result presented in equation (3). To find the average case expected running time we have to average over all possible Q . Without loss of generality let us consider one of the 2^d possible orthonts (i.e., sub-orthotopes corresponding to one corner of the search space). In other words, suppose that

$$Q = [x_1 \tilde{N}_1]P_1 + [x_2 \tilde{N}_2]P_2 + \dots + [x_d \tilde{N}_d]P_d$$

with $x_i \in [0, 1/2]$. The cardinality of the overlap between T and W is

$$M = \left(\frac{1}{2} + x_1 \right) \tilde{N}_1 \cdot \left(\frac{1}{2} + x_2 \right) \tilde{N}_2 \cdots \left(\frac{1}{2} + x_d \right) \tilde{N}_d = \left(\prod_{i=1}^d \left(\frac{1}{2} + x_i \right) \right) N.$$

Therefore we expect only about

$$M/N = \prod_{i=1}^d \left(\frac{1}{2} + x_i \right)$$

of the walks to be in $T \cap W$. So the average case expected running time is approximately

$$\begin{aligned} & 2^d \int_{x_1=0}^{1/2} \int_{x_2=0}^{1/2} \cdots \int_{x_d=0}^{1/2} \prod_{i=1}^d \left(\frac{1}{2} + x_i \right)^{-1/2} \sqrt{\pi N} dx_1 dx_2 \dots dx_d \\ &= 2^d \sqrt{\pi N} \left(\int_{x=0}^{1/2} \left(\frac{1}{2} + x \right)^{-1/2} dx \right)^d = (4 - 2\sqrt{2})^d \sqrt{\pi N} \end{aligned}$$

which proves the result presented in equation (4). \square

3 The Improved Gaudry-Schost Algorithm

We now give the main result of the paper, which is a version of the Gaudry-Schost algorithm which has a faster running time. The key observation is that the running time of the Gaudry-Schost algorithm depends on the size of the overlap between the tame and wild sets. If it is possible to make the size of this overlap constant for all possible Q then the expected running time will be constant for all problem instances. We achieve this by choosing walks which only cover certain subsets of T and W .

Precisely, instead of using the sets T and W of the previous section we use an orthotope T' of size $(2/3)^d N$ centered in T and a space W' of the same size but split into 2^d disjoint sets in the ‘corners’ of W . In the 1-dimensional case we have $T' = [-2N_1/3, 2N_1/3] \cap \mathbb{Z} \subseteq T$ and $W' = ([n_1 - N_1, n_1 - N_1/3] \cup [n_1 + N_1/3, n_1 + N_1]) \cap \mathbb{Z} \subseteq W$ (see Figure 1). The algorithm proceeds in exactly the same way as before.

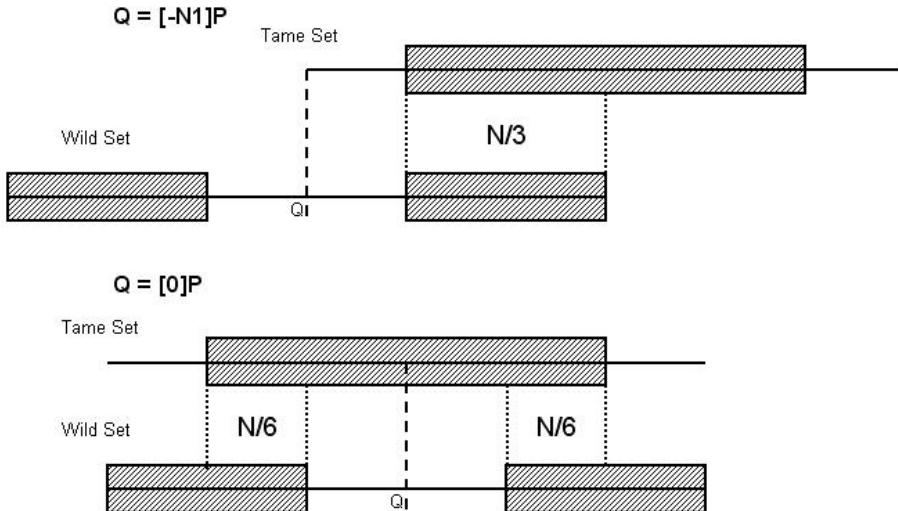


Fig. 1. Depiction of T, W, T' and W' for two extreme cases of Q . The thin horizontal lines denote T and W while the latter shaded rectangles denote T' and W' .

For the theoretical analysis we again use the idealised model where we ignore the issue of pseudorandom walks and assume that we are storing every element visited.

Theorem 3. *Given the multidimensional discrete logarithm problem as described in Definition 1 with N the cardinality of the search space as given by equation (2). Then the expected number of group operations (in the idealised model) for the improved Gaudry-Schost algorithm is*

$$\left(\frac{2^d}{3^{d/2}} + o(1) \right) \sqrt{\pi N}.$$

This is the expected running time in the best, worst and average cases. When $d = 1$ and $d = 2$ this is approximately $2.05\sqrt{N}$ and $2.36\sqrt{N}$ group operations respectively.

Proof. We search in an orthotope of size $k^d N$ centred in the middle of the tame set and we search a space of the same size but split into 2^d disjoint sets in the

‘corners’ of the wild set. Each of these disjoint sets of size $((k/2)^d + o(1))N$. We now compute the volume of the overlap $T' \cap W'$ which is a union of orthotopes. Since the volume of an orthotope is the product of the lengths of its sides our problem reduces to the 1-dimensional case.

It can be easily shown that to make the overlap constant for all problem instances one must take $k = 2/3$ (see Figure 1). Then for all problem instances the cardinality of the overlap is $M' = \frac{N}{3^d}$ and the size of the new search space is given by $N' = \frac{2^d N}{3^d}$. Therefore the proportion of steps in $T' \cap W'$ is approximately

$$\frac{M'}{N'} = \frac{1}{2^d}.$$

Using the Tame-Wild Birthday Paradox analysis, the expected number of group operations before a collision is approximately

$$2^d \sqrt{\pi M'} = 2^d \sqrt{\pi \frac{N}{3^d}} = \frac{2^d}{3^{d/2}} \sqrt{\pi N}.$$

This completes the proof. \square

Theorem 3 is an improvement on the average and the worst case expected running time of the Gaudry-Schost algorithm in all dimensions. In the 2-dimensional case the expected running time falls from $2.43\sqrt{N}$ group operations in the original Gaudry-Schost to $2.36\sqrt{N}$ group operations in the improved Gaudry-Schost. In the worst case the improvement is even bigger from $3.54\sqrt{N}$ to $2.36\sqrt{N}$. The benefit of this new approach increases with d .

In the 1-dimensional case the original Gaudry-Schost algorithm was not competitive with the van Oorschot and Wiener variant of the Pollard kangaroo method. Our improvement is also not competitive.

4 Pseudorandom Walks and Counting Bad Points

In this section we consider some problems which arise in practice and the techniques used by Gaudry and Schost to combat them. Firstly we do not want to store every point visited, which is where the idea of distinguished points comes into play. Denote by θ the probability that an element of the group is a distinguished point. In practice the value of θ is chosen as a tradeoff between the work of each client required to find a distinguished point (approximately $1/\theta$ group operations) and the total storage on the server (proportional to $\theta\sqrt{N}$ group elements). To have an algorithm whose expected number of group elements of storage is constant one would set $\theta = c/\sqrt{N}$ for some constant c .

The more serious assumption in the idealised model is that elements are selected from the tame and wild sets uniformly at random. The Gaudry-Schost algorithm uses a deterministic pseudorandom walk. Our experiments suggest one can choose walks which behave “close enough” (in the sense that the Tame-Wild Birthday Paradox seems to hold) to selecting uniformly at random. In

practice, it seems to be impossible to design pseudorandom walks which cover T or W uniformly but which do not sometimes overstep the boundaries. Steps outside the regions of interest cannot be included in our probabilistic analysis and so such steps are “wasted”. We call these “type 2” bad points. Another assumption of the idealised model is that tame-wild collisions are always detected. In principle it can happen that a walk takes an exceptionally long time to hit a distinguished point (i.e., that a large number of consecutive steps happen to not be distinguished; this phenomena can also be caused by cycles, but that is not our concern in this section). In the parallelised setting we should assume that individual processors may be relatively slow and so, in principle, it could happen that some processor never finds a distinguished point. Such steps are therefore also wasted and we call them “type 1” bad points.

To guard against bad steps of type 1 van Oorschot and Wiener [22] set a maximum number of steps in a walk. If a processor takes this many steps without hitting a distinguished point then it restarts on a new random starting point. They choose this maximum to be $20/\theta$ steps and show that the proportion of bad points of type 1 is at most 5×10^{-8} . This result applies in all dimensions.

We now consider bad points of type 2. This depends on both the pseudorandom walk and the dimension. The main idea is to start walks in proper subsets of T' and W' and to set up these regions so that there is good chance to cover all of T' and W' but also so that the probability that a walk goes outside T' and W' is relatively small.

4.1 1-Dimensional Case

In the 1-dimensional case we use a standard pseudorandom walk as used by Pollard [15] and van Oorschot and Wiener [22] i.e., small positive jumps in the exponent. The average distance in the exponent traveled by a walk is m/θ , where m is the mean step size and θ is the probability that a point is distinguished. For example, one may have $m = c_1\sqrt{N}$ and $\theta = c_2/\sqrt{N}$ for some constants $0 < c_1 < 1$ and $1 < c_2$. Therefore, to reduce the number of bad steps of type 2 we do not start walks within this distance of the right hand end of the interval. In other words, we do not start walks in the following subset of T' .

$$\left[\frac{2N_1}{3} - \frac{m}{\theta}, \frac{2N_1}{3} \right] \cap \mathbb{Z} \quad (5)$$

The analogous omitted subsets for W are the following

$$\left[n_1 - \frac{N_1}{3} - \frac{m}{\theta}, n_1 - \frac{N_1}{3} \right] \cap \mathbb{Z} \text{ and } \left[n_1 + N_1 - \frac{m}{\theta}, n_1 + N_1 \right] \cap \mathbb{Z}. \quad (6)$$

Lemma 1. *Let m be the mean step size and max the maximum step size. Let the subsets where walks will not start be given by equations (5) and (6). The probability that a walk has bad points of type 2 is at most*

$$p = \frac{20 \max - m}{2N_1\theta/3 - m}. \quad (7)$$

Proof. Let $T'' = [-2N_1/3, 2N_1/3 - m/\theta] \cap \mathbb{Z}$ be the set of possible starting points of tame walks and let W'' be the set of possible starts points of wild walks. One expects there to be at least twice as many wild walks with bad points of type 2 as tame walks. Hence it is sufficient to consider the probability for wild walks only.

Let

$$X = [n_1 + N_1/3, n_1 + N_1 - m/\theta]$$

be one of the components of W'' . Note that $\#X = 2N_1/3 - m/\theta$. Since walks travel distance at most $20 \max/\theta$ the only walks which can possibly have bad points of type 2 are ones which start in $[n_1 + N_1 - 20 \max/\theta, n_1 + N_1 - m/\theta]$. Hence the probability that a walk starting in X has bad points of type 2 is

$$\frac{20 \max/\theta - m/\theta}{\#X}.$$

The result follows. \square

For the values $m = c_1\sqrt{N_1}$, $\max = 2m$ and $\theta = c_2/\sqrt{N_1}$ the value of p in equation (7) is $39c_1/(2c_2/3 - c_1)$ which can be made arbitrarily small by taking c_2 sufficiently large (i.e., by storing sufficiently many distinguished points). Hence, even making the over-cautious assumption that all points are wasted if a walk contains some bad points of type 2, the expected number of walks in the improved Gaudry-Schost algorithm can be made arbitrarily close to the desired value when N is large. In practice it is reasonable to store at least 2^{30} distinguished points, which is quite sufficient to minimise the number of bad points of type 2.

We stress that the choices of m , \max and θ are not completely arbitrary. Indeed, if one wants to bound the number of bad points of type 2 as a certain proportion of the total number of steps (e.g., 1%) then for a specific N there may be limits to how large θ and \max can be. For smaller N we cannot have too small a probability of an element being distinguished or too large a mean step size.

4.2 2-Dimensional Case

In the 2-dimensional case, Gaudry and Schost [9] use a walk which goes forwards with respect to one axis and side-to-side in the other. In other words, each step of the walk adds a group element of the form $[a]P_1 + [b]P_2$ where $0 \leq a$ is typically very small (possibly zero sometimes) and $b \in \mathbb{Z}$. Our experience suggests that it is sufficient in practice to use walks of the following form (at least, when $N_1 \approx N_2$ and $N_1 > N_2$).

Definition 3. Let $S : G \rightarrow \{1, \dots, n_S\}$ partition G . Let m_2 be a parameter (the mean absolute step size). Let $z_1, \dots, z_{n_S} \in \mathbb{Z}$ be chosen uniformly at random in the interval $[-2m_2, 2m_2]$. The pseudorandom walk from a given value $x_i \in G$ is

$$x_{i+1} = x_i + [1]P_1 + [z_{S(x_i)}]P_2$$

In the P_1 component, as every step is of size 1, the size of the subsets of exponents where walks do not start will be the expected walk length, $\frac{1}{\theta}$, from the right hand edge of the interval of exponents. Since $m = \max = 1$ in this case, one can apply Lemma 1 directly.

Lemma 2. *Let notation be as above. The probability that a walk has bad points of type 2 in the P_1 component is at most*

$$\frac{19}{2N_1\theta/3 - 1}.$$

If $N_1 \approx N_2$ then $\theta = c/N_1$ and this probability is $19/(2c/3 - 1)$ which can be made arbitrarily small by choosing c to be large (i.e., storing many distinguished points).

The analysis of the bad points of type 2 in the P_2 component is different. We use the following result by Cofman, Flajolet, Flatto and Hofri [4] to address this problem (Gaudry and Schost [9] use a similar result in their analysis).

Lemma 3. *Let y_0, y_1, \dots, y_n be a symmetric random walk that starts at the origin ($y_0 = 0$) and takes steps uniformly distributed in $[-1, +1]$ then the expected value of $\max\{|y_i| : 0 \leq i \leq n\}$ is*

$$\sqrt{\frac{2n}{3\pi}} + O(1).$$

Note that the mean absolute step size in this walk is $\frac{1}{2}$.

We prohibit walks from starting in intervals of length

$$\delta = 2m_2\sqrt{\frac{2}{3\pi\theta}}. \quad (8)$$

at each edge (in the P_2 direction) of the tame and wild regions. To be precise, the region in which tame walks are permitted to start is

$$T'' = ([-2N_1/3, 2N_1/3 - 1/\theta] \times [-2N_2/3 + \delta, 2N_2/3 - \delta]) \cap \mathbb{Z}^2.$$

Note that $\#T'' = (4N_1/3 - 1/\theta)(4N_2/3 - 2\delta)$.

Lemma 4. *Suppose a pseudorandom walk as in Definition 3 is being used. Let m_2 be the mean absolute step size in the P_2 component, and \max_2 the maximum absolute step size. Let δ be as in equation (8). The probability that a walk has bad steps of type 2 in the P_2 component is at most*

$$\frac{40 \max_2}{(2N_2/3 - 2\delta)\theta}. \quad (9)$$

A sharper version of this result, with a more complicated proof, is given in Lemma 5.2.11 of [17].

Proof. As with the earlier results, it suffices to deal with wild walks. Consider the projection of one component of the wild set W'' onto the P_2 axis, for example

$$X = [N_2/3 + \delta, N_2 - \delta] \cap \mathbb{Z}.$$

Note that $\#X = 2N_2/3 - 2\delta$. Since the maximum distance travelled by a walk is, in absolute value, $20 \max_2 / \theta$, bad points of type 2 can only arise from walks which start within this distance of the edge (on either side). Hence, the probability that a walk starting in X has bad points of type 2 is at most

$$2 \frac{20 \max_2 / \theta - \delta}{\#X} \leq 2 \frac{20 \max_2}{\theta \#X}.$$

This gives the result. \square

Suppose $N_1 \approx N_2$ and $\theta = c/N_2$. Then δ is small compared with N_2 as long as m_2 is $o(\sqrt{N_2})$. Hence the denominator in equation (9) is essentially constant (which can be made arbitrarily large by taking c to be large). However, if m_2 grows at all with N_2 then so does \max_2 and the value in equation (9) can become large. Hence, in practice, it is necessary that m_2 and \max_2 be (rather small) constants. The random walk therefore covers a distance $O(\sqrt{N_2})$ in the P_2 component.

Again, even making the over-cautious assumption that all steps in walks which contain bad points of type 2 are wasted, by choosing θ to be sufficiently large one can ensure that only a very small proportion of walks can have bad points of type 2 (at least, this is true when $N_1 \approx N_2$). Hence one may assume that, say, 1% of the walks are wasted (and hence the algorithm runs in about 1.01 times the theoretical prediction of the time).

We now tabulate the complexity statements we have obtained. We give the total number of group operations, but note that the algorithm can be easily parallelised giving linear speedup in the number of processors. The factor $1 + \epsilon$ here is not the same as $1 + o(1)$: As mentioned above, there can be a non-negligible proportion of bad points of types 1 and 2 (even asymptotically as N tends to infinity). There is also the fact that one never expects a pseudorandom walk of the type considered in this paper to have exactly the behaviour of a random walk (and so there is a small correction factor to include in the birthday paradox). This latter issue is discussed at length by Teske [20]; for example her Table 3 suggests that if $n_S = 16$ then the expected number of trials before finding a collision is 1.01 times more than that predicted by the birthday paradox for a random map. Hence, the actual values for ϵ in practice might be between 0.02 and 0.04.

Conjecture 1. The following table gives the expected total number of group operations for the original and improved Gaudry-Schost algorithms to solve the 2-dimensional DLP in the average and worst cases. Here $\epsilon > 0$ denotes a small constant (not necessarily the same value in all places).

Name of Algorithm	Average Case	Worst Case
Original Gaudry-Schost	$2.45(1 + \epsilon)\sqrt{N} + \frac{1}{\theta}$	$3.58(1 + \epsilon)\sqrt{N} + \frac{1}{\theta}$
Improved Gaudry-Schost	$2.38(1 + \epsilon)\sqrt{N} + \frac{1}{\theta}$	$2.38(1 + \epsilon)\sqrt{N} + \frac{1}{\theta}$

5 Higher Dimensions

It is clear that the algorithm can be used to solve the multidimensional DLP in any dimension. The main task is to choose a suitable pseudorandom walk. Indeed, there is an important issue that occurs with θ as d increases.

Recall that the Gaudry-Schost algorithm expects to store approximately $\theta(2^d/3^{d/2})\sqrt{\pi N}$ distinguished points. Hence, it is usual to assume that $\theta = c/\sqrt{N}$ for some large constant c . If using a pseudorandom walk which moves only forwards in some component then the walk will cover a distance $O(1/\theta) = O(\sqrt{N})$ steps. When $d \geq 3$, if $N_1 \approx N_2 \approx N_3$, then this distance is longer than the sides of the orthotope in question. Hence, walks would go outside the region in which the tame-wild birthday paradox is being applied and the complexity of the algorithm would deteriorate. Even when $d = 2$, as we have seen, it is necessary to ensure that θ is sufficiently large to have an algorithm which performs well.

As a result, when $d \geq 3$ it is necessary to use pseudorandom walks with ‘side-to-side’ steps (i.e., with both positive and negative steps and with average zero) in every component. If the mean absolute step size is constant then, by Lemma 3, the expected distance travelled in any component after $O(N^{1/2})$ steps is $O(N^{1/4})$, which is OK when $d = 3$ and $N_1 \approx N_2 \approx N_3$. When $d = 4$ and $N_1 \approx N_2 \approx N_3 \approx N_4$ then the walks are still OK, as long as θ is sufficiently large (i.e., as long as one can store sufficiently many distinguished points).

However for $d \geq 5$ the issue of walks stepping outside the region of interest re-appears and it cannot be resolved by using ‘side-to-side’ walks. Pollard [16] has suggested a solution to this problem which we describe in the appendix.

6 Applications

The 2-dimensional DLP arises in algorithms for computing the number of points on genus 2 curves over finite fields [12]. One uses a Schoof-type algorithm to get information about the coefficients of the characteristic polynomial of the Frobenius modulo some integer, and then uses a baby-step-giant-step algorithm to complete the calculation. Gaudry and Schost [9] developed their low-memory algorithm precisely for this application. These ideas have also been used by Weng [23]. Our results will therefore give an improvement to algorithms of this type.

This approach can be used to count points on curves of any genus (though for curves of sufficiently large genus one might also exploit subexponential algorithms). Depending on the amount of information obtained from the Schoof part of the algorithm, the remaining computation could be a d -dimensional DLP with $d \geq 3$. Our methods would also give an improvement here.

The multidimensional discrete logarithm problem also arises explicitly in the work of Brands [2] and in Section 4 of Cramer, Gennaro and Schoenmakers [5]. The latter paper notes that the problem can be solved using a baby-step-giant-step algorithm but does not mention the possibility of a low-memory or parallelisable algorithm.

The Gallant, Lambert and Vanstone (GLV) method [8] speeds up elliptic curve arithmetic by rewriting $[n]P$ as $[n_1]P + [n_2]\psi(P)$ for some endomorphism ψ and

where $|n_1|, |n_2| \approx \sqrt{n}$. The integers n_1, n_2 are found by solving the closest vector problem in a lattice. An alternative is to choose “smaller” $n_1, n_2 \in \mathbb{Z}$ directly, rather than choosing a random integer n and rewriting it. Solving the DLP for points generated by the GLV method can be phrased as a multidimensional DLP. The methods of this paper imply that n_1 and n_2 cannot be chosen to be too small. See Galbraith and Scott [7] and Galbraith, Lin and Scott [6] for examples of the GLV method with $d > 2$.

Another approach to efficient elliptic curve cryptography is to use Koblitz curves [11] (i.e., ordinary elliptic curves E over \mathbb{F}_2 , considering the group $E(\mathbb{F}_{2^m})$). We rewrite $[n]P$ as

$$\sum_{i=0}^L n_i \tau^i(P)$$

where $n_i = \{-1, 0, 1\}$ and $\tau(P) = (x(P)^2, y(P)^2)$ is the 2-power Frobenius map. Since

$$\sum_{i=0}^L n_i \tau^i \equiv a + b\tau \text{ in } \mathbb{Z}[\tau]/(\tau^2 \pm \tau + 2)$$

where $|a| < 3\sqrt{2^L}$ and $|b| < 2\sqrt{2^L}$ as shown by Benits [1], solving the DLP can again be phrased as a multidimensional DLP i.e., $Q = [a]P + [b]\tau(P)$. It follows that L cannot be chosen to be too small. The same ideas can be applied on genus 2 curves over \mathbb{F}_2 leading to a 4-dimensional DLP.

7 Conclusion

We have presented an improvement to the algorithm given by Gaudry and Schost for solving the 2-dimensional DLP as well as extending the algorithm to the multidimensional DLP. We have also given further depth to the analysis given by Gaudry and Schost [9] specifically for the cases where $d > 2$. In Section 6 we have seen just a smattering of applications of this low-memory algorithm. An open problem is to investigate how best to exploit the algorithm when the search space is not an orthotope but a multidimensional ‘arrowhead’ which arises in the case of point counting on curves of genus 2 and higher.

References

1. Benits Jr., W.: Applications of Frobenius expansions in elliptic curve cryptography. PhD thesis, Royal Holloway, University of London (2008)
2. Brands, S.: An efficient off-line electronic cash system based on the representation problem, CWI Technical Report CS-R9323 (1993)
3. Cheon, J.H., Hong, J., Kim, M.: Speeding up the Pollard rho method on prime fields. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 471–488. Springer, Heidelberg (2008)
4. Cofman, E.G., Flajolet, P., Flatto, L., Hofri, M.: The maximum of a random walk and its application to rectangle packing. Technical report, INRIA (1997)

5. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) *EUROCRYPT 1997*. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
6. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. In: Joux, A. (ed.) *Eurocrypt 2009*. LNCS, vol. 5479, pp. 518–535. Springer, Heidelberg (2009)
7. Galbraith, S.D., Scott, M.: Exponentiation in pairing-friendly groups using homomorphisms. In: Galbraith, S.D., Paterson, K.G. (eds.) *Pairing 2008*. LNCS, vol. 5209, pp. 211–224. Springer, Heidelberg (2008)
8. Gallant, R., Lambert, R., Vanstone, S.: Improving the parallelized Pollard lambda search on binary anomalous curves. *Mathematics of Computation* 69, 1699–1705 (2000)
9. Gaudry, P., Schost, E.: A low-memory parallel version of Matsuo, Chao and Tsujii's algorithm. In: Buell, D.A. (ed.) *ANTS 2004*. LNCS, vol. 3076, pp. 208–222. Springer, Heidelberg (2004)
10. Kim, J.H., Montenegro, R., Peres, Y., Tetali, P.: A birthday paradox for Markov chains, with an optimal bound for collision in the Pollard rho algorithm for discrete logarithm. In: van der Poorten, A.J., Stein, A. (eds.) *ANTS-VIII 2008*. LNCS, vol. 5011, pp. 402–415. Springer, Heidelberg (2008)
11. Koblitz, N.: CM-curves with good cryptographic properties. In: Feigenbaum, J. (ed.) *CRYPTO 1991*. LNCS, vol. 576, pp. 279–287. Springer, Heidelberg (1992)
12. Matsuo, K., Chao, J., Tsujii, S.: An improved baby step giant step algorithm for point counting of hyperelliptic curves over finite fields. In: Fieker, C., Kohel, D.R. (eds.) *ANTS 2002*. LNCS, vol. 2369, pp. 461–474. Springer, Heidelberg (2002)
13. Nishimura, K., Sibuya, M.: Probability to meet in the middle. *Journal of Cryptology* 2, 13–22 (1990)
14. Pollard, J.M.: Monte Carlo methods for index computation mod p . *Mathematics of Computation* 32(143), 918–924 (1978)
15. Pollard, J.M.: Kangaroos, Monopoly and discrete logarithms. *Journal of Cryptology* 13, 437–447 (2000)
16. Pollard, J.M.: Remarks on discrete logs. Private Communication (August 2009)
17. Ruprai, R.S.: Improvements to the Gaudry-Schost algorithm for multidimensional discrete logarithm problems and applications. PhD Thesis, Royal Holloway University of London (2009)
18. Shanks, D.: Class number, a theory of factorization and genera. In: Proc. Symposium in Pure Mathematics, vol. 20, pp. 415–440 (1971)
19. Stinson, D.: *Cryptography: Theory and practice*, 3rd edn. Chapman & Hall/CRC (2006)
20. Teske, E.: On random walks for Pollard's rho method. *Mathematics of Computation* 70(234), 809–825 (2001)
21. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with application to hash functions and discrete logarithms. In: ACM Conference on Computer and Communications Security, pp. 210–218 (1994)
22. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. *Journal of Cryptology* 12, 1–28 (1999)
23. Weng, A.: A low-memory algorithm for point counting on picard curves. *Designs, Codes and Cryptography* 38(3), 383–393 (2006)

A Pollard's Method for Large Dimensions

As mentioned in Section 5, when $d \geq 5$ walks of $O(\sqrt{N})$ steps will usually move outside the tame and wild regions. This is a major problem for the Gaudry-Schost algorithm. Pollard [16] has suggested a way to deal with this problem, and we briefly sketch it here.

We describe a parallel algorithm for the case $d = 1$. Clearly the algorithm can be serial or parallel, and applies to all d .

Let G be the group in question and let N_1, \dots, N_d be the interval sizes in the d -dimensional DLP with $N_1 \leq N_2 \leq \dots \leq N_d$. Let $N = \prod_{i=1}^d (2N_i + 1)$ as usual. It is necessary to define two sets $D \subset S \subset G$, namely *distinguished points* and *special points*. As usual, the probability that a uniformly chosen $x \in G$ is distinguished should be $\theta = c/\sqrt{N}$ for some constant c . The probability p that a uniformly chosen $x \in G$ is special should be much higher, certainly $p > 1/N_1^2$. If this holds then one expects to find a special point after fewer than N_1^2 trials and a side-to-side random walk will cover distance $O(N_1)$ in that many steps and so still has a chance to be inside the region of interest.

We define a mapping $F : G \rightarrow G$ as follows. Let x be a point of G .

1. When x is a non-special point, as usual we have $F(x) = x + [r]P$ where r is small, and $[r]P$ is taken from a small table.
2. When x is a special point, we start a new walk from a new random point $[s]P$ or $Q + [s]P$ where s (and the choice tame/wild) are made deterministically and depending only on x (for example using a hash function). It is necessary that there are a large range of possible values for s .

The computation of $[s]P$ can be done by exponentiation or by multiplying elements from $k \geq 3$ tables of size $N^{1/k}$. The second method does not have constant storage, but we can make the storage as small as we wish.

3. When x is a distinguished point we store x together with: a link to the last distinguished point on this processor, and the distance (number of steps) between the current and last points.

When a distinguished point is repeated, two sequences have met at some point y . If we find y , we have two representations of the same group element as $[a]P$ or $Q + [b]P$. With probability $1/2$, we have one of each type and can solve for the discrete logarithm. Otherwise we continue until a tame-wild collision is found.

We can easily find y by a small storage process. We know the two preceding distinguished points x_1 and x_2 , on the two processors, and the distances travelled to the endpoint x . Suppose the distance from x_1 to x is longer than from x_2 to x . Advance the point x_1 to a point x'_1 of the same distance to x as x_2 . Now advance the points x'_1 and x_2 together until they meet at y .

The final part of the algorithm requires an expected $1/\theta = O(\sqrt{N})$ steps and cannot be parallelised.

On Designs and Multiplier Groups Constructed from Almost Perfect Nonlinear Functions

Yves Edel^{1,*} and Alexander Pott²

¹ Department of Pure Mathematics and Computer Algebra, Ghent University, Krijgslaan 281,
S22, B-9000 Ghent, Belgium

² Department of Mathematics, Otto-von-Guericke-University
Magdeburg, D-39016 Magdeburg, Germany

Abstract. Let $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be an almost perfect nonlinear function (APN). The set $D_f := \{(a, b) : f(x+a) - f(x) = b \text{ has two solutions}\}$ can be used to distinguish APN functions up to equivalence. We investigate the multiplier groups of these sets D_f . This extends earlier work done by the authors [1].

1 Introduction

The investigation of highly nonlinear functions is of interest in cryptography. We do not want to go into details about applications of highly nonlinear functions, but we refer to the literature, in particular [2,3].

There are several concepts of nonlinearity. Here we focus on differential nonlinearity. If $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is linear, then

$$f(x+a) - f(x) = b \quad (1)$$

has 0 or 2^n solutions (we have 2^n solutions if $b = f(a)$). In order to be “as nonlinear as possible”, the maximum number of solutions to (1) should be small for $a \neq 0$. We have $f(x+a) - f(x) = f((x+a)+x) - f(x+a)$ (note that the computations are done in a vector space over \mathbb{F}_2), hence if x is a solution of (1), then $x+a$ is a solution, too, so the number of solutions is always even. This motivates the following definition:

Definition 1. A function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is called **almost perfect nonlinear** or **APN** if the equations

$$f(x+a) - f(x) = b$$

have 0 or 2 solutions for all $a, b \in \mathbb{F}_2^n$, $a \neq 0$.

The main goal in the investigation of APN functions are *constructions*. There are by now many constructions known (they are summarized in [3]), hence it is necessary to find powerful methods how to *distinguish* APN functions. There are several papers which

* The research of the first author takes place within the project “Linear codes and cryptography” of the Fund for Scientific Research Flanders (FWO-Vlaanderen) (Project nr. G.0317.06), and is supported by the Interuniversity Attraction Poles Programme - Belgian State - Belgian Science Policy: project P6/26-Bcrypt.

survey some possible ways to distinguish functions up to equivalence ([1,4,5,6,7]). Here we will investigate the sets

$$D_f := \{(a, b) : f(x + a) - f(x) = b \text{ has two solutions}\}. \quad (2)$$

If f and g are equivalent (we will explain different concepts of equivalence in Section 2), then the sets D_f and D_g are, in a certain sense, equivalent. Hence inequivalence of the sets D_f and D_g implies inequivalence of the functions f and g . We will investigate, in particular, the *multiplier group*. Together with the so called *triple intersection numbers*, the sets D_f seem to be appropriate to distinguish equivalence classes of f . In Section 4, we summarize our computational results and pose some (what we think) interesting questions.

If f is an almost bent function (Definition 5), then the set D_f is a Hadamard difference set, equivalently its indicator function $\mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{C}$ is a bent function. Therefore, the ideas which we are going to develop in this note may be applied not just to the sets D_f constructed from APN functions but to arbitrary bent functions or Hadamard difference sets. We expect that it is possible to characterize certain highly symmetric bent functions by the automorphism or multiplier groups of the corresponding designs or difference sets. In this context, we refer to the interesting paper [8] which explains the orders of the multiplier groups $\mathcal{M}(D_f)$ for the Gold power mappings if n is odd. Moreover, it shows that the designs D_f , hence also the functions f , are not equivalent for different Gold power mappings (if n is odd), see also [7].

In this paper, we investigate different concepts of equivalence for different types of structures. Table 1 summarizes our notation.

Table 1. Different types of equivalence

type of equivalence	reference	remark
graph equivalence of f and g	Definition 2	$f, g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$
multiplier equivalence of A and B	Definition 3	$A, B \in \mathbb{C}[G]$
design equivalence of A and B	Definition 4	$A, B \in \mathbb{C}^{(n,n)}$ or $A, B \in \mathbb{C}[G]$

If $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, then graph equivalence is the same as multiplier equivalence for the sets G_f and G_g (Remark 3). Multiplier equivalence always implies design equivalence (5), but not vice versa (Example 1). We note that there is no concept of “multiplier equivalence” for arbitrary matrices $A \in \mathbb{C}^{(n,n)}$.

We will discuss these types of equivalence for the sets G_f and D_f related to APN functions f . Table 2 shows the *known* implications between these equivalences.

Table 2. Dependencies between equivalences of G_f and D_f

G_f multiplier equivalent to $G_g \Rightarrow G_f$ design equivalent to G_g
\Downarrow
D_f multiplier equivalent to $D_g \Rightarrow D_f$ design equivalent to D_g

This paper is partly motivated by the question whether the converse of any of these implications hold. To the best of our knowledge, there is no example of a pair of APN functions f and g which violates any of the possible converse implications in the diagram Table 1.

2 APN Functions

The most important definition for this paper is Definition 1. In order to investigate APN functions, group rings are an adequate algebraic tool.

Let \mathbb{K} be a field, and let G be a (multiplicatively written) abelian group. In this paper, \mathbb{K} will be almost exclusively the field \mathbb{C} of complex numbers, and the group will be in most cases an elementary abelian group whose order is a power of 2.

The set of formal sums

$$\sum_{g \in G} a_g \cdot g, \quad a_g \in \mathbb{K}$$

is called the **group algebra** $\mathbb{K}[G]$, where addition and multiplication on $\mathbb{K}[G]$ is defined as follows:

$$\left(\sum_{g \in G} a_g \cdot g \right) + \left(\sum_{g \in G} b_g \cdot g \right) := \sum_{g \in G} (a_g + b_g) \cdot g$$

and

$$\left(\sum_{g \in G} a_g \cdot g \right) \cdot \left(\sum_{g \in G} b_g \cdot g \right) := \sum_{g \in G} \left(\sum_{h \in G} a_h b_{gh^{-1}} \right) \cdot g.$$

Moreover,

$$\lambda \cdot \left(\sum_{g \in G} a_g \cdot g \right) := \sum_{g \in G} (\lambda a_g) \cdot g$$

for $\lambda \in \mathbb{K}$.

A subset A of G can be identified with the group algebra element $\sum_{g \in A} g$, which we denote (by abuse of notation) by A , again. If $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is a mapping, then its associated **graph** G_f is the set $\{(x, f(x)) : x \in \mathbb{F}_2^n\}$ which is a subset of $G = \mathbb{F}_2^n \times \mathbb{F}_2^n$. We denote the corresponding group algebra element in $\mathbb{C}[G]$ by G_f , too. In G , we denote the subgroup $\{(x, 0) : x \in \mathbb{F}_2^n\}$ by H , and the subgroup $\{(0, x) : x \in \mathbb{F}_2^n\}$ by N . The following proposition is obvious:

Proposition 1. *Let $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$. Then f is APN if and only if there is a subset D_f in $G = \mathbb{F}_2^n \times \mathbb{F}_2^n$ such that*

$$G_f^2 = 2^n + 2 \cdot D_f. \quad (3)$$

Remark 1. The set D_f contains no element of the form $(0, x)$, $x \in \mathbb{F}_2^n$, hence $D_f \cap N = \{\}$. Therefore, N is sometimes called a **forbidden subgroup**.

Proposition 1 shows that we may construct many more APN functions from a given one by applying affine transformations to G_f . Functions which can be constructed from f in this way are called *equivalent* to f . More precisely, we have

Definition 2. Two functions $f, g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ are called **graph equivalent** if there is an automorphism φ of $G = \mathbb{F}_2^n \times \mathbb{F}_2^n$ and an element $g \in G$ such that $\varphi(G_f) = G_g + g$. Here “addition plus g ” means that we add g to all elements of G_f , hence it is not “addition” in the group algebra.

Remark 2. If $g = (a, b)$, $a, b \in \mathbb{F}_2^n$, then $G_f + g$ is the graph of the mapping $x \mapsto f(x + a) + b$.

Definition 3. Let G be a multiplicatively written abelian group. We say that two group algebra elements A and B in $\mathbb{C}[G]$ are **multiplier equivalent** if there is a group automorphism φ of G such that $\varphi(A) = Bg$ for some $g \in G$. Note that we have to write $B \cdot g$ instead of “ $B + g$ ” since we write G multiplicatively. In Definition 2, the group was written additively.

Remark 3. The terminology “multiplier equivalence” is motivated by the investigation of difference sets, see [4] or [9], for instance. Note that graph equivalence for two functions f and g is the same as multiplier equivalence of G_f and G_g .

Remark 4. An automorphism φ of G does not fix, in general, the subgroups H and N setwise. If $\varphi(N) \neq N$, then $\varphi(G_f)$ is, in general, not the graph of a mapping $H \rightarrow N$. That makes the definition of graph equivalence seemingly less attractive since not all the elements in the orbit of G_f under group automorphisms are graphs of functions $H \rightarrow N$. We refer to [10] for a thorough discussion of graph equivalence (in that papers, the term *CCZ equivalence* was used, since CCZ equivalence was first introduced in a paper by Carlet, Charpin and Zinoviev [11]).

There is another nice way to interpret graph equivalence via code equivalence. We will introduce this concept in Section 3.

The group algebra over \mathbb{C} (or any algebra over an algebraically closed field) can be also described as a subalgebra of a matrix algebra: We label the rows and columns of a matrix with the elements of G . If $A = \sum_{g \in G} a_g g \in \mathbb{C}[G]$, then we define an embedding ι of $\mathbb{C}[G]$ into $\mathbb{C}^{|G| \times |G|}$ by $\iota(A) = (a_{g,h})_{g,h \in G}$ with $a_{g,h} = a_{g^{-1}h}$. It is easy to see and well known that ι is an injective homomorphism (actually independent from G being abelian or not). Equation (3) becomes

$$(\iota(G_F))^2 = 2^n + 2 \cdot \iota(D_F). \quad (4)$$

Since the group is elementary abelian, G_f is symmetric, and (4) shows that any two different rows of $\iota(G_f)$ have inner product 0 or 2. We may think of this property as the “defining” property of an APN mapping, and this property is preserved by row and column permutations. This gives rise to another concept of “equivalence”, which we call *design equivalence*:

Definition 4. Two matrices A and B in $\mathbb{C}^{(n,n)}$ are called **design equivalent** if there are permutation matrices P and Q such that $B = P \cdot A \cdot Q$. If G is a group of order n , then we call two group algebra elements A and B **design equivalent** if $\iota(A)$ and $\iota(B)$ have this property.

Remark 5. If $A, B \in \mathbb{C}[G]$ are multiplier equivalent, then $\iota(A)$ and $\iota(B)$ are design equivalent, but not vice versa, as the following example shows:

Example 1. We define the two sets

$$\begin{aligned} A &:= \{x \in \mathbb{F}_2^6 : x_1x_2 + x_3x_4 + x_5x_6 = 1\} \\ B &:= \{x \in \mathbb{F}_2^6 : x_1x_2 + x_3x_4 + x_5x_6 + x_1x_5x_3 = 1\} \end{aligned}$$

Using Magma [12] it is not very difficult to see that $\iota(A)$ and $\iota(B)$ are design equivalent. For this purpose, we define two *designs* using the matrices $\iota(A)$ and $\iota(B)$ as their incidence matrices: You may think of a design simply as a matrix with entries 0 and 1, where the columns correspond to points of the design, and the rows are the incidence vectors of blocks, see [9] for background from design theory. Magma checks quickly that the two designs corresponding to A and B are isomorphic which shows that there are permutation matrices P and Q such that $\iota(B) = P \cdot \iota(A) \cdot Q$. But the two sets are not multiplier equivalent since the function $f(x_1, \dots, x_6) = x_1x_2 + x_3x_4 + x_5x_6$ which defines A is quadratic, and the function $g(x_1, \dots, x_6) = x_1x_2 + x_3x_4 + x_5x_6 + x_1x_5x_3$ is of degree 3.

We note that the two functions f and g are *bent functions*, and the sets A and B are (Hadamard) difference sets, see Definition 6.

Two subsets associated with an APN function f are the graph G_f (which is basically the function) and D_f (which is a kind of derivative). In Section 3, we will investigate D_f in more detail. We note that design equivalence of the sets G_f and G_g (or graph equivalence of f and g) implies design equivalence of D_f and D_g . We state a more general result:

Proposition 2. *If $A, B \in \mathbb{C}^{(n,n)}$ are design equivalent, then $A^* \cdot A$ is design equivalent to $B^* \cdot B$, where “ $*$ ” denotes “complex conjugate transpose”.*

Proof. Write $B = PAQ$ for suitable permutation matrices P and Q . Then $Q^T A^* A Q = B^* B$. \square

Since we may also add a multiple of the identity matrix to design equivalent matrices and do not destroy equivalence in this way, we obtain the following corollary:

Corollary 1. *If G_f and G_g are design equivalent for APN functions f and g , then D_f and D_g are also design equivalent, since $\iota(D_f) = \iota(G_f)^* \iota(G_f) - 2^n I$.*

There is another concept, closely related to APN functions, called “almost bentness”. It is connected with the Walsh transform, which can be easily described in terms of group rings.

As before, let G be an abelian group of order v . There are v different homomorphisms $\chi : G \rightarrow \mathbb{K}^*$, provided that \mathbb{K} contains a v^* -th root of unity (v^* is the exponent of G , i.e. it is the least common multiple of the orders of the elements in G). In our cases, this condition is trivially satisfied since \mathbb{K} will be the field of complex numbers.

The homomorphisms are called **characters**. The set of characters form a group \widehat{G} : If χ_1 and χ_2 are two characters, then $\chi_1 \cdot \chi_2 : G \rightarrow \mathbb{K}^*$ is the character with $(\chi_1 \cdot \chi_2)(g) :=$

$\chi_1(g) \cdot \chi_2(g)$. The identity element in this **character group** is the so called trivial character or **principal character** $\chi_0 : G \rightarrow \mathbb{K}^*$ with $\chi_0(g) = 1$ for all $g \in G$. The group \widehat{G} is isomorphic to G .

If ψ is an automorphism of G , then the mapping χ^ψ defined by $\chi^\psi(g) := \chi(\psi(g))$ is a character, again.

We can extend characters (by linearity) to homomorphisms $\mathbb{K}[G] \rightarrow \mathbb{K}$: We define $\chi(\sum_{g \in G} a_g \cdot g) := \sum_{g \in G} a_g \cdot \chi(g)$. Note that these mappings are indeed homomorphisms, which means that they satisfy $\chi(A \cdot B) = \chi(A) \cdot \chi(B)$ and $\chi(A + B) = \chi(A) + \chi(B)$. The element

$$\sum_{\chi \in \widehat{G}} \chi(A) \cdot \chi \in \mathbb{K}[\widehat{G}]$$

is called the **Fourier transform** of $A \in \mathbb{K}[G]$. The following **orthogonality relations** are well known and easy to prove:

$$\begin{aligned} \sum_{g \in G} \chi(g) &= \begin{cases} 0 & \text{if } \chi \neq \chi_0, \\ |G| & \text{if } \chi = \chi_0, \end{cases} \\ \sum_{\chi \in \widehat{G}} \chi(g) &= \begin{cases} 0 & \text{if } g \neq 1, \\ |G| & \text{if } g = 1. \end{cases} \end{aligned}$$

Moreover,

$$a_g = \frac{1}{|G|} \sum_{\chi \in \widehat{G}} \chi(A) \cdot \chi(g^{-1}),$$

where $A = \sum_{g \in G} a_g g$. This last statement is called the **Fourier inversion formula**. In other words: If we know all the character values $\chi(A)$ of some group algebra element $A \in \mathbb{K}[G]$, then we know A .

The inversion formula implies what is called **Parseval's equation**:

$$\sum_{g \in G} a_g^2 = \frac{1}{|G|} \sum_{\chi \in \widehat{G}} |\chi(A)|^2.$$

Characters in elementary abelian 2-groups \mathbb{F}_2^m can be easily described. We take any nondegenerate symmetric bilinear form $(\cdot|\cdot)$. Then the mapping $\chi_u : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^m$ with $\chi_u(v) := (-1)^{(u|v)}$ is a character.

Quite often, \mathbb{F}_2^m is realized as the additive group of \mathbb{F}_2^m . In this case, we may take the trace-bilinear form $(u|v) := \text{tr}(u \cdot v)$, where $u, v \in \mathbb{F}_2^m$ and tr is the usual trace function $\text{tr}(x) = x + x^2 + x^4 + \dots + x^{2^{m-1}}$.

The multiset of character values of a group algebra element is called the **Walsh spectrum**. It is not invariant under equivalence since adding an element g to G_f gives multiplication of $\chi(G_f)$ by $\chi(g)$. The multiset of *absolute values* of the character values, which is called the **extended Walsh spectrum**, is invariant under graph equivalence. If f is APN, then Equation (3) gives the following connection between the Walsh coefficients of D_f and G_f :

$$\frac{\chi(G_f)^2 - 2^n}{2} = \chi(D_f).$$

Therefore, the extended Walsh spectrum of f uniquely determines the Walsh spectrum of D_f and vice versa.

If f is linear then $G_f^2 = 2^n G_f$, hence the *nonzero* character values have absolute value 2^n . There must be a nonzero character value $\chi(G_f)$ for some nontrivial character χ : Otherwise G_f would be a group algebra element with $\chi_0(G_f) = 2^n$ and $\chi(G_f) = 0$ for all other characters. Fourier inversion shows that this is possible only if $G_f = \frac{1}{2^n}G$, which is absurd since G_f has coefficients 0 and 1. Hence, another nonlinearity criteria is to minimize the maximum nontrivial Walsh coefficient (in absolute value) of G_f . One can show that there is at least one character χ with $\chi(G_f)^2 \geq 2^{n+1}$ (apply Parseval's equation to G_f^2 and note that all coefficients in G_f^2 are even).

Definition 5. A function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is **almost bent (AB)** if $|\chi(G_f)| \leq 2^{(n+1)/2}$ for all nontrivial characters χ .

Remark 6. If f is an AB function, then the nontrivial character values are 0 and $\pm 2^{(n+1)/2}$. Moreover, AB functions can exist only for n odd, see [3], for instance.

From the proof that the maximum Walsh coefficient is $2^{(n+1)/2}$, the following Proposition follows almost immediately:

Proposition 3. [13] Any AB function is APN.

Remark 7. The converse of this proposition is not true. First of all, APN functions do exist also if n is even, where no AB functions can exist. Moreover, there are also APN functions with n odd which are not AB, for instance the mapping x^{-1} .

We are mainly interested in APN functions rather than AB functions. The following Theorem is important and justifies that the concept of “design equivalence” is also useful if one investigates the Walsh coefficients of functions:

Theorem 1. Let A and B be elements in $\mathbb{C}[G]$, where G is an arbitrary abelian group. If A and B are design equivalent, then the extended Walsh spectrum of A and B are the same.

Proof (Compare with the proof of Proposition 2). It is well known that the vectors $(\chi(h))_{h \in G}$ are eigenvectors of $\iota(A)$ with eigenvalue $\chi(A)$: Note that

$$\sum_{h \in G} a_{g^{-1}h}\chi(h) = \sum_{h \in G} a_h\chi(gh) = \chi(A) \cdot \chi(g),$$

hence all the elements in the matrix algebra $\iota(\mathbb{C}[G])$ can be diagonalized simultaneously, since it is an algebra of commuting matrices.

Let $\iota(B) = P \cdot \iota(A) \cdot Q$ for some permutation matrices P and Q . The extended Walsh spectrum of A is the multiset of eigenvalues of $A^* \cdot A$, where $A^* = \sum_{g \in G} \overline{a_g} g^{-1}$ since $\chi(A^*) = \overline{\chi(A)}$, the complex conjugate of $\chi(A)$. It is not difficult to see that $\iota(A^*) = \iota(A)^*$, where $\iota(A)^*$ is the complex conjugate transpose matrix of $\iota(A)$. We have $\iota(B)^* \cdot \iota(B) = Q^T \cdot \iota(A)^* \cdot \iota(A) \cdot Q$, hence the multisets of eigenvalues of $\iota(B)^* \cdot \iota(B)$ and $\iota(A)^* \cdot \iota(A)$ coincide. \square

If f is AB, then $\chi(D_f) = 2^{n-1}$ if $\chi(G_f) = 2^{(n+1)/2}$ and $\chi(D_f) = -2^{n-1}$ if $\chi(G_f) = 0$. Therefore, D_f is a subset of \mathbb{F}_2^{2n} with $|\chi(D_f)|^2 = 2^{2n-2}$ for all non-trivial characters χ and $\chi_0(D_f) = 2^{2n-1} - 2^{n-1}$. Subsets with this property are called *Hadamard difference sets*. The indicator function of a Hadamard difference set D , i.e. the function $\text{ind}(x) = 1$ if $x \in D$ and $\text{ind}(x) = 0$ otherwise, is called a *bent function*. More precisely:

Definition 6. Let G be an abelian group of order $4u^2$. A subset D of G , $|D| = 2u^2 - u$, such that the list of differences $d - d'$ with $d, d' \in D$, $d \neq d'$, covers every nonidentity element of G exactly $u^2 - u$ times is called a **Hadamard difference set** of type $-$. The complement D' of D has the property that every element is covered exactly $u^2 + u$ times, and the order of D' is $2u^2 + u$ (Hadamard difference sets of type $+$).

Remark 8. For the general definition of difference sets and many references and examples and theoretical approaches, we refer to [9].

We summarize the discussion above in the following Proposition:

Proposition 4. [11] If $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is AB, then the set D_f is a Hadamard difference set of type $-$.

Many examples of Hadamard difference sets are known if G is an elementary abelian 2-group. Again, we refer to [2]. The most classical construction is the following:

Example 2. If $m = 2n$ is even, then the set

$$D := \{x \in \mathbb{F}_2^{2n} : x_1x_2 + x_3x_4 + \cdots + x_{2n-1}x_{2n} = 1\}$$

is a Hadamard difference set of type $-$.

It seems that only very few of the known Hadamard difference sets can be constructed as a set D_f for some AB function f . For instance, there are, up to affine equivalence, precisely 4 different Hadamard difference sets in \mathbb{F}_2^6 (see [14,15]), but only one of them occurs as D_f , since there is, up to affine equivalence, just one AB function $f : \mathbb{F}_2^3 \rightarrow \mathbb{F}_2^3$. The Hadamard difference set in this case is the classical quadratic example in 2. However, for larger n , the D_f 's are other bent functions. For quadratic functions, they all belong to the Maiorana-McFarland class, as we will see later. Here we just mention this important class of Hadamard difference sets:

Example 3 (Maiorana-McFarland construction, see [16]). Let H_1, \dots, H_{2^n-1} be the $2^n - 1$ different hyperplanes in \mathbb{F}_2^n . Let g_1, \dots, g_{2^n} be arbitrary elements in \mathbb{F}_2^n . Let v_1, \dots, v_{2^n-1} be different elements of \mathbb{F}_2^n . Then the set

$$\bigcup_{i=1}^{2^n-1} (v_i, H_i + g_i) \subset \mathbb{F}_2^n \times \mathbb{F}_2^n$$

is a Hadamard difference set of type $-$.

Remark 9. The Hadamard difference set in Example 2 is of Maiorana-McFarland type.

3 APN Functions and Their Groups

If $f, g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ are APN functions (or any functions), then we have called the two functions *graph equivalent* if G_f and G_g are *multiplier equivalent*. The group of affine transformations $v \mapsto \varphi(v) + g$ preserves the APN property, but, as explained in the last Section, may map G_f to some group algebra element which is not the graph of a function.

Let us describe several groups corresponding to a group algebra element $A \in \mathbb{C}[G]$ in general. Then we may apply the definitions both to G_f and D_f .

Definition 7. Let G be a multiplicatively written group G , and let $A \in \mathbb{C}[G]$. The **multiplier group** $\mathcal{M}(A)$ of A consists of all automorphisms φ of G such that $\varphi(A) = A \cdot g$ for some $g \in G$. The **automorphism group** of A consists of all affine transformations $\tau_{\varphi,g} : x \mapsto \varphi(x) \cdot g$ such that $\tau_{\varphi,g}(A) = A \cdot h$ for some $h \in G$. The **design automorphism group** $\text{Aut}(A)$ consists of all pairs of permutation matrices (P, Q) such that $P \cdot \iota(A) \cdot Q = \iota(A)$.

Remark 10. The automorphism group of A is contained in the design automorphism group of A , hence the **translations** $x \mapsto x \cdot g$ are design automorphisms.

Remark 11. The group generated by $\mathcal{M}(A)$ and the translations is the normalizer of the group of all translations $x \mapsto x \cdot g$, $g \in G$, in the design automorphism group.

From now on, G is always an elementary abelian group \mathbb{F}_2^m . We are going to describe how we can determine the multiplier group of A and how we can explain multiplier equivalence of two subsets $A, B \in G$ via code equivalence. We define an $(m+1) \times |A|$ -matrix A^{ext} over \mathbb{F}_2 as follows: The columns of the matrix are the vectors $(1, v)^T$ with $v \in A$. The row space of this matrix is called the *code* \mathcal{A} of A^{ext} . We define the analogous matrix for a subset B . If A and B are equivalent, then obviously $|A| = |B|$, and denote this number by a . The two codes \mathcal{A} and \mathcal{B} are called **code equivalent** if there is a permutation matrix P of size $a \times a$ and an invertible matrix U of size $m+1 \times m+1$ such that

$$U \cdot A^{\text{ext}} = A^{\text{ext}} \cdot P,$$

see [17], for instance. Since both the row space of A^{ext} and of B^{ext} contain the all-one-vector $(1, \dots, 1)$, we may assume without loss of generality that the first row of U is $(1, 0, \dots, 0)$. Thus, U is of type

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ v_1 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ v_m & * & \cdots & * \end{pmatrix},$$

i.e. there is an invertible matrix $W \in \mathbb{F}_2^{(m,m)}$ and $v \in \mathbb{F}_2^m$ such that

$$U = \begin{pmatrix} 1 & 0 \\ v & W \end{pmatrix}.$$

This shows that there is an automorphism φ (defined by the matrix W) of \mathbb{F}_2^m and an element $v \in \mathbb{F}_2^m$ such that $\varphi(A) + v = B$. We summarize this discussion in the following Theorem:

Theorem 2. *Two subsets A, B of \mathbb{F}_2^m are multiplier equivalent (see Definition 3) if and only if the codes defined by the rows of the extended matrices A^{ext} and B^{ext} are isomorphic.*

Given a permutation matrix P , the corresponding matrix U is, in general, not unique. However, if the rank of A^{ext} is $m + 1$, then U and hence W is uniquely determined by P . This shows the following:

Corollary 2. *Let A be a subset of \mathbb{F}_2^m , such that $m + 1$ is the \mathbb{F}_2 -rank of A^{ext} . Then the automorphism group of the code \mathcal{A} is isomorphic to the automorphism group of A .*

The condition in Corollary 2 is satisfied for the sets G_f and D_f corresponding to APN functions $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, $n > 2$:

Proposition 5. *Let $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be an APN function, $n > 2$. Then the \mathbb{F}_2 -rank of the matrices D_f^{ext} and G_f^{ext} is $2n + 1$.*

Proof. It is shown in [18], for instance, that the rank of G_f^{ext} is $2n + 1$. In other words, there is no vector $v \in \mathbb{F}_2^{2n}$ such that $(v|w) = 1$ for all $w \in G_f$ or $(v|w) = 0$ for all $w \in G_f$. The matrix D_f has the vectors $w + w'$, $w, w' \in G_f$, $w \neq w'$, as columns. If the rank of D_f^{ext} were smaller than $2n + 1$, there would be a vector v with $(v|w) + (v|w') = 1$ or $= 0$ for all $w, w' \in G_f$, $w \neq w'$. Obviously, it is impossible that $(v|w) + (v|w') = 1$ for all w, w' : We choose three different elements w_1, w_2 and w_3 in G_f . Then $(v|w_1) + (v|w_2) = 1$ and $(v|w_2) + (v|w_3) = 1$, hence $(v|w_1) + (v|w_3) = 0$. If $(v|w) + (v|w') = 0$ for all w, w' , we had $(v|w) = (v|w')$ for all $w, w' \in G_f$, which contradicts $\text{rank}(G_f^{\text{ext}}) = 2n + 1$, as indicated at the beginning of this proof. \square

Since it is rather easy (using Magma) to determine the automorphism groups of “small” codes (we can handle the codes associated with D_f up to $n = 8$), Magma provides us with a powerful tool to determine the automorphism and the multiplier groups of both sets G_f and D_f associated with APN functions. It seems to be harder to determine the design automorphism groups, see also [8].

4 Computational Results

It seems to be quite difficult to determine invariants like the automorphism groups of the sets D_f and G_f theoretically. Therefore, we do not include a table of all known infinite families of APN functions here, since we cannot prove any theoretical results about these series. We refer to [3] for the known families. Here we just mention that by now many infinite families of so called quadratic APN functions are known: We call f **quadratic** if the functions $x \mapsto f(x+a) + f(x) + f(a) + f(0)$ are linear.

Many APN’s for small values of n ($n \leq 12$) are constructed by computer, which are not yet members of infinite families of APN functions. With the exception of one

example in [1], all recently constructed functions are graph equivalent to a quadratic function.

Besides the many quadratic examples, we think that the classical *Kasami* family is one of the most interesting series:

Proposition 6 ([19,20], see also [21]). *Let $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be the power mapping x^d , where $d = 2^i + 1$ or $d = 2^{2i} - 2^i + 1$. If $\gcd(i, n) = 1$ (if n is odd) or $n/\gcd(i, n)$ is odd (if n is even), then f is APN. The cases $d = 2^i + 1$ are called the **Gold** cases, the cases $d = 2^{2i} - 2^i + 1$ the **Kasami** cases.*

Remark 12. The Gold power mappings are quadratic.

In the next tables, we determine the orders of the multiplier groups of the sets D_f , where f is one of the APN functions in [1] with $n \leq 8$.

Another question is whether the sets D_f or G_f are equivalent. It turns out that in all known examples with $n \leq 8$ so far, the sets D_f are not design equivalent if the functions are not graph equivalent. [In the case $n = 9$, we did not check so far that the designs D_f corresponding to x^3 , x^5 and x^{17} are not isomorphic (in that case, the triple intersection numbers, the ranks and the Walsh spectra are the same!).] This implies, in view of Corollary 1, that also the G_f are not design equivalent for functions which are not graph equivalent. So we think that one should try to find criteria to distinguish the “designs” corresponding to G_f and D_f theoretically. The situation with the sets D_f is similar: The sets D_f are “weaker”, i.e. when you compute D_f from G_f , you “lose” information about f . There seems to be no reason that a set A can occur as the set D_f for just one APN function f , but for the small examples in our tables, that is the situation.

Let us summarize this observation in the following proposition:

Proposition 7. *The sets D_f and therefore also the sets G_f are pairwise design inequivalent for the different APN functions listed in [1].*

Proof. The proof relies on computations done with Magma. We have checked some invariants which are easy to compute (Walsh spectrum, \mathbb{F}_2 -ranks, full (design) automorphism groups). However, this is not sufficient to distinguish all the sets D_f . In this case, we computed the so called *triple intersection numbers*: We may think of D_f as a $0-1$ matrix $\iota(D_f)$. Given three different rows of $\iota(D_f)$, we call the number of columns where all rows have entry 1 a *triple intersection number*.

The spectrum, i.e. the multiset of all these triple intersection numbers, is an invariant under design isomorphism. We used these numbers in order to distinguish the isomorphism type of the sets which could not be distinguished otherwise. \square

We think that it should be possible to determine some of the invariants which we discussed here (triple intersection numbers, automorphism groups) as well as some of the invariants discussed elsewhere (in particular the \mathbb{F}_2 -ranks of the incidence matrices $\iota(D_f)$ and $\iota(G_f)$) theoretically, in particular, if the functions f are quadratic. If f is quadratic, then the functions $x \mapsto f(x+a) - f(x)$ are (affinely) linear for all $a \in \mathbb{F}_2^n$. Hence the sets

$$H_a := \{f(x+a) - f(x) : x \in \mathbb{F}_2^n\}$$

are (affine) hyperplanes of codimension 1. The sets D_f are (for quadratic f)

$$D_f = \bigcup_{a \in \mathbb{F}_2^n, a \neq 0} (a, H_a).$$

If f is AB, these are precisely Maiorana-McFarland Hadamard difference sets.

APN functions f for which the sets H_a are always affine hyperplanes are called *crooked*, see [22]. There is an interesting conjecture about crooked functions, see [23,24] for partial results towards this conjecture.

Conjecture 1. A crooked functions must be quadratic.

As mentioned above, we know no example of a set D_f such that there exists a function g which is graph or design inequivalent to f and which satisfies $D_g = D_f$.

Question 1. Do the sets D_f determine f up to graph or design equivalence?

We note that the first author of this paper describes an interesting way how to reconstruct f from D_f if f is quadratic (under some additional assumption) [25].

Finally, we know of no example of an APN function f such that the “classical” example of a Hadamard difference set (Example 2) occurs as the set D_f (if $n > 3$).

Conjecture 2. Show that none of the bent functions D_f which occur from APN functions $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ are quadratic (Example 2).

4.1 Tables of APN Functions, $n \leq 8$

In Tables 5–7, we recall the list of APN functions constructed via “switching” in [1]. For the convenience of the reader, we use the same numbering as in [1], which is related to the switching process that has been used to construct the examples. We emphasize that this list of APN functions is complete only if $n = 5$ (see [26]): For $n \geq 6$, many

Table 3. Used primitive polynomials $p(x)$

n	$p(x)$
5	$x^5 + x^2 + 1$
6	$x^6 + x^4 + x^3 + x + 1$
7	$x^7 + x + 1$
8	$x^8 + x^4 + x^3 + x^2 + 1$

Table 4. All graph equivalence classes of APN’s in \mathbb{F}_2^5

$n = 5$	
No.	$F(x)$
1.1	x^3
1.2	x^5
2.1	x^{-1}

Table 5. Known graph equivalence classes of APN's in \mathbb{F}_2^6

$n = 6$	
No.	$F(x)$
1.1	x^3
1.2	$x^3 + u^{11}x^6 + ux^9$
2.1	$x^3 + ux^{24} + x^{10}$
2.2	(No. 2.1) + $u^3(\text{tr}(u^{10}x^3 + u^{53}x^5) + \text{tr}_{8/2}(u^{36}x^9))$
2.3	(No. 2.1) + $\text{tr}(u^{34}x^3 + u^{48}x^5) + \text{tr}_{8/2}(u^9x^9)$
2.4	(No. 2.1) + $u^2(\text{tr}(u^{24}x^3 + u^{28}x^5) + \text{tr}_{8/2}(x^9))$
2.5	(No. 2.3) + $u^{42}(\text{tr}(u^{10}x^3 + u^{51}x^5) + \text{tr}_{8/2}(u^9x^9))$
2.6	(No. 2.3) + $u^{23}(\text{tr}(u^{31}x^3 + u^{49}x^5) + \text{tr}_{8/2}(u^9x^9))$
2.7	(No. 2.3) + $u^{12}(\text{tr}(u^{42}x^3 + u^{13}x^5) + \text{tr}_{8/2}(u^{54}x^9))$
2.8	(No. 2.3) + $u(\text{tr}(u^{51}x^3 + u^{60}x^5) + \text{tr}_{8/2}(u^{18}x^9))$
2.9	(No. 2.3) + $u^{14}(\text{tr}(u^{18}x^3 + u^{61}x^5) + \text{tr}_{8/2}(u^{18}x^9))$
2.10	(No. 2.3) + $u^{17}(\text{tr}(u^{50}x^3 + u^{56}x^5))$
2.11	(No. 2.3) + $u^{19}(\text{tr}(u^{11}x^3 + u^7x^5 + u^{38}x^7 + u^{61}x^{11} + u^{23}x^{13}) + \text{tr}_{8/2}(u^{54}x^9) + \text{tr}_{4/2}(u^{42}x^{21}))$
2.12	(No. 2.4) + $u(\text{tr}(u^{54}x^3 + u^{47}x^5) + \text{tr}_{8/2}(u^9x^9))$

Table 6. Known graph equivalence classes of APN's in \mathbb{F}_2^7

$n = 7$	
No.	$F(x)$
1.1	x^3
1.2	$x^3 + \text{tr}(x^9)$
2.1	$x^{34} + x^{18} + x^5$
2.2	$x^3 + x^{17} + x^{33} + x^{34}$
3.1	x^5
4.1	x^9
5.1	x^{13}
6.1	x^{57}
7.1	x^{-1}
8.1	$x^{65} + x^{10} + x^3$
9.1	$x^3 + x^9 + x^{18} + x^{66}$
10.1	$x^3 + x^{12} + x^{17} + x^{33}$
10.2	$x^3 + x^{17} + x^{20} + x^{34} + x^{66}$
11.1	$x^3 + x^{20} + x^{34} + x^{66}$
12.1	$x^3 + x^{12} + x^{40} + x^{72}$
13.1	$x^3 + x^5 + x^{10} + x^{33} + x^{34}$
14.1	$x^3 + x^6 + x^{34} + x^{40} + x^{72}$
14.2	$x^3 + x^5 + x^6 + x^{12} + x^{33} + x^{34}$
14.3	(No. 14.1) + $u^{27}(\text{tr}(u^{20}x^3 + u^{94}x^5 + u^{66}x^9))$

more APN functions may exist. For $n \leq 4$, only one APN function exists (up to graph equivalence), which is x^3 . The examples listed here are graph inequivalent, and the sets G_f are pairwise design inequivalent. This follows from [1].

In the tables, u always denotes a primitive element in the respective field.

Table 7. Known graph equivalence classes of APN's in \mathbb{F}_2^8

$n = 8$	
No.	$F(x)$
1.1	x^3
1.2	$x^3 + \text{tr}(x^9)$
1.3	(No. 1.1) + $u(\text{tr}(u^{63}x^3 + u^{252}x^9))$
1.4	(No. 1.2) + $u^{38}(\text{tr}(u^{84}x^3 + u^{213}x^9))$
1.5	(No. 1.2) + $u^{51}(\text{tr}(u^{253}x^3 + u^{102}x^9))$
1.6	(No. 1.3) + $u^{154}(\text{tr}(u^{68}x^3 + u^{235}x^9))$
1.7	(No. 1.4) + $u^{69}(\text{tr}(u^{147}x^3 + u^{20}x^9))$
1.8	(No. 1.5) + $u^{68}(\text{tr}(u^{153}x^3 + u^{51}x^9))$
1.9	(No. 1.6) + $u^{35}(\text{tr}(u^{216}x^3 + u^{116}x^9))$
1.10	(No. 1.7) + $u^{22}(\text{tr}(u^{232}x^3 + u^{195}x^9))$
1.11	(No. 1.8) + $u^{85}(\text{tr}(u^{243}x^3 + u^{170}x^9)) \quad (\sim x^9 + \text{tr}(x^3))$
1.12	(No. 1.9) + $u^{103}(\text{tr}(u^{172}x^3 + u^{31}x^9))$
1.13	(No. 1.10) + $u^{90}(\text{tr}(u^{87}x^3 + u^{141}x^5 + u^{20}x^9) + \text{tr}_{16/2}(u^{51}x^{17} + u^{102}x^{34}))$
1.14	(No. 1.11) + $u^5(\text{tr}(u^{160}x^3 + u^{250}x^9))$
1.15	x^9
1.16	(No. 1.14) + $u^{64}(\text{tr}(u^{133}x^3 + u^{30}x^9))$
1.17	(No. 1.16) + $u^{78}(\text{tr}(u^{235}x^3 + u^{146}x^9))$
2.1	$x^3 + x^{17} + u^{16}(x^{18} + x^{33}) + u^{15}x^{48}$
3.1	$x^3 + u^{24}x^6 + u^{182}x^{132} + u^{67}x^{192}$
4.1	$x^3 + x^6 + x^{68} + x^{80} + x^{132} + x^{160}$
5.1	$x^3 + x^5 + x^{18} + x^{40} + x^{66}$
6.1	$x^3 + x^{12} + x^{40} + x^{66} + x^{130}$
7.1	x^{57}

Table 8. Orders of the groups of the sets G_f and D_f for $n = 5$

No.	$ \mathcal{M}(G_f) $	$\frac{ \text{Aut}(G_f) }{2^{2n} \mathcal{M}(G_f) }$	$ \mathcal{M}(D_f) $	$\frac{ \text{Aut}(D_f) }{2^{2n} \mathcal{M}(D_f) }$
1.1	$2^5 \cdot 5 \cdot 31$	1	1	1
1.2	$2^5 \cdot 5 \cdot 31$	1	32	1
2.1	$2 \cdot 5 \cdot 31$	1	1	1

4.2 Tables of Orders of Multiplier Groups

In Tables 9–11, we determine the multiplier groups and automorphism groups of G_f and D_f . This extends the tables in [1], where we did not determine the multiplier groups of D_f .

Moreover, we point out that in [1] there is a misprint in Table 10. The order of the group $\mathcal{M}(G_f)$ for $n = 8$, for the function f No. 1.2 is incorrect. The correct value is contained in Table 11.

Table 9. Orders of the groups of the sets G_f and D_f for $n = 6$

No.	$ \mathcal{M}(G_f) $	$\frac{ \text{Aut}(G_f) }{2^{2n} \mathcal{M}(G_f) }$	$ \mathcal{M}(D_f) $	$\frac{ \text{Aut}(D_f) }{2^{2n} \mathcal{M}(D_f) }$
1.1	$2^6 \cdot 6 \cdot 63$	1	1	2
1.2	$2^6 \cdot 63$	1	1	2
2.1	$2^7 \cdot 7$	1	1	1
2.2	2^6	1	1	1
2.3	2^6	1	1	1
2.4	2^6	1	1	1
2.5	$2^6 \cdot 5$	1	1	1
2.6	$2^6 \cdot 5$	1	1	1
2.7	2^6	1	1	1
2.8	2^6	1	1	1
2.9	2^6	1	1	1
2.10	2^6	1	1	1
2.11	2^3	1	1	1
2.12	$2^6 \cdot 7$	1	2	1

Table 10. Orders of the groups of the sets G_f and D_f for $n = 7$

No.	$ \mathcal{M}(G_f) $	$\frac{ \text{Aut}(G_f) }{2^{2n} \mathcal{M}(G_f) }$	$ \mathcal{M}(D_f) $	$\frac{ \text{Aut}(D_f) }{2^{2n} \mathcal{M}(D_f) }$
1.1	$2^7 \cdot 7 \cdot 127$	1	1	1
1.2	$2^7 \cdot 7$	1	1	1
2.1	$2^7 \cdot 7$	1	1	1
2.2	$2^7 \cdot 7$	1	1	1
3.1	$2^7 \cdot 7 \cdot 127$	1	1	1
4.1	$2^7 \cdot 7 \cdot 127$	1	2^7	1
5.1	$7 \cdot 127$	1	1	1
6.1	$7 \cdot 127$	1	1	1
7.1	$2 \cdot 7 \cdot 127$	1	1	1
8.1	$2^7 \cdot 7$	1	1	1
9.1	$2^7 \cdot 7$	1	1	1
10.1	$2^7 \cdot 7$	1	1	1
10.2	$2^7 \cdot 7$	1	1	1
11.1	$2^7 \cdot 7$	1	1	1
12.1	$2^7 \cdot 7$	1	1	1
13.1	$2^7 \cdot 7$	1	1	1
14.1	$2^7 \cdot 7$	1	1	1
14.2	$2^7 \cdot 7$	1	1	1
14.3	2^7	1	1	1

Table 11. Orders of the groups of the sets G_f and D_f for $n = 8$

No.	$ \mathcal{M}(G_f) $	$\frac{ \mathcal{M}(D_f) }{ \mathcal{M}(G_f) }$
1.1	$2^{11} \cdot 255$	1
1.2	$2^{11} \cdot 3$	1
1.3	$2^{10} \cdot 3$	1
1.4	$2^8 \cdot 3$	1
1.5	$2^{10} \cdot 3$	1
1.6	$2^{10} \cdot 3$	1
1.7	$2^9 \cdot 3$	1
1.8	$2^{10} \cdot 3$	1
1.9	$2^{10} \cdot 3$	1
1.10	$2^9 \cdot 3$	1
1.11	$2^{11} \cdot 3$	1
1.12	$2^{10} \cdot 3$	1
1.13	2^9	1
1.14	$2^8 \cdot 3$	1
1.15	$2^{11} \cdot 255$	16
1.16	$2^9 \cdot 3$	1
1.17	$2^9 \cdot 3$	1
2.1	$2^{10} \cdot 9 \cdot 5$	1
3.1	$2^{10} \cdot 3$	1
4.1	2^{11}	1
5.1	2^{11}	1
6.1	2^{11}	1
7.1	$2^3 \cdot 255$	1

5 Some Recent New Results on APN Functions

In this paper we have discussed the problem how to determine the isomorphism class of an APN function using the set D_f . We think that finding good invariants for the equivalence classes of designs is a challenging problem.

In this section, we would like to mention three very interesting recent results on APN functions which are not quite related to the topic of this paper, but which deserves to be mentioned.

5.1 Nonquadratic APN Functions

We have noted that the many new examples of APN functions which have been constructed in the last few years are all graph equivalent to quadratic functions. There is only one exception: A single example of a new nonquadratic APN function $\mathbb{F}_2^6 \rightarrow \mathbb{F}_2^6$ has been constructed in [1]. This function (Case 2.11 in our table 5) is also inequivalent to a power mapping. It is the only nonquadratic example on \mathbb{F}_2^6 which is known. It is not yet a member of an infinite family. The construction uses a “switching” of known quadratic APN functions (switching means “changing one coordinate function”, see [1]).

5.2 APN Permutations

Until recently, no APN permutation on \mathbb{F}_2^n with n even was known, and it was conjectured that none can exist. This conjecture was shattered recently by John F. Dillon and Adam Wolfe [27]. They constructed an APN permutation on \mathbb{F}_2^6 . The function is graph equivalent to Example 2.1 in Table 5, and it is so far the only one! There is a nice coding theoretic interpretation for the existence of bijective APN functions, see [27]. Using this interpretation, Dillon was able to show that for small n , none of the other known APN functions is equivalent to a permutation. This has been also confirmed by the first author of this paper: He checked that none of the APN functions in [1] is graph equivalent to a permutation, except the Dillon permutation.

It is now a challenging problem to find more APN permutations or to prove that no other example may exist.

5.3 Exceptional Exponents

There are several power mappings x^d which are APN functions. The *Gold* and the *Kasami* exponents (see example 6) are APN functions for infinitely many fields \mathbb{F}_2^n . An exponent d with the property that there are infinitely many fields where x^d is APN has been called an exceptional exponent. It has been conjectured (see [28]) that the *Gold* and *Kasami* exponents are the only exceptional exponents. This conjecture has now been proven: A major step towards a proof is contained in [29], and the missing cases are treated in [30].

6 Conclusions

We have determined the automorphism groups of the sets D_f where f is one of the APN function on \mathbb{F}_2^n , $n \leq 8$, described in [1]. Surprisingly, all the sets D_f are inequivalent.

If n is odd and f is almost bent, the sets D_f are Hadamard difference sets (or, equivalently, bent functions). None of these difference sets is equivalent to the classical quadratic Hadamard difference sets.

The results of this paper indicate that the sets D_f are apparently good “distinguishers” for APN functions. In the quadratic case, they are quite easy to describe and therefore they can be used (hopefully) for theoretical (computer free) proofs for the inequivalence of quadratic APN functions.

Most of the ideas in this paper can be also used to investigate arbitrary Hadamard difference sets or related objects (like partial difference sets) in elementary abelian groups.

References

1. Edel, Y., Pott, A.: A new almost perfect nonlinear function which is not quadratic. *Adv. Math. Commun.* 3, 59–81 (2009)
2. Carlet, C.: Boolean functions for cryptography and error correcting codes. In: Crama, Y., Hammer, P. (eds.) *Boolean Methods and Models*. Cambridge University Press, Cambridge (to appear), <http://www-roc.inria.fr/secret/Claude.Carlet/chap-fcts-Bool.pdf>

3. Carlet, C.: Vectorial boolean functions for cryptography. In: Crama, Y., Hammer, P. (eds.) Boolean Methods and Models. Cambridge University Press, Cambridge (to appear), <http://www-roc.inria.fr/secret/Claude.Carlet/chap-vectorial-fcts.pdf>
4. Edel, Y., Pott, A.: On the equivalence of nonlinear functions. In: Preneel, B., Dodunekov, S., Rijmen, V., Nikova, S. (eds.) Enhancing Cryptographic Primitives with Techniques from Coding Theory, NATO Advanced Research Workshop, pp. 87–103. IOS Press, Amsterdam (2009)
5. Göloğlu, F., Pott, A.: Almost perfect nonlinear functions: A possible geometric approach. In: Nikova, S., Preneel, B., Storne, L., Thas, J. (eds.) Coding Theory and Cryptography II, Koninklijke Vlaamse Academie van België voor Wetenschappen en Kunsten, pp. 75–100 (2007)
6. Edel, Y., Kyureghyan, G., Pott, A.: A new APN function which is not equivalent to a power mapping. *IEEE Trans. Inform. Theory* 52(2), 744–747 (2006)
7. Budaghyan, L., Carlet, C., Leander, G.: On inequivalence between known power APN functions. In: International Conference on Boolean Functions: Cryptography and Applications (2008) (to appear)
8. Dempwolff, U.: Automorphisms and equivalence of bent functions and of difference sets in elementary abelian 2-groups. *Comm. Algebra* 34(3), 1077–1131 (2006)
9. Beth, T., Jungnickel, D., Lenz, H.: Design Theory, 2nd edn. Cambridge University Press, Cambridge (1999)
10. Budaghyan, L., Carlet, C., Pott, A.: New classes of almost bent and almost perfect nonlinear polynomials. *IEEE Trans. Inform. Theory* 52(3), 1141–1152 (2006)
11. Carlet, C., Charpin, P., Zinoviev, V.: Codes, bent functions and permutations suitable for DES-like cryptosystems. *Des. Codes Cryptogr.* 15(2), 125–156 (1998)
12. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. i. the user language. *J. Symbolic Comput.* 24(3–4), 235–265 (1997)
13. Chabaud, F., Vaudenay, S.: Links between differential and linear cryptanalysis. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 356–365. Springer, Heidelberg (1995)
14. Preneel, B.: Analysis and Design of Cryptographic Hash Functions. PhD thesis, Katholieke Universiteit Leuven (1993)
15. Rothaus, O.S.: On “bent” functions. *J. Combinatorial Theory Ser. A* 20(3), 300–305 (1976)
16. McFarland, R.L.: Difference sets in abelian groups of order $4p^2$. *Mitt. Math. Sem. Giessen* (192), i–iv, 1–70 (1989)
17. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes. II, vol. 16. North-Holland Publishing Co., North-Holland Mathematical Library, Amsterdam (1977)
18. Browning, K., Dillon, J., Kibler, R., McQuistan, M.: APN polynomials and related codes (2008) (submitted)
19. Gold, R.: Maximal recursive sequences with 3-valued recursive cross-correlation function. *IEEE Trans. Inf. Th.* 14, 154–156 (1968)
20. Kasami, T.: The weight enumerators for several classes of subcodes of the 2nd order binary Reed-Muller codes. *Information and Control* 18, 369–394 (1971)
21. Dillon, J.F.: Multiplicative difference sets via additive characters. *Des., Codes, Cryptogr.* 17(1/2/3), 225–235 (1999)
22. Bending, T.D., Fon-Der-Flaass, D.: Crooked functions, bent functions, and distance regular graphs. *Electron. J. Combin.* 5, 14 (1998), Research Paper 34 (electronic)
23. Bierbrauer, J., Kyureghyan, G.M.: Crooked binomials. *Des. Codes Cryptogr.* 46(3), 269–301 (2008)
24. Kyureghyan, G.M.: Crooked maps in \mathbb{F}_{2^n} . *Finite Fields Appl.* 13(3), 713–726 (2007)
25. Edel, Y.: On quadratic APN functions and dimensional dual hyperovals, <http://www.mathi.uni-heidelberg.de/~yves/Papers/APNdho.html>

26. Brinkmann, M., Leander, G.: On the classification of APN functions up to dimension five. *Des., Codes, Cryptogr.* 1, 273–288 (2008)
27. Dillon, J.F.: slides from invited talk "APN Polynomials—An Update". In: The 9th International Conference on Finite Fields and their Applications, University College Dublin (2009), <http://mathsci.ucd.ie/~gmg/Fq9Talks/Dillon.pdf>
28. Janwa, H., Wilson, R.M.: Hyperplane sections of Fermat varieties in \mathbf{P}^3 in char. 2 and some applications to cyclic codes. In: Moreno, O., Cohen, G., Mora, T. (eds.) AAECC 1993. LNCS, vol. 673, pp. 180–194. Springer, Heidelberg (1993)
29. Jedlicka, D.: APN monomials over $GF(2^n)$ for infinitely many n . *Finite Fields Appl.* 13(4), 1006–1028 (2007)
30. Hernando, F., McGuire, G.: Proof of a conjecture on the sequence of exceptional numbers. classifying cyclic codes and APN functions arXiv:0903.2016v3 (2009)

A New Family of Hyper-Bent Boolean Functions in Polynomial Form

Sihem Mesnager

Department of Mathematics, University of Paris VIII and University of Paris XIII,

CNRS UMR 7539 LAGA (Laboratoire Analyse, Géometrie et Applications)

mesnager@math.jussieu.fr

Abstract. Bent functions are maximally nonlinear Boolean functions and exist only for functions with even number of inputs. These combinatorial objects, with fascinating properties, are rare. The class of bent functions contains a subclass of functions the so-called hyper-bent functions whose properties are still stronger and whose elements are still rarer. (Hyper)-bent functions are not classified. A complete classification of these functions is elusive and looks hopeless. So, it is important to design constructions in order to know as many of (hyper)-bent functions as possible. Few constructions of hyper-bent functions defined over the Galois field \mathbb{F}_{2^n} ($n = 2m$) are proposed in the literature. The known ones are mostly monomial functions.

This paper is devoted to the construction of hyper-bent functions. We exhibit an infinite class over \mathbb{F}_{2^n} ($n = 2m$, m odd) having the form $f(x) = Tr_1^{o(s_1)}(ax^{s_1}) + Tr_1^{o(s_2)}(bx^{s_2})$ where $o(s_i)$ denotes the cardinality of the cyclotomic class of 2 modulo $2^n - 1$ which contains s_i and whose coefficients a and b are, respectively in $\mathbb{F}_{2^{o(s_1)}}$ and $\mathbb{F}_{2^{o(s_2)}}$. We prove that the exponents $s_1 = 3(2^m - 1)$ and $s_2 = \frac{2^n - 1}{3}$, where $a \in \mathbb{F}_{2^n}$ ($a \neq 0$) and $b \in \mathbb{F}_4$ provide a construction of hyper-bent functions over \mathbb{F}_{2^n} with optimum algebraic degree. We give an explicit characterization of the bentness of these functions, in terms of the Kloosterman sums and the cubic sums involving only the coefficient a .

Keywords: Boolean function, Bent functions, Hyper-bent functions, Maximum nonlinearity, Walsh-Hadamard transformation, Kloosterman sums, Cubic sums.

1 Introduction

Bent functions, introduced by Rothaus [15] exist only for functions with even number of inputs $n = 2m$ and are those Boolean functions with Hamming distance $2^{n-1} \pm 2^{\frac{n}{2}-1}$ to all affine functions. Youssef and Gong [16] introduced and studied a subclass of the class of bent functions, that they called *hyper-bent* Boolean functions. Hyper-bent functions defined on the Galois field \mathbb{F}_{2^n} of order 2^n are those with Hamming distance to all functions of the form $Tr_1^n(ax^i) + \epsilon$ (where i is an integer co-prime with $2^n - 1$, $a \in \mathbb{F}_{2^n}$, $\epsilon \in \mathbb{F}_2$ and, where Tr_1^n denotes the absolute trace function from \mathbb{F}_{2^n} to \mathbb{F}_2) equals $2^{n-1} \pm 2^{\frac{n}{2}-1}$. Equivalently, they are defined with a property stronger than bentness, that is, functions

which are bent up to a change of primitive roots in the finite field \mathbb{F}_{2^n} . More precisely, f is hyper-bent if and only if the function $f(x^k)$ is bent for every k , co-prime with $2^n - 1$. In [16], Youssef and Gong have proved the existence of hyper-bent functions defined on \mathbb{F}_{2^n} , for every even n . Further, Carlet and Gaborit [1] have showed that the hyper-bent functions exhibited in [16] are those, up to a linear transformation, elements of the \mathcal{PS}_{ap} class due to Dillon [8,9]. Recall that, the class \mathcal{PS}_{ap} consists of all Boolean functions f defined on $\mathbb{F}_{2^n} \simeq \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, where $n = 2m$, whose expression are of the form $f(x, y) = g(xy^{2^m-2})$ for every $(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, where g is a balanced Boolean function on \mathbb{F}_{2^m} (that is of Hamming weight equal 2^{m-1}) which vanishes at 0.

However, the classification of hyper-bent functions remains open. In particular, as mentioned in [3], it seems difficult to define an infinite class of hyper-bent functions. In fact, since [16], very few infinite classes of hyper-bent Boolean functions have been presented in the literature. The known hyper-bent Boolean functions are mostly monomial functions that were already known to be bent (and that belong all to the \mathcal{PS}_{ap} class). Recently, Charpin and Gong [4,3] have provided new tools to describe hyper-bent Boolean functions with multiple trace terms by means of Dickson polynomials. They studied precisely Boolean functions defined on \mathbb{F}_{2^n} whose expression is of the form $\sum_{r \in E} Tr_1^n(\beta_r x^{r(2^m-1)})$, where E is a subset of the set of representatives of the cyclotomic cosets modulo $2^m + 1$ of maximal size $n = 2m$, and the coefficients β_r are in \mathbb{F}_{2^n} . As a consequence of their new approach, a characterization of a new class of binomial hyper-bent functions has been obtained. But the precise characterization of such functions which are hyper-bent, by giving explicitly the coefficients β_r , is still an open problem (see [3]). When the positive integer r is a co-prime with $2^m + 1$, the Boolean functions considered by Charpin and Gong are the sums of several Dillon monomial functions, that is, functions whose expression is precisely of the form $Tr_1^n(a_r x^{r(2^m-1)})$, with r is co-prime with $2^m + 1$ and which have been introduced by Dillon [8](1974). Dillon monomial hyper-bent functions are related to the zeros of some Kloosterman sums (see [8],[3],[12]).

In [14], we have exhibited a new infinite class of hyper-bent functions defined on \mathbb{F}_{2^n} whose expression is the sum of a Dillon monomial function and a trace function whose expression is of the from $Tr_1^k(ax^d)$ with $k < n$. More precisely of the form $Tr_1^n(ax^{(2^m-1)}) + Tr_1^2(bx^{\frac{2^n-1}{3}})$, where $m = \frac{n}{2}$ is odd, $a \in \mathbb{F}_{2^n}$ and $b \in \mathbb{F}_4$. We have shown that hyper-bent functions are related to the zeros of some Kloosterman sums minus 4. In [13], we extend this class of hyper-bent functions to the one whose functions are of the form $Tr_1^n(ax^{r(2^m-1)}) + Tr_1^2(bx^{\frac{2^n-1}{3}})$, where r is a positive integer co-prime with $2^m + 1$ and $m = \frac{n}{2}$ odd.

In this paper, we present another infinite class of Boolean functions that we denote by \mathfrak{G}_n , containing hyper-bent functions and whose expression is of the form:

$$\forall x \in \mathbb{F}_{2^n}, \quad Tr_1^n(ax^{3(2^m-1)}) + Tr_1^2(bx^{\frac{(2^n-1)}{3}}) \quad (1)$$

with $m = \frac{n}{2}$ odd and where $a \in \mathbb{F}_{2^n}^*$ and $b \in \mathbb{F}_4$. This infinite class is not contained in the class presented in [14,13] (since 3 is a divisor of $2^m + 1$ (m being

odd)) nor in the class studied by Charpin and Gong [3] that we have mentioned above.

The paper is organized as follows. In Section 2, we fix our main notation and recall the necessary background. Next, in Section 3, we investigate a precise characterization of such functions of \mathfrak{G}_n which are hyper-bent, by giving explicit conditions on the coefficients a and b . We firstly show that one can restrict oneself to study the bentness for some particular forms of functions belonging to \mathfrak{G}_n (Lemma 11). Afterwards, we show that \mathfrak{G}_n is a subclass of the well known *Partial Spreads* class for which the bentness of its functions can be characterized by means of the Hamming weight of their restrictions to a certain set (Lemma 13). We show in Proposition 18 that bent functions of the class \mathfrak{G}_n are also hyper-bent and more precisely, are (up to a linear transformation) elements of the \mathcal{PS}_{ap} class. We prove in Proposition 21 and Proposition 22 that, deciding whether an element of \mathfrak{G}_n is bent or not, depends strongly on the Kloosterman sums and also (in some cases) on the cubic sums involving only the coefficient a . Theorem 25 recapitulates the results of our study in which we prove that the class \mathfrak{G}_n contains hyper-bent functions when $m \not\equiv 3 \pmod{6}$ while, there is no hyper-bent functions in this class when $m \equiv 3 \pmod{6}$; an important point is that this class does not contain other bent functions except those which are hyper-bent. Finally, at the end of Section 3, we show that a bent function of the class \mathfrak{G}_n is normal and we compute its dual function.

2 Notation and Preliminaries

For any set E , $E^* = E \setminus \{0\}$ and $|E|$ will denote the cardinality of E .

2.1 Background on Boolean Functions

Let n be a positive integer. A Boolean function f on \mathbb{F}_{2^n} is an \mathbb{F}_2 -valued function on the Galois field \mathbb{F}_{2^n} of order 2^n . The *weight* of f , denoted by $\text{wt}(f)$, is the *Hamming weight* of the image vector of f , that is, the cardinality of its support $\{x \in \mathbb{F}_{2^n} \mid f(x) = 1\}$.

We denote the *absolute trace* over \mathbb{F}_2 of an element $x \in \mathbb{F}_{2^n}$ by $\text{Tr}_1^n(x) = \sum_{i=0}^{n-1} x^{2^i}$. The function Tr_1^n from \mathbb{F}_{2^n} to its prime field \mathbb{F}_2 is \mathbb{F}_2 -linear and satisfies $(\text{Tr}_1^n(x))^2 = \text{Tr}_1^n(x) = \text{Tr}_1^n(x^2)$ for every $x \in \mathbb{F}_{2^n}$. For any positive integer k , and r dividing k , the trace function from \mathbb{F}_{2^k} to \mathbb{F}_{2^r} , denoted by Tr_r^k , is the mapping defined as:

$$\forall x \in \mathbb{F}_{2^k}, \quad \text{Tr}_r^k(x) := \sum_{i=0}^{\frac{k}{r}-1} x^{2^{ir}} = x + x^{2^r} + x^{2^{2r}} + \cdots + x^{2^{k-r}}$$

Recall that, for every integer r dividing k , the trace function Tr_r^k satisfies the transitivity property, that is, $\text{Tr}_1^k = \text{Tr}_1^r \circ \text{Tr}_r^k$.

Every non-zero Boolean function f defined on \mathbb{F}_{2^n} has a (unique) trace expansion of the form:

$$\forall x \in \mathbb{F}_{2^n}, \quad f(x) = \sum_{j \in \Gamma_n} Tr_1^{o(j)}(a_j x^j) + \epsilon(1 + x^{2^n - 1}), \quad a_j \in \mathbb{F}_{2^{o(j)}} \quad (2)$$

called its polynomial form, where Γ_n is the set of integers obtained by choosing one element in each cyclotomic class of 2 modulo $2^n - 1$, the most usual choice being the smallest element in each cyclotomic class, called the coset leader of the class, and $o(j)$ is the size of the cyclotomic coset containing j , $\epsilon = \text{wt}(f)$ modulo 2. Recall that, given an integer e , $0 \leq e \leq 2^n - 1$, having the binary expansion: $e = \sum_{i=0}^{n-1} e_i 2^i$, $e_i \in \{0, 1\}$, the 2-weight of e , denoted by $w_2(e)$, is the Hamming weight of the binary vector $(e_0, e_1, \dots, e_{n-1})$.

The algebraic degree of f , denoted by $\deg(f)$, is equal to the maximum 2-weight of an exponent j for which $a_j \neq 0$ if $\epsilon = 0$ and to n if $\epsilon = 1$. Note that $\epsilon = 0$ when $\text{wt}(f)$ is even, that is, when the algebraic degree of f is less than n .

Let f be a Boolean function on \mathbb{F}_{2^n} . Its “sign” function is the integer-valued function $\chi_f(f) := (-1)^f$. The Walsh transform of f is the discrete Fourier transform of χ_f , whose value at $\omega \in \mathbb{F}_{2^n}$ is defined as follows:

$$\forall \omega \in \mathbb{F}_{2^n}, \quad \widehat{\chi_f}(\omega) = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{f(x) + Tr_1^n(\omega x)}.$$

The extended Hadamard transform of f is defined as follows:

$$\forall \omega \in \mathbb{F}_{2^n}, \quad \widehat{\chi_f}(\omega, k) = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{f(x) + Tr_1^n(\omega x^k)}, \text{ with } \gcd(k, 2^n - 1) = 1.$$

Bent (and hyper-bent) functions exist only for even n . They can be defined as follows:

Definition 1. A Boolean function $f : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2$ (n even) is said to be bent if its Walsh Hadamard transform takes only the values $\pm 2^{\frac{n}{2}}$, that is, $\widehat{\chi_f}(\omega) = \pm 2^{\frac{n}{2}}$, for all $\omega \in \mathbb{F}_{2^n}$.

Youssef and Gong proposed in [16] to strengthen the bent concept by using the extended Hadamard transform and stated the following.

Definition 2. A Boolean function $f : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_2$ (n even) is said to be hyper-bent if its extended Hadamard transform takes only the values $\pm 2^{\frac{n}{2}}$, that is, $\widehat{\chi_f}(\omega, k) = \pm 2^{\frac{n}{2}}$, for all $\omega \in \mathbb{F}_{2^n}$ and, for all k such that $\gcd(k, 2^n - 1) = 1$.

From now on, throughout the whole paper, we assume that $n = 2m$ ($m > 1$) is an even integer.

2.2 Some Additional Background

Let x be an element of \mathbb{F}_{2^n} . The conjugate of x over a subfield \mathbb{F}_{2^m} of \mathbb{F}_{2^n} will be denoted by $\bar{x} = x^{2^m}$ and the relative norm with respect to the quadratic field

extension $\mathbb{F}_{2^n}/\mathbb{F}_{2^m}$ by $norm(x) = x\bar{x}$. Also, we denote by U the set $\{u \in \mathbb{F}_{2^n} \mid norm(u) = 1\}$, which is the group of $(2^m + 1)$ -st roots of unity. Note that since the multiplicative group of the field \mathbb{F}_{2^n} is cyclic and $2^m + 1$ divides $2^n - 1$, the order of U is $2^m + 1$. Finally, note that the unit 1 is the single element in \mathbb{F}_{2^m} of norm one and every non-zero element x of \mathbb{F}_{2^n} has a unique decomposition as: $x = yu$ with $y \in \mathbb{F}_{2^m}^*$ and $u \in U$.

We also need two classical binary exponential sums on \mathbb{F}_{2^m} (where m is an arbitrary positive integer):

Definition 3. *The classical binary Kloosterman sums on \mathbb{F}_{2^m} are:*

$$K_m(a) := \sum_{x \in \mathbb{F}_{2^m}} \chi(Tr_1^m(ax + \frac{1}{x})), \quad a \in \mathbb{F}_{2^m}$$

The Kloosterman sums are generally defined on the multiplicative group $\mathbb{F}_{2^m}^*$ of \mathbb{F}_{2^m} . In this paper we extend to 0 assuming that $\chi(Tr_1^m(\frac{1}{x})) = 1$ for $x = 0$ (in fact, $Tr_1^m(\frac{1}{x}) = Tr_1^m(x^{2^{m-1}-1})$).

The following Proposition is directly obtained from the result of Lachaud and Wolfmann in [11] which is suitable for any m (even or odd).

Proposition 4. [11] *Let m be a positive integer. The set $\{K_m(a), a \in \mathbb{F}_{2^m}\}$, is the set of all the integers multiple of 4 in the range $[-2^{(m+2)/2} + 1, 2^{(m+2)/2} + 1]$.*

Divisibility properties of the Kloosterman sums have been studied in several recent papers. We recall , in particular, the following result given by Charpin, Helleseth and Zinoviev in [7] on the divisibility by 3 of $K_m(a) - 1$.

Proposition 5. [7] *Let $m \geq 3$ be an odd integer, and let $a \in \mathbb{F}_{2^m}^*$. Then*

$$K_m(a) - 1 \equiv 0 \pmod{3} \iff Tr_1^m(a^{1/3}) = 0$$

Definition 6. *The cubic sums on \mathbb{F}_{2^m} are:*

$$C_m(a, b) := \sum_{x \in \mathbb{F}_{2^m}} \chi(Tr_1^m(ax^3 + bx)), \quad a \in \mathbb{F}_{2^m}^*, b \in \mathbb{F}_{2^m}$$

The exact values of the cubic sums $C_m(a, a)$ on \mathbb{F}_{2^m} can be computed thanks to Carlitz's result [2] by means of the Jacobi symbol. Recall that the Jacobi symbol $(\frac{2}{m})$ is a generalization of the Legendre symbol (which is defined when m is an odd prime). For m odd, $(\frac{2}{m}) = (-1)^{\frac{(m^2-1)}{8}}$.

Proposition 7. [2] *Let m be an odd integer. Recall that the cubic sums on \mathbb{F}_{2^m} are the sums $C_m(a, c) := \sum_{x \in \mathbb{F}_{2^m}} \chi(Tr_1^m(ax^3 + cx))$ where $a \in \mathbb{F}_{2^m}^*$ and $c \in \mathbb{F}_{2^m}$. Then we have:*

1. $C_m(1, 1) = (\frac{2}{m}) 2^{(m+1)/2}$ where $(\frac{2}{m})$ is the Jacobi symbol.
2. If $Tr_1^m(c) = 0$, then $C_m(1, c) = 0$.
3. If $Tr_1^m(c) = 1$ (with $c \neq 1$), then $C_m(1, c) = \chi(Tr_1^m(\gamma^3 + \gamma)) (\frac{2}{m}) 2^{(m+1)/2}$ where $c = \gamma^4 + \gamma + 1$ for some $\gamma \in \mathbb{F}_{2^m}$.

Remark 8. Note that when $Tr_1^m(c) = 1$ and $c \neq 1$ then, the cubic sums $C_m(1, c)$ can be computed more precisely thanks to a recent result of Charpin et al. in [5]. More precisely : If $Tr_1^m(d) = 1$ (with $d \neq 1$) then $C_m(1, d) = (-1)^{Tr_1^m(\gamma^3)} \left(\frac{2}{m}\right) 2^{(m+1)/2}$ where γ is the unique element of \mathbb{F}_{2^m} satisfying $d = \gamma^4 + \gamma + 1$ and $Tr_1^m(\gamma) = 0$.

3 The Study of the Bentness of Elements of \mathfrak{G}_n

Let $n = 2m$ be an positive even integer. Let $a \in \mathbb{F}_{2^m}^*$ and $b \in \mathbb{F}_4^*$. Denote by \mathfrak{G}_n the set of the Boolean functions $g_{a,b}$ defined on \mathbb{F}_{2^n} whose polynomial form is given by this expression (note that $o(3(2^m - 1)) = n$ and $o(\frac{2^n - 1}{3}) = 2$):

$$\forall x \in \mathbb{F}_{2^n}, \quad g_{a,b}(x) = Tr_1^n\left(ax^{3(2^m - 1)}\right) + Tr_1^2\left(bx^{\frac{2^n - 1}{3}}\right). \quad (3)$$

Proposition 9. *The elements $g_{a,b}$ of \mathfrak{G}_n are all of algebraic degree m .*

Proof. Note that the 2-weights of $3(2^m - 1)$ and $\frac{2^n - 1}{3}$ are both equal to m (since $3(2^m - 1) = 1 + 2^2 + 2^3 + \dots + 2^{m-1} + 2^{m+1}$ and $\frac{2^n - 1}{3} = 1 + 4 + \dots + 4^{m-1}$). Thus, the two Boolean functions $x \mapsto Tr_1^n(ax^{3(2^m - 1)})$ and $x \mapsto Tr_1^2(bx^{\frac{2^n - 1}{3}})$ are of algebraic degree equal to m . The trace functions $Tr_1^n(ax^{3(2^m - 1)})$ and $Tr_1^2(bx^{\frac{2^n - 1}{3}})$ are two separate parts in the trace representation of $g_{a,b}$, the algebraic degree of $g_{a,b}$ is then equal to m .

It is well known that the algebraic degree of any bent Boolean function on \mathbb{F}_{2^n} is at most m (in the case that $n = 2$, the bent functions have degree 2). Bent functions of \mathfrak{G}_n are then of maximum algebraic degree.

From now, we assume that $n = 2m$ be an even integer with m odd.

Remark 10. Recall that an integer d is called a bent exponent if there exists $a \in \mathbb{F}_{2^n}^*$ for which the function $x \mapsto Tr_1^n(ax^d)$ is bent. Now, recall that if an integer d is a bent exponent then, either $\gcd(d, 2^m - 1) = 1$ or $\gcd(d, 2^m + 1) = 1$, where $m = n/2$ (see for instance [12]). Consequently, unlike the functions presented in [14,13], the monomial functions of the class \mathfrak{G}_n (case $b = 0$) are never bent since the exponent $d = 3(2^m - 1)$ is not co-prime with $2^m - 1$ nor with $2^m + 1$ (because when m is odd then 3 divides $2^m + 1$).

Now, recall that the set of n -variable bent Boolean functions is invariant under the action of the general affine group of \mathbb{F}_{2^n} and the addition of n -variable affine Boolean functions. In particular, if f and f' are two n -variable Boolean functions such that f' is linearly equivalent to f (that is, there exists an \mathbb{F}_2 -linear automorphism L of \mathbb{F}_{2^n} such that $f' = f \circ L$) then, f is bent if and only if f' is bent.

Now, let $a \in \mathbb{F}_{2^m}^*$, $\lambda \in \mathbb{F}_{2^n}^*$ and $b \in \mathbb{F}_4^*$. Set $a' = a\lambda^{3(2^m - 1)}$ and $b' = b\lambda^{\frac{2^n - 1}{3}}$. Then we remark that, for every $x \in \mathbb{F}_{2^n}$, we have:

$$g_{a',b'}(x) = Tr_1^n(a(\lambda x)^{3(2^m - 1)}) + Tr_1^2(b(\lambda x)^{\frac{2^n - 1}{3}}) = g_{a,b}(\lambda x) \quad (4)$$

This means that $g_{a',b'}$ is linearly equivalent to $g_{a,b}$. Consequently, we are not obliged to consider all the possible values of $a \in \mathbb{F}_{2^n}$ in our study of the bentness of an element of \mathfrak{G}_n . Indeed, recall that every element of x in $\mathbb{F}_{2^n}^*$ admits a unique polar decomposition $x = uy$ where $y \in \mathbb{F}_{2^m}^*$ and $u \in U := \{u \in \mathbb{F}_{2^n}^* \mid u^{2^m+1} = 1\}$. Now, m being odd, one can decompose U as follows

$$U = V \cup \zeta V \cup \zeta^2 V \quad (5)$$

where $V = \{u^3 \mid u \in U\}$ and $\zeta = \xi^{2^m-1}$ where ξ denotes a primitive element of the field \mathbb{F}_{2^n} . Thus, every element $u \in U$ can be uniquely decomposed as $u = \zeta^i v$ with $i \in \{0, 1, 2\}$ and $v \in V$. Therefore, one deduces straightforwardly from (4) the following Lemma.

Lemma 11. *Let $n = 2m$ with m odd. Let $a' \in \mathbb{F}_{2^n}^*$ and $b' \in \mathbb{F}_4^*$. Suppose that $a' = a\zeta^i v$ with $a \in \mathbb{F}_{2^m}^*$, $i \in \{0, 1, 2\}$, ζ be a generator of the cyclic group $U := \{u \in \mathbb{F}_{2^n}^* \mid u^{2^m+1} = 1\}$ and, $v \in V := \{u^3 \mid u \in U\}$. Then, there exists $b \in \mathbb{F}_4^*$ such that $g_{a',b'}$ is linearly equivalent to $g_{a\zeta^i,b}$.*

Every element $a' \in \mathbb{F}_{2^n}^*$ can be (uniquely) decomposed as $a' = a\zeta^i v$ with $a \in \mathbb{F}_{2^m}^*$, $i \in \{0, 1, 2\}$, ζ be a generator of the cyclic group $U := \{u \in \mathbb{F}_{2^n}^* \mid u^{2^m+1} = 1\}$ and, $v \in V := \{u^3 \mid u \in U\}$. Therefore, according to the preceding Lemma, one can restrict oneself to study the bentness of $g_{a\zeta^i,b}$ with $a \in \mathbb{F}_{2^m}^*$ $b \in \mathbb{F}_4^*$.

Now, recall the following well-known result [8] which includes the definition of the Partial Spreads class \mathcal{PS}^- introduced by Dillon.

Theorem 12. [8] *Let E_i , $i = 1, 2, \dots, N$, be N subspaces of \mathbb{F}_{2^n} of dimension m satisfying $E_i \cap E_j = \{0\}$ for all $i, j \in \{1, 2, \dots, N\}$ with $i \neq j$. Let f be a Boolean function over \mathbb{F}_{2^n} ($n = 2m$). Assume that the support of f can be written as*

$$\text{supp}(f) = \bigcup_{i=1}^N E_i^*, \quad \text{where } E_i^* := E_i \setminus \{0\}$$

Then f is bent if and only if $N = 2^{m-1}$. In this case f is said to be in the \mathcal{PS}^- class.

Lemma 13. *Let $a \in \mathbb{F}_{2^n}^*$ and $b \in \mathbb{F}_4^*$. Suppose that m is odd. Then, a function $g_{a,b}$ of the family \mathfrak{G}_n is bent if and only if $\Gamma(a,b) := \sum_{u \in U} \chi(g_{a,b}(u)) = 1$. Moreover, bent functions $g_{a,b}$ of the family \mathfrak{G}_n belong to the Partial Spreads class \mathcal{PS}^- .*

Proof. Recall that every element x of $\mathbb{F}_{2^n}^*$ has a unique decomposition as: $x = yu$, with $y \in \mathbb{F}_{2^m}^*$ and $u \in U := \{u \in \mathbb{F}_{2^n}^* \mid u^{2^m+1} = 1\}$. Then, since 3 divides $2^m + 1$ when m is odd, for every $x \in \mathbb{F}_{2^n}^*$, we have

$$g_{a,b}(x) = g_{a,b}(uy) = Tr_1^n \left(au^{3(2^m-1)} \right) + Tr_1^2 \left(bu^{\frac{2^m-1}{3}} \right) = g_{a,b}(u) \quad (6)$$

The function $g_{a,b}$ is then constant on the cosets $u\mathbb{F}_{2^m}^*$, $u \in U$. Therefore, the support of $g_{a,b}$ can be decomposed into the disjoint union sets (with the null vector, these sets are vector subspaces of dimension 2^m) as follows

$$\text{supp}(g_{a,b}) = \bigcup_{u \in S_{a,b}} u\mathbb{F}_{2^m}^* \text{ where } S_{a,b} := \{u \in U \mid g_{a,b}(u) = 1\}. \quad (7)$$

According to Theorem 12, this implies that the bentness of $g_{a,b}$ is equivalent to the fact that the Hamming weight of the restriction of $g_{a,b}$ to U is equal to 2^{m-1} and that bent functions $g_{a,b}$ of the class \mathfrak{G}_n are in the class \mathcal{PS}^- . To conclude, it suffices to note that $\Gamma(a, b) = |U| - 2 \text{wt}(g_{a,b}|_U)$ ($|U| = 2^m + 1$).

We have seen that, when m is odd, bent functions $g_{a,b}$, with $a \in \mathbb{F}_{2^n}^*$ and $b \in \mathbb{F}_4^*$ are in the class \mathcal{PS}^- class. In the following, we will give a more precise statement of Lemma 13, in particular, we will see that when m is odd then, bent Boolean functions of \mathfrak{G}_n are in the class of hyper-bent Boolean functions.

Let us recall the following result of Youssef and Gong given in [16].

Proposition 14. [16] Let $n = 2m$ be an even integer. Let α be a primitive element of \mathbb{F}_{2^n} . Let f be a Boolean function defined on \mathbb{F}_{2^n} such that $f(\alpha^{2^m+1}x) = f(x)$ for every $x \in \mathbb{F}_{2^n}$ and $f(0) = 0$. Then, f is hyper-bent function if and only if the weight of the vector $(f(1), f(\alpha), f(\alpha^2), \dots, f(\alpha^{2^m}))$ equals 2^{m-1} .

Charpin and Gong [3] have derived a slightly different version of the preceding Proposition.

Proposition 15. [3] Let $n = 2m$ be an even integer. Let α be a primitive element of \mathbb{F}_{2^n} . Let f be a Boolean function defined on \mathbb{F}_{2^n} such that $f(\alpha^{2^m+1}x) = f(x)$ for every $x \in \mathbb{F}_{2^n}$ and $f(0) = 0$. Denote by G the cyclic subgroup of $\mathbb{F}_{2^n}^*$ of order $2^m + 1$. Let ζ be a generator of G . Then, f is hyper-bent function if and only if the cardinality of the set $\{i \mid f(\zeta^i) = 1, 0 \leq i \leq 2^m\}$ equals 2^{m-1} .

Remark 16. It is important to point out that bent Boolean functions f defined on \mathbb{F}_{2^n} such that $f(\alpha^{2^m+1}x) = f(x)$ for every $x \in \mathbb{F}_{2^n}$ (where α be a primitive element of \mathbb{F}_{2^n}) and $f(0) = 0$ are hyper-bent (see proof of Proposition 15 in [3] or remark that the support $\text{supp}(f)$ of such Boolean functions f can be decomposed as $\text{supp}(f) = \bigcup_{i \in S} \alpha^i \mathbb{F}_{2^m}^*$, where $S = \{i \mid f(\alpha^i) = 1\}$, that is, thanks to Theorem 12, they are bent if and only if $|S| = 2^{m-1}$, proving that these bent functions are hyper-bent functions, according to Proposition 14).

Recall that, the class \mathcal{PS}_{ap} consists of all Boolean functions f defined on $\mathbb{F}_{2^n} \simeq \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, where $n = 2m$, whose expression are of the form $f(x, y) = g(xy^{2^m-2})$ for every $(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$, where g is a balanced Boolean function on \mathbb{F}_{2^m} (that is of Hamming weight equal 2^{m-1}) such that $g(0) = 0$. Carlet and Gaborit have proved in [1] the following more precise statement of Proposition 14.

Proposition 17. [1] Boolean functions of Proposition 14 such that $f(1) = 0$ are elements of the class \mathcal{PS}_{ap} . Those such that $f(1) = 1$ are the functions of the form $f(x) = g(\delta x)$ for some $g \in \mathcal{PS}_{ap}$ and $\delta \in \mathbb{F}_{2^n}^*$ such that $g(\delta) = 1$.

Thus, according to Proposition 15, Proposition 17 and Lemma 13, one can straightforwardly deduce a more precise statement of Lemma 13.

Proposition 18. *Let $a \in \mathbb{F}_{2^n}^*$ and $b \in \mathbb{F}_4^*$. Let $g_{a,b}$ be a Boolean function belonging to the family \mathfrak{G}_n , ($n = 2m$, m odd). Then $g_{a,b}$ is hyper-bent if and only if $\Gamma(a, b) := \sum_{u \in U} \chi(g_{a,b}(u)) = 1$ (where U denotes the set of the $(2^m + 1)$ -th roots of unity in \mathbb{F}_{2^n}). Moreover, $g_{a,b}$ is in the class \mathcal{PS}_{ap} if and only if $Tr_1^n(a) + Tr_1^2(b) = 0$.*

Proof. If α is a primitive element of \mathbb{F}_{2^n} then, $g_{a,b}(\alpha^{2^m+1}x) = g_{a,b}(x)$ for every $x \in \mathbb{F}_{2^n}$ (since 3 divides $2^m + 1$ when m is odd) and 0 is not in the support of $g_{a,b}$. The conditions of the bentness given by Proposition 15 are then satisfied thanks to Lemma 13. The second part of the Proposition is a direct application of Proposition 17.

According to Lemma 11 and Proposition 18, the question of deciding whether an element $g_{a,b}$ of \mathfrak{G}_n is hyper-bent or not can be reduced to computing the sum $\Gamma(a\zeta^i, \beta^j)$ for $(i, j) \in \{0, 1, 2\}^2$. For that, we need the following important technical results.

Lemma 19. *Let m be an odd integer. Let $a \in \mathbb{F}_{2^m}^*$. U denotes the set of the $(2^m + 1)$ -th roots of unity in \mathbb{F}_{2^n} . Then, we have*

$$\sum_{u \in U} \chi(Tr_1^n(au^3)) = 1 - K_m(a) + 2C_m(a, a).$$

Proof. Using the transitivity rule of trace function, we have

$$Tr_1^n(au^3) = Tr_1^m(Tr_m^n(au^3)) = Tr_1^m(au^3 + (au^3)^{2^m}).$$

Hence

$$\sum_{u \in U} \chi(Tr_1^n(au^3)) = \sum_{u \in U} \chi(Tr_1^m(a(u^3 + u^{-3}))).$$

Now, recall that every element $1/c$ where $c \in \mathbb{F}_{2^m}^*$ with $Tr_1^m(c) = 1$ can be uniquely represented as $u + u^{2^m}$ with $u \in U$. Therefore, since $1/c^3 + 1/c = u^3 + u^{-3}$ (indeed, $1/c^3 = (u + u^{2^m})^3 = (u + u^{-1})^3 = u^3 + u^{-3} + uu^{-1}(u + u^{-1}) = u^3 + u^{-3} + 1/c$), we have

$$\begin{aligned} \sum_{u \in U} \chi(Tr_1^n(au^3)) &= 1 + \sum_{u \in U \setminus \{1\}} \chi(Tr_1^m(a(u^3 + u^{-3}))) \\ &= 1 + 2 \sum_{\substack{c \in \mathbb{F}_{2^m} \\ Tr_1^m(c)=1}} \chi(Tr_1^m(a/c^3 + a/c)) \\ &= 1 + 2 \sum_{\substack{c \in \mathbb{F}_{2^m} \\ Tr_1^m(1/c)=1}} \chi(Tr_1^m(ac^3 + ac)) \end{aligned}$$

In the last equality, we use the fact that the map $c \mapsto 1/c$ is a permutation on \mathbb{F}_{2^m} . Now, Charpin, Helleseth and Zinoviev have proved in [6] that when m is odd, we have

$$2 \sum_{c \in \mathbb{F}_{2^m}, Tr_1^m(1/c)=1} \chi(Tr_1^m(ac^3 + ac)) = 2C_m(a, a) - K_m(a).$$

from which we deduce the result.

Now, recall that $V := \{u^3 \mid u \in U\}$. Let ζ be a generator of the cyclic group U . We introduce the sums

$$\forall a \in \mathbb{F}_{2^m}^*, \forall i \in \{0, 1, 2\}, \quad S_i(a) = \sum_{v \in V} \chi(Tr_1^n(a\zeta^i v)). \quad (8)$$

The sums $S_i(a)$ can be expressed in terms of Kloosterman sums and cubic sums. These expressions can be obtained from Proposition 9 in [14]. For completeness, we include the proof.

Lemma 20. *For every $a \in \mathbb{F}_{2^m}^*$, we have:*

$$S_0(a) = \frac{1 - K_m(a) + 2C_m(a, a)}{3}, \quad S_2(a) = S_1(a) = \frac{1 - K_m(a) - C_m(a, a)}{3}$$

Proof. Note firstly that the mapping $x \mapsto x^3$ being 3-to-1 on U , then, thanks to Lemma 19, one has

$$\sum_{v \in V} \chi(Tr_1^n(av)) = \frac{1}{3} \sum_{u \in U} \chi(Tr_1^n(au^3)) = \frac{1}{3}(1 - K_m(a) + 2C_m(a, a))$$

Now, since ζ^{2^m-2} is an element of V (because 3 divides (2^m+1)) and the mapping $v \mapsto \zeta^{2^m-2}v^{2^m}$ is a permutation on V , then, we have:

$$\begin{aligned} S_1(a) &= \sum_{v \in V} \chi(Tr_1^n(a\zeta v)) = \sum_{v \in V} \chi(Tr_1^n(a\zeta^{2^m} v^{2^m})) \\ &= \sum_{v \in V} \chi(Tr_1^n(a\zeta^2(\zeta^{2^m-2}v^{2^m}))) = S_2(a) \end{aligned}$$

Next, using the well-known result: $\sum_{u \in G} \chi(Tr_1^n(au)) = 1 - K_m(a)$, where G is a cyclic group of order $2^m + 1$ (different proofs can be found in [5,10,11,12]), we obtain :

$$S_0(a) + S_1(a) + S_2(a) = \sum_{u \in U} \chi(Tr_1^n(au)) = 1 - K_m(a)$$

Therefore, $S_1(a) = \frac{1 - K_m(a) - S_0(a)}{2}$. To conclude, it suffices to note that, the mapping $x \mapsto x^3$ being 3-to-1 from U to itself, one has $\sum_{u \in U} \chi(Tr_1^n(au^3)) = 3S_0(a)$ and that $\sum_{u \in U} \chi(Tr_1^n(au^3)) = 1 - K_m(a) + 2C_m(a, a)$ according to Lemma 19.

At this stage, we have the material to study of the (hyper)-bentness of Boolean functions belonging to the family \mathfrak{G}_n .

Proposition 21. *Let $n = 2m$ be an even integer with m odd. Let $a \in \mathbb{F}_{2^m}^*$, β be a primitive element of \mathbb{F}_4 and ζ be a generator of the cyclic group U of $(2^m + 1)$ -th of unity. Suppose that $\text{Tr}_1^n(a^{1/3}) = 0$. For $(i, j) \in \{0, 1, 2\}^2$, let $g_{a\zeta^i, \beta^j}$ be a Boolean function defined on \mathbb{F}_{2^n} whose expression is of the form (3). Suppose that $m \not\equiv 3 \pmod{6}$. Then, $g_{a\zeta^i, \beta^j}$ is bent if and only if $K_m(a) = 4$.*

Proof. Recall that,

$$\Gamma(a, b) \text{ denotes the sum } \sum_{u \in U} \chi(g_{a,b}(u)).$$

For $(i, j) \in \{0, 1, 2\}^2$, we have (using the fact that the mapping $u \mapsto u^{2^m-1}$ is a permutation of U)

$$\begin{aligned} \Gamma(a\zeta^i, \beta^j) &:= \sum_{u \in U} \chi(g_{a\zeta^i, \beta^j}(u)) = \sum_{u \in U} \chi\left(\text{Tr}_1^n(a\zeta^i u^{3(2^m-1)}) + \text{Tr}_1^2(\beta^j u^{\frac{2^m-1}{3}})\right) \\ &= \sum_{u \in U} \chi\left(\text{Tr}_1^n(a\zeta^i u^3) + \text{Tr}_1^2(\beta^j u^{\frac{2^m+1}{3}})\right) \end{aligned}$$

Now, thanks to (5), we have seen that every element $u \in U$ can be uniquely decomposed as $u = \zeta^l v$ with $l \in \{0, 1, 2\}$ and $v \in V := \{u^3 \mid u \in U\}$. Hence, for $(i, j) \in \{0, 1, 2\}^2$, we have (in the last equality, we use the fact that v is a cube of an element of U which is a group of order $2^m + 1$)

$$\begin{aligned} \Gamma(a\zeta^i, \beta^j) &= \sum_{l=0}^2 \sum_{v \in V} \chi\left(\text{Tr}_1^n(a\zeta^{3l+i} v^3) + \text{Tr}_1^2(\beta^j \zeta^{l\frac{2^m+1}{3}} v^{\frac{2^m+1}{3}})\right) \\ &= \sum_{l=0}^2 \sum_{v \in V} \chi\left(\text{Tr}_1^n(a\zeta^{3l+i} v^3) + \text{Tr}_1^2(\beta^j \zeta^{l\frac{2^m+1}{3}})\right) \end{aligned}$$

Next, $m \not\equiv 3 \pmod{6}$ then, integers 3 and $\frac{2^m+1}{3}$ are co-prime. The mapping $x \mapsto x^3$ is then a permutation of V and thus for $(i, j) \in \{0, 1, 2\}^2$, we have (in the last equality, we use the fact that the mapping $v \mapsto \zeta^{3l} v$ is a permutation of V)

$$\begin{aligned} \Gamma(a\zeta^i, \beta^j) &= \sum_{l=0}^2 \sum_{v \in V} \chi(\text{Tr}_1^n(a\zeta^{3l+i} v) + \text{Tr}_1^2(\beta^j \zeta^{l\frac{2^m+1}{3}})) \\ &= \sum_{l=0}^2 \sum_{v \in V} \chi(\text{Tr}_1^n(a\zeta^i v) + \text{Tr}_1^2(\beta^j \zeta^{l\frac{2^m+1}{3}})) \end{aligned}$$

But, for every $j \in \{0, 1, 2\}$, the set $\{\beta^j, \beta^j \zeta^{\frac{2^m+1}{3}}, \beta^j \zeta^{2\frac{2^m+1}{3}}\}$ is equal to \mathbb{F}_4^* (which contains two elements of absolute trace 1 on \mathbb{F}_4 and one element of absolute trace 0 on \mathbb{F}_4). We thus conclude that

$$\Gamma(a\zeta^i, \beta^j) = - \sum_{v \in V} \chi(\text{Tr}_1^n(a\zeta^i v)) =: -S_i(a). \quad (9)$$

Next, since m is odd, the mapping $x \mapsto x^3$ is permutation on \mathbb{F}_{2^m} . Hence, every element $a \in \mathbb{F}_{2^m}$ can be (uniquely) written as $a = c^3$ with $c \in \mathbb{F}_{2^m}$. One has

$$\begin{aligned} C_m(a, a) &:= \sum_{x \in \mathbb{F}_{2^m}} \chi(Tr_1^m(ax^3 + ax)) = \sum_{x \in \mathbb{F}_{2^m}} \chi(Tr_1^m((cx)^3 + ax)) \\ &= \sum_{x \in \mathbb{F}_{2^m}} \chi(Tr_1^m((cx)^3 + a^{2/3}(cx))) = \sum_{x \in \mathbb{F}_{2^m}} \chi(Tr_1^m(x^3 + a^{2/3}x)) \\ &= C_m(1, a^{2/3}). \end{aligned}$$

Now, since $Tr_1^m(a^{2/3}) = Tr_1^m(a^{1/3})$ and $Tr_1^m(a^{1/3}) = 0$ (by hypothesis), one has $C_m(a, a) = 0$, according to Proposition 7. Therefore, thanks to Lemma 20, we obtain

$$\Gamma(a\zeta^i, \beta^j) = \frac{K_m(a) - 1}{3}.$$

We conclude thanks to Lemma 13.

Proposition 22. *Let $n = 2m$ be an even integer with m odd. Let $a \in \mathbb{F}_{2^m}^*$, β be a primitive element of \mathbb{F}_4 and, ζ be a generator of the cyclic group U of $(2^m + 1)$ -th of unity. Suppose that $Tr_1^m(a^{1/3}) = 1$. For $(i, j) \in \{0, 1, 2\}^2$, let $g_{a\zeta^i, \beta^j}$ be a Boolean function on \mathbb{F}_{2^n} whose expression is of the form (3). Suppose that $m \not\equiv 3 \pmod{6}$. Then*

1. *The function g_{a, β^j} is not bent for every $j \in \{0, 1, 2\}$.*
2. *For every $i \in \{1, 2\}$ and $j \in \{0, 1, 2\}$, the function $g_{a\zeta^i, \beta^j}$ is bent if and only if $K_m(a) + C_m(a, a) = 4$.*

Proof. We have seen in the proof of Proposition 21, that $C_m(a, a) = C_m(1, a^{2/3})$. Then, according to Proposition 7, one has $C_m(a, a) = \epsilon_a \left(\frac{2}{m}\right) 2^{(m+1)/2}$ with $\epsilon_a = \pm 1$ (since $Tr_1^m(a^{2/3}) = Tr_1^m(a^{1/3})$ and $Tr_1^m(a^{1/3}) = 1$, by hypothesis).

1. Let $j \in \{0, 1, 2\}$. According to (9), valid only if $m \not\equiv 3 \pmod{6}$, and thanks to Lemma 20, we have that $\Gamma(a, \beta^j) = \frac{K_m(a) - 1 - \epsilon_a \left(\frac{2}{m}\right) 2^{(m+3)/2}}{3}$. Then, according to Lemma 13, the Boolean function g_{a, β^j} is therefore bent if and only if $K_m(a) = 4 \pm \left(\frac{2}{m}\right) 2^{(m+3)/2}$, which is impossible for $m > 3$, since the Kloosterman sums $K_m(a)$ take values in the range $[-2^{(m+2)/2} + 1, 2^{(m+2)/2} + 1]$, according to Proposition 4.
2. According to (9) and Lemma 20, for every $i \in \{1, 2\}$ and $j \in \{0, 1, 2\}$, we have $\Gamma(a\zeta^i, \beta^j) = \frac{K_m(a) + C_m(a, a) - 1}{3}$. The Boolean function $g_{a\zeta^i, \beta^j}$ is therefore bent if and only if $K_m(a) + C_m(a, a) = 4$, according to Lemma 13.

Remark 23. Since the cubic sums $C_m(a, a)$ equal $\epsilon_a \left(\frac{2}{m}\right) 2^{(m+1)/2}$ with $\epsilon_a = \pm 1$ (when $Tr_1^m(a^{1/3}) = 1$, m odd) and the Jacobi symbol $\left(\frac{2}{m}\right)$ equals $(-1)^{\frac{(m^2-1)}{8}}$ (when m is odd) then, the condition $K_m(a) + C_m(a, a) = 4$ on $a \in \mathbb{F}_{2^m}^*$ says that the Kloosterman sums $K_m(a)$ take the values $4 \pm 2^{(m+1)/2}$.

Proposition 24. Let $n = 2m$. Suppose that m is odd such that $m \equiv 3 \pmod{6}$. Let $a \in \mathbb{F}_{2^m}^*$, $b \in \mathbb{F}_4$ and, ζ be a generator of the cyclic group U of $(2^m + 1)$ -th of unity. For $i \in \{0, 1, 2\}$, let $g_{a\zeta^i, b}$ be a Boolean function on \mathbb{F}_{2^n} whose expression is of the form (3). Then, $g_{a\zeta^i, b}$ is not bent.

Proof. According to Lemma 11 and Lemma 13, it suffices to compute the value $\sum_{u \in U} \chi(g_{a\zeta^i, b}(u))$ to decide whether $g_{a\zeta^i, b}$ is bent or not. Note now that (recall that 9 divides $2^m + 1$ if $m \equiv 3 \pmod{6}$)

$$\sum_{u \in U} \chi(g_{a\zeta^i, b}(u)) = \sum_{u \in U} \chi(Tr_1^n(a\zeta^i u^{3(2^m-1)}) + Tr_1^2(bu^{3(2^m-1) \cdot \frac{2^m+1}{9}}))$$

The mapping $x \mapsto x^{3(2^m-1)}$ is 3-to-1 from U to itself. Thus, we get that

$$\sum_{u \in U} \chi(g_{a\zeta^i, b}(u)) = 3 \sum_{v \in V} \chi(Tr_1^n(a\zeta^i v) + Tr_1^2(bv^{\frac{2^m+1}{9}}))$$

where $V = \{u^3 \mid u \in U\}$. The sum $\sum_{u \in U} \chi(g_{a\zeta^i, b}(u))$ is therefore a multiple of 3 and cannot be equal to 1 implying that $g_{a\zeta^i, b}$ cannot be bent.

Collecting the results obtained in Proposition 21, Proposition 22 and Proposition 24 we obtain the following characterization of the bentness for Boolean function of the form (3).

Theorem 25. Let $n = 2m$. Suppose that m is odd. Let $a \in \mathbb{F}_{2^m}^*$. Let β be a primitive element of \mathbb{F}_4 . Let ζ be a generator of the cyclic group U of $(2^m + 1)$ -th of unity. For $(i, j) \in \{0, 1, 2\}^2$, let $g_{a\zeta^i, \beta^j}$ be a Boolean function on \mathbb{F}_{2^n} whose expression is of the form (3).

1. Assume $m \not\equiv 3 \pmod{6}$. Then, we have:

- If $Tr_1^m(a^{1/3}) = 0$ then, for every $(i, j) \in \{0, 1, 2\}^2$, a function $g_{a\zeta^i, \beta^j}$ is (hyper)-bent if and only if $K_m(a) = 4$.

– If $Tr_1^m(a^{1/3}) = 1$ then:

- (a) g_{a, β^j} is not bent for every $j \in \{0, 1, 2\}$.
- (b) For every $i \in \{1, 2\}$, $g_{a\zeta^i, \beta^j}$ is (hyper)-bent if and only if $K_m(a) + C_m(a, a) = 4$.

2. Assume $m \equiv 3 \pmod{6}$. Then, for every $i \in \{0, 1, 2\}$, $g_{a\zeta^i, b}$ is not bent for every $a \in \mathbb{F}_{2^m}^*$ and $b \in \mathbb{F}_4^*$.

Example 26. Let us describe for example the set of bent Boolean functions $g_{a, b}$ belonging to the class \mathfrak{G}_{10} (with $b \neq 0$), that is, of the form $Tr_1^{10}(ax^{93}) + Tr_1^2(bx^{341})$ where $a \in \mathbb{F}_{2^{10}}^*$ and $b \in \mathbb{F}_4^*$.

Let α be a primitive element of $\mathbb{F}_{32} = \mathbb{F}_2(\alpha)$ with $\alpha^5 + \alpha^2 + 1 = 0$. Let ξ be a primitive element of $\mathbb{F}_{2^{10}}$. According to table 4 in [5], the set $\{a \in \mathbb{F}_{25}^*, Tr_1^5(a^{1/3}) = 0\}$ is equal to $\{\alpha^3, \alpha^{21}, \alpha^{14}\}$ and, the set $\{a \in \mathbb{F}_{25}^*, Tr_1^5(a^{1/3}) = 1\}$ is equal to $\{1, \alpha^2, \alpha^9, \alpha^{15}\}$. The elements a of \mathbb{F}_{25}^* whose the Kloosterman sums $K_5(a)$ on \mathbb{F}_{25} equals 4 (those elements a satisfy necessary $Tr_1^5(a^{1/3}) = 0$)

are α^3 and α^{21} while, those such that $K_5(a) + C_5(a, a) = 4$ are 1 and α^9 (more precisely, we have $K_5(1) = 12$ and $K_5(\alpha^9) = -4$).

According to Theorem 25 and Lemma 11 we conclude that there exist 330 hyper-bent Boolean functions defined on the field $\mathbb{F}_{2^{10}}$ belonging to the class \mathfrak{G}_{10} (with $b \neq 0$). Such functions are $g_{\alpha^3 v, b}$, $g_{\alpha^{21} v, b}$, $g_{\xi^{31} v, b}$, $g_{\alpha^3 \xi^{31} v, b}$, $g_{\alpha^9 \xi^{31} v, b}$, $g_{\alpha^{21} \xi^{31} v, b}$, $g_{\xi^{62} v, b}$, $g_{\alpha^3 \xi^{62} v, b}$, $g_{\alpha^9 \xi^{62} v, b}$, $g_{\alpha^{21} \xi^{62} v, b}$, with $b \in \mathbb{F}_4^*$ and v runs through the set $\{u^3 \mid u \in U\}$ where U is the cyclic group of 33-rd root of unity of $\mathbb{F}_{2^{10}}$.

Example 27. Let $n = 14$ then, according to table 4 in [5], we find that there exist 1935 hyper-bent Boolean functions $g_{a,b}$ (with $b \neq 0$) defined on the field \mathbb{F}_{16384} belonging to the class \mathfrak{G}_{14} . Such functions are of the form

- $Tr_1^{14}(cvx^{381}) + Tr_1^2(bx^{5461})$, $c \in \{\alpha^{14}, \alpha^{15}, \alpha^{62}\}$,
- $Tr_1^{14}(c'\xi^{127i}vx^{381}) + Tr_1^2(bx^{5461})$, $i \in \{1, 2\}$, $c' \in \{1, \alpha^{14}, \alpha^{15}, \alpha^{21}, \alpha^{62}, \alpha^{93}\}$,

where α is a primitive element of \mathbb{F}_{128} satisfying $\alpha^7 + \alpha^3 + 1 = 0$, ξ is a primitive element of $\mathbb{F}_{2^{14}}$, v runs through the set $\{u^3 \mid u \in U\}$ where U is the cyclic group of 129-th root of unity of $\mathbb{F}_{2^{14}}$ and $b \in \{1, \beta, \beta^2\}$ where β is a primitive element of \mathbb{F}_4 .

Example 28. Let $n = 18$ then, according to Theorem 25, there exist no bent Boolean functions in the class \mathfrak{G}_{18} .

Now recall that bent Boolean functions always occur in pairs. In fact, given a bent function f on \mathbb{F}_{2^n} , we define the *dual Boolean function* \tilde{f} of f by considering the signs of the values $\widehat{\chi_f}(a)$, $a \in \mathbb{F}_2^n$ of the Walsh transform of f as follows:

$$\widehat{\chi_f}(x) = 2^{\frac{n}{2}}(-1)^{\tilde{f}(x)}$$

Due to the involution law the Fourier transform is self-inverse. Thus, the dual \tilde{f} of a bent function f is again a bent function and its own dual is f itself.

Dual functions of elements of \mathfrak{G}_n can be explicitly computed as follows.

Proposition 29. *Let $n = 2m$ with m odd. Let $(a, b) \in \mathbb{F}_{2^n}^* \times \mathbb{F}_4^*$. The dual function of a bent function $g_{a,b}$ of \mathfrak{G}_n is equal g_{a^{2m}, b^2} , that is, we have*

$$\forall \omega \in \mathbb{F}_{2^n}, \quad \widehat{\chi_{g_{a,b}}}(\omega) = 2^m \chi(g_{a^{2m}, b^2}(\omega)).$$

Proof. The arguments are for the most part the same as those used in [14]. Nevertheless, for the sake of completeness, we present below a little shorter proof. Recall that, since the function $g_{a,b}$ is assumed to be bent then, according to Lemma 13, $\sum_{u \in U} \chi(g_{a,b}(u)) = 1$. Given, $\omega \in \mathbb{F}_{2^n}$, since every element x of $\mathbb{F}_{2^n}^*$ has a unique decomposition as : $x = yu$, with $y \in \mathbb{F}_{2^m}^*$ and $u \in U$, one has (in the last equality, we use (6))

$$\begin{aligned} \widehat{\chi_{g_{a,b}}}(w) &:= \sum_{x \in \mathbb{F}_{2^n}} \chi(g_{a,b}(x) + Tr_1^n(wx)) \\ &= 1 + \sum_{u \in U} \sum_{y \in \mathbb{F}_{2^m}^*} \chi(g_{a,b}(yu) + Tr_1^n(wyu)) \\ &= 1 + \sum_{u \in U} \sum_{y \in \mathbb{F}_{2^m}^*} \chi(g_{a,b}(u) + Tr_1^n(wyu)) \end{aligned}$$

Note first that $\widehat{\chi_{g_{a,b}}}(0) = 2^m$. Now, if w is an element of $\mathbb{F}_{2^n}^*$, we have $Tr_m^n(wu) = 0$ if and only if $uw + u^{2^m}w^{2^m} = 0$, that is, $u^{2^m-1} = w^{1-2^m}$. Classical results about character sums says that

$$\sum_{y \in \mathbb{F}_{2^m}} \chi(Tr_1^n(\omega uy)) = \sum_{y \in \mathbb{F}_{2^m}} \chi(Tr_1^m(Tr_m^n(\omega u)y)) = 2^m$$

if $Tr_m^n(\omega u) = 0$, that is, if $u^{2^m-1} = \omega^{1-2^m}$ and, is equal to 0 otherwise. Hence, using properties of trace functions, we have

$$\begin{aligned} \widehat{\chi_{g_{a,b}}}(w) &= 1 + \sum_{u \in U} \chi(g_{a,b}(u)) \left(\sum_{y \in \mathbb{F}_{2^m}} \chi(Tr_1^n(wyu)) - 1 \right) \\ &= 1 - \sum_{u \in U} \chi(g_{a,b}(u)) + 2^m \sum_{\substack{u \in U \\ Tr_m^n(wu)=0}} \chi(g_{a,b}(u)) \\ &= 2^m \chi(Tr_1^n(aw^{3(1-2^m)}) + Tr_1^2(bw^{\frac{1-2^m}{3}})) \\ &= 2^m \chi(Tr_1^n(a^{2^m}w^{3(2^m-1)}) + Tr_1^2(b^{2^m}w^{\frac{2^m-1}{3}})) \\ &= 2^m \chi(Tr_1^n(a^{2^m}w^{3(2^m-1)}) + Tr_1^2(b^{2^m}w^{\frac{2^m-1}{3}})) = 2^m \chi(g_{a^{2^m}, b^2}(\omega)) \end{aligned}$$

($b^{2^m-2} = 1$ because m is being odd then, 3 divides $(2^m + 1)$ and then divides $(2^m - 2)$).

Now, recall that a bent function defined on \mathbb{F}_{2^n} is said to be normal if it is constant on an $\frac{n}{2}$ -dimensional flat $b + E$ where E is a subspace of $\mathbb{F}_{2^{\frac{n}{2}}}$.

Proposition 30. *The bent functions $g_{a,b}$ of \mathfrak{G}_n (where $n = 2m$ with m odd) are normal.*

Proof. Recall that $g_{a,b}$ is constant on each set $u\mathbb{F}_{2^m}^*$, $u \in U = \{x \in \mathbb{F}_{2^n}^* \mid x^{2^m+1} = 1\}$. Choose then u such that $g_{a,b}(u) = 0$. Then $g_{a,b}$ is constant of the vector space $u\mathbb{F}_{2^m}$ (of dimension m) proving that $g_{a,b}$ is normal.

Remark 31. By computer experiments, for small values of n ($n \leq 14$, because of the complexity of the problem) we have found that, the family \mathfrak{G}_n does not contain bent functions when $m = \frac{n}{2}$ is even.

4 Conclusion

In this paper, we contribute to the knowledge of the class of hyper-bent Boolean functions by exhibiting a new infinite family of hyper-bent Boolean functions defined on \mathbb{F}_{2^n} whose expression is of the form $Tr_1^n(ax^{3(2^m-1)}) + Tr_1^2(bx^{\frac{2^m-1}{3}})$ when $m = n/2$ is odd. We thus provide together with the hyper-bent Boolean functions exhibited recently in [14,13] new families of hyper-bent Boolean functions (that does not belong to the class studied in [3]) in which the property

to be hyper-bent is strongly related to the Kloosterman sums. In particular, we extend the known link between Dillon monomial hyper-bent functions and the zeroes of Kloosterman sums to others values and to others functions. In addition, it is important to point out that, unlike the family of hyper-bent functions presented in [14,13], the monomial functions presented in this paper are never bent (and then are not hyper-bent).

References

1. Carlet, C., Gaborit, P.: Hyperbent functions and cyclic codes. *Journal of Combinatorial Theory, Series A* 113(3), 466–482 (2006)
2. Carlitz, L.: Explicit evalution of certain exponential sums. *Math. Scand.* 44, 5–16 (1979)
3. Charpin, P., Gong, G.: Hyperbent functions, Kloosterman sums and Dickson polynomials. *IEEE Trans. Inform. Theory* 9(54), 4230–4238 (2008)
4. Charpin, P., Gong, G.: Hyperbent functions, Kloosterman sums and Dickson polynomials. In: ISIT 2008, Toronto, Canada, July 6–11, pp. 1758–1762 (2008)
5. Charpin, P., Helleseth, T., Zinoviev, V.: The divisibility modulo 24 of Kloosterman sums of $GF(2^m)$, m odd. *Journal of Combinatorial Theory, Series A* 114, 322–338 (2007)
6. Charpin, P., Helleseth, T., Zinoviev, V.: Divisibility properties of Kloosterman sums over finite fields of characteristic two. In: ISIT 2008, Toronto, Canada, July 6–11, pp. 2608–2612 (2008)
7. Charpin, P., Helleseth, T., Zinoviev, V.: Divisibility properties of classical binary Kloosterman sums. *Discrete Mathematics* 309, 3975–3984 (2009)
8. Dillon, J.: Elementary Hadamard difference sets. In: PhD dissertation, University of Maryland (1974)
9. Dillon, J.F.: Elementary Hadamard Difference sets (1975)
10. Dillon, J.F., Dobbertin, H.: New cyclic difference sets with Singer parameters. *Finite Fields and Their Applications* 10(3), 342–389 (2004)
11. Lachaud, G., Wolfmann, J.: The weights of the orthogonals of the extended quadratic binary Goppa codes. *IEEE Trans. Inform. Theory* 36(3), 686–692 (1990)
12. Leander, G.: Monomial Bent Functions. *IEEE Trans. Inform. Theory* 2(52), 738–743 (2006)
13. Mesnager, S.: A new class of bent boolean functions. Submitted to journal Design, Codes and Cryptography (special issue WCC 2009) (2009)
14. Mesnager, S.: A new class of bent boolean functions in polynomial forms. In: Proceedings of international Workshop on Coding and Cryptography, WCC 2009, pp. 5–18 (2009)
15. Rothaus, O.S.: On "bent" functions. *J. Combin. Theory (A)* 20, 300–305 (1976)
16. Youssef, A.M., Gong, G.: Hyper-Bent Functions. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 406–419. Springer, Heidelberg (2001)

The Rayleigh Quotient of Bent Functions

Lars Eirik Danielsen¹, Matthew G. Parker¹, and Patrick Solé²

¹ The Selmer Center, Department of Informatics, University of Bergen, PB 7800,
N-5020 Bergen, Norway
`larsed@ii.uib.no, matthew@ii.uib.no`
² CNRS LTCI, TelecomParisTech, Dept Comelec, Paris, France
`patrick.sole@telecom-paristech.fr`

Abstract. The Rayleigh quotient of a bent function is an invariant under the action of the orthogonal group, and it measures the distance of the function to its dual. An efficient algorithm is derived that generates all bent functions of given Rayleigh quotient. The Rayleigh quotient of some bent functions obtained by primary (Maiorana McFarland, Dillon) or secondary (direct and indirect sum) constructions is computed.

Keywords: Boolean functions, bent functions, Walsh Hadamard transform, Rayleigh quotient, plateaued functions.

1 Introduction

Ever since its introduction by Rothaus, the main problem with the class of Boolean functions known as bent has been the classification. In this article we study a parameter that measures the distance between a function and its dual and this parameter is invariant under the action of the extended orthogonal group, a subgroup of the affine group. We introduced this parameter in [4] and called it the Rayleigh quotient as it is proportional to the Rayleigh quotient (in the sense of numerical analysis) of the matrix of the Walsh Hadamard transform. For a bent function in n variables this quantity in the normalization used here, is an even integer in the range $[-2^n, 2^n]$. It was proved in [4] that a bent function is equal to its dual iff its Rayleigh quotient is 2^n , in which case the function is called **self dual**. Likewise a bent function is the complement of its dual iff its Rayleigh quotient is -2^n , in which case the function is called **anti self dual**. [4] then tabulated the Rayleigh quotient values for all self dual bent Boolean functions in ≤ 6 variables and all quadratic such functions in 8 variables, up to the action of the extended orthogonal group.

This article builds on the results of [4], by tabulating the Rayleigh quotient of all bent Boolean functions of ≤ 6 variables, up to equivalence with respect to the extended orthogonal group, and is organized as follows. Section 2 contains the necessary notation. Section 3 develops the linear algebra needed to study the Rayleigh quotient. Section 4 exploits these ideas to derive an algorithm, more

effective than exhaustive search, to construct all bent functions with prescribed Rayleigh quotient - this algorithm is a variant on that used in [4]. Along the way a connection with plateaued functions is pointed out. Section 5 presents computational results, tabulating the Rayleigh quotients of all bent functions up to 6 variables. Section 6 studies some properties and symmetries of the Rayleigh quotient. Section 7 gives evaluations of the Rayleigh quotient for some special constructions of bent functions: Maiorana McFarland and Dillon, as well as for secondary constructions like the direct and indirect sum.

We are not aware of much work in the literature pertaining to the Rayleigh quotient. However, [6] has considered the respective algebraic degree of a Boolean function and its dual. Moreover, the decompositions of bent functions have been studied in [1], where a link is made between the Walsh Hadamard spectra of the restrictions of a function and the decompositions of its dual.

2 Definitions and Notation

A **Boolean function** f in n variables is any map from \mathbb{F}_2^n to \mathbb{F}_2 . Its **Walsh Hadamard Transform** (WHT), namely $\hat{F} \in \mathbb{R}^{2^n}$, can be defined as

$$\hat{F}(x) := \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)+x \cdot y},$$

where $x \cdot y$ denotes the dot product of x with y . The **sign function** of f is defined by $F := (-1)^f$. The Boolean function, f , is said to be **bent** iff $|\hat{F}(x)| = 2^{n/2}$, $\forall x$, which is only possible if n is even. If f is bent then its **dual** with respect to the WHT, \tilde{f} , is also a bent Boolean function. Let \tilde{F} be the sign function of \tilde{f} for the case that f is bent. Then the duality of \tilde{F} to f is defined by

$$\tilde{F}(x) := 2^{-n/2} \hat{F}(x) \Leftrightarrow (-1)^{\tilde{f}(x)} := 2^{-n/2} \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)+x \cdot y}, \quad f \text{ bent.}$$

The matrix of the WHT is the Hadamard matrix H_n of Sylvester type, which we now define by tensor products. Let

$$H := \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Let $H_n := H^{\otimes n}$ be the n -fold tensor product of H with itself. Thus $H_n = H_{n-1} \otimes \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}$, where $H_1 = H$. Let $\mathcal{H}_n := 2^{-n/2} H_n$, be its normalized version. Recall the Hadamard property

$$H_n H_n^T = 2^n I_{2^n},$$

where we denote by I_M the M by M identity matrix. View F as a vector $F = (F_{0...00}, F_{0...01}, \dots, F_{1...11}) \in \mathbb{F}_2^n$, whose elements, F_x , are ordered lexicographically in x . Let \hat{F} have a similar vector interpretation. Then we can express the WHT in matrix-vector form as

$$\hat{F} = FH_n.$$

For example, when $n = 2$ and $f(y_1, y_2) = y_1 y_2$, we have $F = (1, 1, 1, -1)$ and

$$\hat{F} = FH_2 = \begin{pmatrix} 1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 2 & -2 \end{pmatrix}.$$

In this case $|\hat{F}_x| = 2$, $\forall x$, so f is bent and its dual is $\tilde{f}(x_1, x_2) = x_1 x_2$, where $\tilde{F} = (-1)^{\tilde{f}} = 2^{-1}\hat{F} = (1, 1, 1, -1)$.

If f is not only bent but, furthermore, $f = \tilde{f}$, then f is **self dual bent** - such is the case for the example just given. This means that the sign function of f is an eigenvector of \mathcal{H}_n attached to the eigenvalue 1. Similarly, if $f = \tilde{f} + 1$ then f is **anti self dual bent**. For example, $f(y_1, y_2) = y_1 y_2 + y_1 + y_2$ is anti self dual bent. This means that its sign function is an eigenvector of \mathcal{H}_n attached to the eigenvalue -1 . Define the **Rayleigh quotient** S_f of a Boolean function f in n variables by the character sum

$$S_f := \sum_{x, y \in \mathbb{F}_2^n} (-1)^{f(x)+f(y)+x \cdot y} = \sum_{x \in \mathbb{F}_2^n} F(x)\hat{F}(x).$$

Define the **normalized Rayleigh quotient** N_f of a bent Boolean function f in n variables by the character sum

$$N_f := \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+\tilde{f}(x)} = 2^{-n/2} S_f.$$

We see that $N_f = 2^n$ if f is self dual bent and $N_f = -2^n$ if f is anti self dual bent.

3 Linear Algebra

We now establish an orthogonal eigen-decomposition of the sign function of a Boolean function and use it to obtain expressions for the Rayleigh quotient of a bent Boolean function in terms of this eigen-decomposition. Recall the following elementary Lemma from [4].

Lemma 1. *The spectrum of \mathcal{H}_n consists of the two eigenvalues ± 1 , each with multiplicity 2^{n-1} . A basis of the eigenspace attached to the eigenvalues 1 (resp. -1) is formed from the rows of the $2^{n-1} \times 2^n$ matrix $(H_{n-1} + 2^{n/2}I_{2^{n-1}}, H_{n-1})$ (resp. $(H_{n-1} - 2^{n/2}I_{2^{n-1}}, H_{n-1})$). An orthogonal decomposition of \mathbb{R}^{2^n} in eigenspaces of H_n is*

$$\mathbb{R}^{2^n} = \text{Ker}(H_n + 2^{n/2}I_{2^n}) \oplus \text{Ker}(H_n - 2^{n/2}I_{2^n}).$$

Proof. The basis characterization follows because

$$\begin{aligned} & (H_{n-1} + 2^{n/2}I_{2^{n-1}}, H_{n-1})\mathcal{H}_n \\ &= (H_{n-1} + 2^{n/2}I_{2^{n-1}}, H_{n-1})2^{-n/2}(H_{n-1} \otimes H_1) \\ &= 2^{-n/2}(2H_{n-1} + 2^{n/2}I_{2^{n-1}}, 2^{n/2}I_{2^{n-1}})(H_{n-1} \otimes I_2) \\ &= (2^{n/2}I_{2^{n-1}} + H_{n-1}, H_{n-1}). \end{aligned}$$

Similar arguments show that

$$(H_{n-1} - 2^{n/2}I_{2^{n-1}}, H_{n-1})\mathcal{H}_n = -(H_{n-1} + 2^{n/2}I_{2^{n-1}}, H_{n-1}).$$

The kernel (i.e. nullspace) of $H_n + 2^{n/2}I_{2^n}$ is the row space of a matrix, say M^+ , such that $(H_n + 2^{n/2}I_{2^n})M^+ = 0$. From the above basis characterization we see that one choice is $M^+ = (H_{n-1} - 2^{n/2}I_{2^{n-1}}, H_{n-1})$. Similarly, $\text{Ker}(H_n - 2^{n/2}I_{2^n})$ is the row space of $M^- = (H_{n-1} + 2^{n/2}I_{2^{n-1}}, H_{n-1})$. The orthogonal decomposition of \mathbb{R}^{2^n} follows because the two kernels are orthogonal, i.e. because $M^+M^{-T} = 0$. \square

By Lemma 1, the orthogonal decomposition in eigenspaces of H_n yields the following decomposition for the sign function F of a Boolean function, $F = F^+ + F^-$, with $F^\pm \in \text{Ker}(H_n \pm 2^{n/2}I_{2^n})$, and $\langle F, F \rangle = \langle F^+, F^+ \rangle + \langle F^-, F^- \rangle$, where $\langle A, B \rangle$ is the inner product of real vectors A and B . By observing that $\hat{F} = 2^{n/2}(F^+ - F^-)$, and that $S_f = \langle F, \hat{F} \rangle$, we obtain

$$N_f = \langle F^+, F^+ \rangle - \langle F^-, F^- \rangle,$$

and by the triangle inequality, $|N_f| \leq 2^n$, with equality if and only if $F = F^+$ or $F = F^-$. If f is bent then the sign function, \tilde{F} , of its dual exists, and

$$\tilde{F} = F^+ - F^-.$$

Thus $F \pm \tilde{F} = 2F^\pm$ has entries in $\{0, \pm 2\}$, so both F^+ and F^- have entries in $\{0, \pm 1\}$. Denote by S_+ (resp. S_-) the set of $x \in \mathbb{F}_2^n$ such that $F_x^+ = 0$ (resp. $F_x^- = 0$). Because $F = F^+ + F^-$ has entries in $\{\pm 1\}$, it follows that the sets S_+ and S_- partition \mathbb{F}_2^n . Conversely, given a pair of eigenvectors of \mathcal{H}_n , F^+ and F^- , with entries in $\{0, \pm 1\}$, and with corresponding sets S_+ and S_- , such that $S_+ \cup S_- = \mathbb{F}_2^n$, then the sum of F^+ and F^- is the sign function of a bent function. In summary

Proposition 1. *Let F be the sign function of a bent Boolean function of n variables. Then there exist two vectors F^+ and F^- , and two subsets, S_+ and S_- , with the following properties.*

1. $F = F^+ + F^-$
2. F^+ and F^- have entries in $\{0, \pm 1\}$.
3. the sets S_+ and S_- partition \mathbb{F}_2^n .
4. $F_x^\pm = 0$ iff $x \in S_\pm$.

Conversely, given eigenvectors, F^\pm , of \mathcal{H}_n , and sets S_\pm with the last three properties, the sum $F^+ + F^-$ is the sign function of a bent function with normalized Rayleigh quotient

$$N_f = |S_-| - |S_+| = 2^n - 2|S_+| = 2|S_-| - 2^n.$$

Moreover, $|S_+| = d_H(f, \tilde{f})$, where $d_H(\cdot, \cdot)$ denotes the Hamming distance.

Example: Let $n = 4$ and $f = x_1x_3 + x_2x_3 + x_2x_4 + x_2$. Then $F = (1, 1, 1, 1, -1, 1, 1, -1, 1, 1, -1, -1, 1, 1, -1, 1)$. As f is a bent function it has a dual. By computation, $\tilde{f} = x_1x_3 + x_1x_4 + x_2x_4 + x_4$ and $\tilde{F} = (1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, -1, -1, 1)$. It follows that $F^+ = (F + \tilde{F})/2 = (1, 0, 1, 0, 0, 1, 1, 0, 1, 1, -1, -1, 0, 0, -1, 1)$, giving $S_+ = \{0010, 0011, 1000, 1011, 1100, 1110\}$. Therefore the normalized Rayleigh quotient of f is $N_f = 2^n - 2|S_+| = 4$.

4 Search Algorithm

We now describe an algorithm where we construct all bent Boolean functions, F , of n variables, given a specified zero set, S_+ , for F^+ , where $F = F^+ + F^-$. For an arbitrary n -variable function, $A(x_1, x_2, \dots, x_n)$, with domain \mathbb{F}_2^n , let $A|_{x_1=0}$ (resp. $A|_{x_1=1}$) be the restrictions of A to $x_1 = 0$ and $x_1 = 1$ respectively, such that $A(x) = (A|_{x_1=0}, A|_{x_1=1})$.

Let $F^+ := (Y, Z)$, where $Y, Z \in \mathbb{R}^{2^{n-1}}$, such that $Y := F^+|_{x_1=0}$, and $Z := F^+|_{x_1=1}$, i.e. F^+ is the concatenation of Y with Z . Let $S_+ \subset \mathbb{F}_2^n$ similarly be decomposed into S_+^Y and S_+^Z , where

$$S_+^Y := \{s|_{x_1=0} \mid s \in S_+\} \subset \mathbb{F}_2^{n-1}, \quad S_+^Z := \{s|_{x_1=1} \mid s \in S_+\} \subset \mathbb{F}_2^{n-1}.$$

We want to construct an F^+ with zero set S_+ .

Theorem 1. *Let Z have entries in $\{0, \pm 1\}$, with $Z_x = 0$ iff $x \in S_+^Z$. Define $Y := Z + \frac{2H_{n-1}}{2^{n/2}}Z$. If Y has entries in $\{0, \pm 1\}$, with $Y_x = 0$ iff $x \in S_+^Y$, then the vector $F^+ = (Y, Z)$ is in the eigenspace of \mathcal{H}_n attached to 1 with zero set S_+ .*

Proof. By Lemma 1, for eigenspace 1, we consider an X such that $F^+ = X(H_{n-1} + 2^{n/2}I_{2^{n-1}}, H_{n-1}) = (Y, Z)$, from which it follows that $Y = Z + \frac{2H_{n-1}}{2^{n/2}}Z$. Moreover, we require that Y and Z both have, by Proposition 1, entries in $\{0, \pm 1\}$. For each arbitrary choice of Z with entries in $\{0, \pm 1\}$, we can then check whether Y has entries in $\{0, \pm 1\}$. \square

A similar result holds for F^- , for $F^- := (Y, Z)$, $Y := F^-|_{x_1=0}$ and $Z := F^-|_{x_1=1}$. The proof is analogous and is omitted.

Theorem 2. *Let Z have entries in $\{0, \pm 1\}$, with $Z_x = 0$ iff $x \in S_-^Z$. Define $Y := Z - \frac{2H_{n-1}}{2^{n/2}}Z$. If Y has entries in $\{0, \pm 1\}$, with $Y_x = 0$ iff $x \in S_-^Y$, then the vector $F^- = (Y, Z)$ is in the eigenspace of \mathcal{H}_n attached to -1 with zero set S_- .*

Based on Proposition 1 and the above two theorems we give an algorithm to generate all bent functions with given zero set S_+ , and therefore, from Proposition 1, with fixed Rayleigh quotient $2^n - 2|S_+|$.

Algorithm *BWS*(n, S_+)

1. Pick Z with entries in $\{0, \pm 1\}$, and $Z_x = 0$ iff $x \in S_+^Z$
2. Compute all candidate Y as $Y := Z + \frac{2H_{n-1}}{2^{n/2}}Z$.
3. If Y has entries in $\{0, \pm 1\}$ and $Y_x = 0$ iff $x \in S_+^Y$ let $F^+ := (Y, Z)$, else go to next Z .
4. Pick Z with entries in $\{0, \pm 1\}$, and $Z_x = 0$ iff $x \notin S_+^Z$
5. Compute all candidate Y as $Y := Z - \frac{2H_{n-1}}{2^{n/2}}Z$.
6. If Y has entries in $\{0, \pm 1\}$ and $Y_x = 0$ iff $x \notin S_+^Y$ let $F^- := (Y, Z)$, else go to next Z .
7. Output $F = F^+ + F^-$ for all F^+ found in step 3 and all F^- found in step 6.

It should be noted that, compared to brute force exhaustive search of complexity 2^{2^n} this algorithm is of complexity 2^R with $R \leq 2^{n-1}$, depending on the size of S .

We point out a connection with **plateaued functions**. Recall that, according to [9], a Boolean function f in n variables is plateaued of order r if the entries of its WHT, \hat{F} , have modulus either zero or $2^{n-r/2}$, where r is even and can range from 0 to n . If $r = n$ then f is bent.

Theorem 3. *Keep the notation of Proposition 1. Write*

$F^+ = (Y^+, Z^+)$ and $F^- = (Y^-, Z^-)$. If Y^+ and Z^+ (resp. Y^- and Z^-) have the same supports, that is

$$\{x \mid Y_x^\pm = 0\} = \{x \mid Z_x^\pm = 0\},$$

then both $Z^+ + Z^-$ and $Z^+ - Z^-$ (resp. $Y^+ + Y^-$ and $Y^+ - Y^-$) are sign functions of plateaued functions of order $n-2$ in $n-1$ variables.

Proof. By Proposition 1 both $Z^+ \pm Z^-$ and $Y^+ \pm Y^-$ have entries in $\{\pm 1\}$ and are thus legitimate sign functions of Boolean functions in $n-1$ variables. By hypothesis, $\frac{Y^+ - Z^+}{2}$ and $\frac{Z^- - Y^-}{2}$ have entries in $\{0, \pm 1\}$. By Proposition 1 their sum and difference still have entries in $\{0, \pm 1\}$. Like in Theorem 1 and 2 we have

$$H_{n-1}Z^+ = 2^{n/2}\left(\frac{Y^+ - Z^+}{2}\right) \quad (1)$$

and, symmetrically,

$$H_{n-1}Z^- = 2^{n/2}\left(\frac{Z^- - Y^-}{2}\right) \quad (2)$$

The result follows now by adding and subtracting equations 1 and 2. \square

Note that this result is different from the construction of bent functions from complementary plateaued functions in [8].

5 Numerical Results

In previous work, we have classified self dual bent functions [4]. We here extend this result by calculating the Rayleigh quotient of all bent functions of up to six

Table 1. Number of Bent Functions of Four Variables with given Rayleigh Quotient

N_f	Functions
± 16	40
± 8	192
± 4	384
0	280
Total	896

Table 2. Number of Bent Functions of Six Variables with given Rayleigh Quotient

N_f	Functions
± 64	85,792
± 48	814,080
± 40	5,225,472
± 36	10,813,440
± 32	33,686,400
± 28	61,931,520
± 24	159,169,536
± 20	327,155,712
± 16	548,066,304
± 12	865,075,200
± 8	1,194,362,880
± 4	1,434,058,752
0	784,985,440
Total	5,425,430,528

Table 3. Number of Quadratic Bent Functions of Six Variables with given Rayleigh Quotient

N_f	Functions
± 64	1504
± 32	44,160
± 16	503,808
± 8	737,280
0	490,912
Total	1,777,664

variables. Tables 1 and 2 list the number of bent functions, where no symmetries are taken into account, of four and six variables with normalized Rayleigh quotient N_f . Table 3 gives the Rayleigh quotients of all quadratic bent functions of six variables. It follows from Theorem 5 that there will always be the same

number of functions with $N_f = -x$ as there are functions with $N_f = x$, so these functions are counted together. (For instance, there are 20 bent functions of four variables with $N_f = 16$, and 20 such functions with $N_f = -16$.)

According to Prop. 3, $|N_f| \leq 60$ for a bent function of six variables that is neither self dual nor anti self dual. We observe that no function meeting this bound with equality exists. Up to equivalence, where we consider the functions f and g to be in the same equivalence class if $g(x) = f(Lx + d) + d \cdot x + c$, where $LL^T = I$, $d \in \mathbb{Z}_2^n$, $\text{wt}(d)$ even, and $c \in \mathbb{Z}_2$, there are seven bent functions of six variables with $N_f = 48$. Representatives from each equivalence class are listed below. The functions with $N_f = -48$ can be obtained from Theorem 5 (For a classification of the bent functions of six variables with $|N_f| = 64$, we refer to previous work [4].)

1. $x_1x_2x_6 + x_1x_3x_6 + x_1x_4x_5 + x_1x_3x_5 + x_2x_4x_6 + x_3x_4x_6 + x_2x_4x_5 + x_2x_3x_5 + x_1x_2 + x_1x_5 + x_2x_6 + x_3x_4 + x_3x_6 + x_4x_5 + x_5x_6$
2. $x_2x_4x_6 + x_3x_5x_6 + x_1x_4x_5 + x_1x_2x_3 + x_1x_6 + x_2x_5 + x_2x_6 + x_3x_4 + x_3x_5 + x_3x_6 + x_4x_5 + x_4x_6 + x_5x_6 + x_2 + x_3$
3. $x_2x_3x_4 + x_1x_3x_4 + x_2x_3x_6 + x_2x_3x_5 + x_2x_4x_6 + x_2x_4x_5 + x_2x_5x_6 + x_1x_5x_6 + x_1x_2 + x_3x_5 + x_4x_6 + x_5x_6$
4. $x_1x_2x_3 + x_1x_2x_5 + x_1x_3x_4 + x_1x_4x_5 + x_1x_6 + x_2x_4 + x_2x_6 + x_3x_5 + x_3x_6 + x_4x_6 + x_5x_6 + x_1 + x_2 + x_3$
5. $x_1x_2x_3 + x_1x_4x_5 + x_1x_3x_5 + x_3x_5x_6 + x_1x_6 + x_2x_5 + x_3x_4 + x_3x_6 + x_4x_5 + x_5x_6 + x_1 + x_2 + x_3 + x_4$
6. $x_1x_3x_5 + x_1x_2x_5 + x_1x_3x_4 + x_1x_2x_4 + x_1x_6 + x_2x_5 + x_3x_4 + x_4x_6 + x_5x_6 + x_1 + x_2 + x_4$
7. $x_2x_3x_6 + x_2x_4x_6 + x_3x_5x_6 + x_4x_5x_6 + x_1x_6 + x_2x_3 + x_4x_5$

Table 4. Number of Boolean Functions of 4 Variables with given Rayleigh Quotient

N_f	Functions
± 16	40
± 13	416
± 12	800
± 11	1504
± 10	2560
± 9	2944
± 8	3904
± 7	4992
± 6	5632
± 5	6816
± 4	7264
± 3	7648
± 2	8192
± 1	8448
0	4376
Total	65,536

Table 5. Number of Boolean Functions of 5 Variables with given Non-Normalized Rayleigh Quotient

S_f	Functions
± 160	8960
± 155	23,040
± 150	50,688
± 145	150,528
± 140	840,320
± 135	1,039,360
± 130	1,627,392
± 125	2,581,792
± 120	9,404,480
± 115	7,907,840
± 110	10,849,152
± 105	14,716,416
± 100	44,280,000
± 95	31,537,920
± 90	38,784,320
± 85	47,529,984
± 80	125,472,000
± 75	79,892,480
± 70	92,338,176
± 65	105,623,232
± 60	254,490,560
± 55	149,760,000
± 50	164,694,016
± 45	180,602,112
± 40	404,723,200
± 35	224,425,920
± 30	236,937,728
± 25	249,284,160
± 20	529,400,320
± 15	277,094,400
± 10	284,104,704
± 5	288,219,136
0	436,572,960
Total 4,294,967,296	

We have also calculated the Rayleigh quotients of all Boolean functions of four and five variables, listed in Tables 4 and 5. For five variables, the non-normalized values S_f are given, since the values N_f are not integer for odd n .

We observe that for Boolean functions of four variables, the highest value of $|N_f| < 16$ is $|N_f| = 13$. Up to equivalence, the following three functions have $N_f = 13$. (For a classification of the functions of four variables with $|N_f| = 16$, we refer to previous work [4].)

1. $x_1x_2x_3x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_1$
2. $x_1x_2x_3x_4 + x_1x_2 + x_1x_3 + x_2x_4 + x_3x_4$
3. $x_1x_2x_3x_4 + x_1x_2 + x_3x_4$

For Boolean functions of five variables, the highest obtainable Rayleigh quotient is $|S_f| = 160$. Up to equivalence there are four functions with $S_f = 160$, which are listed below. In general, it is not known what the highest possible value of $|S_f|$ for odd n is.

1. $x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_4x_5 + x_3x_4x_5 + x_1x_4 + x_1x_5 + x_2x_3$
2. $x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3 + x_1x_4 + x_1x_5 + x_3x_4 + x_3x_5 + x_4x_5 + x_1$
3. $x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_3x_4 + x_3x_5 + x_4x_5 + x_1 + x_2 + x_3 + x_4$
4. $x_1x_2x_3 + x_1x_2 + x_1x_3 + x_4x_5$

6 Properties of the Rayleigh Quotient

6.1 Elementary Properties

The normalized Rayleigh quotient is the sum of 2^n terms ± 1 . Therefore

Proposition 2. *The normalized Rayleigh quotient N_f of a bent Boolean function f is an even integer (negative or positive).*

For bent functions that are neither self dual nor anti self dual we can improve on the estimate of N_f over [4].

Proposition 3. *Let f be a bent function in n variables. If f is neither self dual nor anti self dual then $|N_f| \leq 2^n - 4$.*

Proof. By the proof of Theorem 1 we see that if $Z = 0$ then $X = 0$ and $F^+ = 0$ forcing f to be anti self dual. A similar argument for F^- shows that we cannot have $Z = 0$ for F^- . It follows, to avoid either situation, that S_+ cannot have size $2^n - 1$. The result follows. \square

6.2 Symmetries

In this section we give symmetries, that is operations, on Boolean functions that preserve bentness and the Rayleigh quotient. Define, following [7], the **orthogonal group** of index n over \mathbb{F}_2 as

$$\mathcal{O}_n := \{L \in GL(n, 2) \& LL^t = I_n\}.$$

Observe that $L \in \mathcal{O}_n$ if and only if (I_n, L) is the generator matrix of a self dual binary code of length $2n$. Thus, for even n , an example is $I_n + J_n$ with J_n = all-one matrix.

Theorem 4. *Let f denote a bent function in n variables. If $L \in \mathcal{O}_n$ and $c \in \{0, 1\}$ then $g(x) := f(Lx) + c$ is also bent, and $N_g = N_f$.*

Proof. The WHT of g is

$$\hat{G}(x) = (-1)^c \hat{F}(Lx) = 2^{n/2} (-1)^{\tilde{f}(Lx)+c} = 2^{n/2} (-1)^{\tilde{g}(x)},$$

where the first equality holds by observing that $x.y = L(x).L(y)$, and by a change of variable involving $L^{-1} = L^T$, and the last equality by definition of \tilde{g} . Computing the normalized Rayleigh quotient of g yields

$$N_g = \langle (-1)^g, (-1)^{\tilde{g}} \rangle = \sum_x (-1)^{f(Lx)} (-1)^{\tilde{f}(Lx)} = N_f. \quad \square$$

Theorem 5. Let f denote a bent function in n variables. Define g by $g(x) := f(x+d) + d \cdot x$. If $d \in \mathbb{F}_2^n$ then g is also bent, and $N_g = (-1)^{d \cdot d} N_f$.

Proof. A change of variables $y = x + d$ in the definition of \hat{G} yields $\hat{G}(y) = (-1)^{d \cdot (y+d)} \times \hat{F}(y+d)$. Therefore g is bent with dual function

$$\tilde{g}(y) = d \cdot (y+d) + \tilde{f}(y+d).$$

Adding up yields

$$g(y) + \tilde{g}(y) = d \cdot d + f(y+d) + \tilde{f}(y+d).$$

The result follows after a change of variables. \square

Theorem 5 explains why, for every function, f , with normalized Rayleigh quotient N_f , there exists a family of functions, $\{f_e\}$ each with normalized Rayleigh quotient, N_f , and an equal size family of functions, $\{f_o\}$, each with normalized Rayleigh quotient, $-N_f$, as obtained by evaluating g for even and odd weight values of d , respectively.

We refer, in this and related work, to the combined action of the symmetries of theorems 4 and 5 as the action of the **extended orthogonal group**, being a subgroup of the affine group.

7 Special Constructions

In [4] primary constructions for (anti) self dual bent functions were presented for the case of Maiorana McFarland, Dillon's partial spreads, and monomial power functions. Secondary constructions using both direct and indirect sum were also presented. We now generalise this work, in the cases of Maiorana McFarland and partial spreads, and for direct and indirect sum, to the situation where the Rayleigh quotient can have magnitude less than 2^n .

7.1 Maiorana McFarland

A general class of bent functions is the **Maiorana McFarland** class, that is functions of the form

$$x \cdot \phi(y) + g(y)$$

with $x, y \in \mathbb{F}_2^{n/2}$, $\phi : \mathbb{F}_2^{n/2} \rightarrow \mathbb{F}_2^{n/2}$, a permutation, and g arbitrary Boolean.

Theorem 6. A Maiorana-McFarland function $f = x \cdot \phi(y) + g(y)$ with $\phi(x) = L(x) + a$, $L \in GL(n/2, 2)$ and unitary ($L^T = L^{-1}$), and $a \in \mathbb{F}_2^{n/2}$, has normalized Rayleigh quotient

$$N_f = (-1)^{a \cdot a} \times \left(\sum_x (-1)^{g(x) + a \cdot L(x)} \right)^2.$$

Proof. The dual of a Maiorana-McFarland bent function $x \cdot \phi(y) + g(y)$ is equal to $\phi^{-1}(x) \cdot y + g(\phi^{-1}(x))$ [5]. Computing the normalized Rayleigh quotient of f yields, after replacing x by $\phi(x)$,

$$N_f = \langle (-1)^f, (-1)^{\tilde{f}} \rangle = \sum_{x,y} (-1)^{\phi(x) \cdot \phi(y) + x \cdot y + g(x) + g(y)},$$

and, since, for L unitary, $L(x) \cdot L(y) = x \cdot y$,

$$N_f = \sum_{x,y} (-1)^{g(x) + a \cdot L(x) + g(y) + a \cdot L(y) + a \cdot a}.$$

The result follows. \square

The proof of the following corollary is omitted.

Corollary 1. If $g(x) + a \cdot L(x)$ is constant, then f is self dual (resp. anti self dual) if a has even (resp. odd) weight, i.e. $N_f = 1$ (resp. $N_f = -1$), and, if $g(x) + a \cdot L(x)$ is balanced then $N_f = 0$.

7.2 Dillon Functions

Let $x, y \in \mathbb{F}_{2^{n/2}}$. The class denoted by \mathcal{PS}_{ap} in [5] consists of so-called Dillon's function of the type

$$f(x, y) = g(x/y)$$

with the convention that $x/y = 0$ if $y = 0$, and where g is a balanced Boolean function and $g(0) = 0$. We introduce the character sum

$$K_g := \sum_u (-1)^{g(u) + g(1/u)}.$$

In particular, if $g = Tr$ then K_g is a **Kloosterman sum**.

Theorem 7. Let f be a bent function constructed from a Dillon g as above. Its Rayleigh quotient is

$$N_f = 2^{n/2} + (2^{n/2} - 1)K_g.$$

Proof. The dual of $f = g(x/y)$ is $\tilde{f} = g(y/x)$. Therefore

$N_f = \sum_{x,y} (-1)^{g(x/y) + g(y/x)}$. Noting when y vanishes and making the change of variables $u = x/y$ when $y \neq 0$ gives the result. \square

7.3 Direct and Indirect Sums

If f and g are Boolean functions in n and m variables, respectively, define the **direct sum** of f and g as the Boolean function on $n + m$ variables given by $f(x) + g(y)$. The following result is immediate, and its proof is omitted.

Proposition 4. *If f and g are bent functions their direct sum is bent of Rayleigh quotient $N_f N_g$.*

A more general construction involving four functions can be found in [3]. If a, b and c, d are two pairs of Boolean functions in n and m variables, respectively, define the **indirect sum** of these four functions by

$$f(x, y) := a(x) + d(y) + (a(x) + b(x))(c(y) + d(y)).$$

It is shown in [3], and also reviewed in [2], that, if a, b, c, d are bent functions, then f is a bent function, and

Lemma 2

$$\tilde{f} = \tilde{a} + \tilde{d} + (\tilde{a} + \tilde{b})(\tilde{c} + \tilde{d}).$$

We further show that,

Proposition 5. *If a, b and c, d are two pairs of dual bent functions, i.e. such that $b = \tilde{a}$ and $d = \tilde{c}$, then f and $g = b + c + (a + b)(c + d)$ are also dual bent functions, i.e. $g = \tilde{f}$. Furthermore the Rayleigh quotient of both f and g is*

$$N_f = N_a N_c.$$

Proof. Comes from

$$\tilde{f} = f + (a + b) + (c + d),$$

and the definition of N_a and N_b . The result follows. \square

A generalisation on this theme is the following

Proposition 6. *If a, b and c, d are two pairs of bent functions satisfying $b = \tilde{a} + \epsilon$, $d = \tilde{c} + \mu$, for $\epsilon, \mu \in \{0, 1\}$, then $f = a + d + (a + b)(c + d)$ and $g = b + c + (a + b)(c + d)$ are both bent. Furthermore the Rayleigh quotient of both is*

$$N_f = N_a N_c.$$

Proof. A direct computation using Lemma 2 shows that

$$f + \tilde{f} = (a + b) + (c + d) + (\epsilon + \mu).$$

The result follows. \square

Finally,

Proposition 7. *Let a and b be self dual or anti self dual bent Boolean functions over m variables. Let $d_H(a, b)$ be the Hamming distance between a and b . Let c and d both be bent Boolean functions over n variables. Then, $f = a + d + (a + b)(c + d)$ and $g = b + c + (a + b)(c + d)$ are both bent over $n + m$ variables, and*

$$\begin{aligned} N_f &= (2^m N_d + d_H(a, b)(N_c - N_d))(-1)^{e_a}, \quad e_a = e_b, \\ N_g &= (2^m N_c - d_H(a, b)(N_c - N_d))(-1)^{e_b}, \quad " \\ N_g &= (2^m N_d + d_H(a, b)(N_c - N_d))(-1)^{e_a}, \quad e_a \neq e_b, \\ N_f &= (2^m N_c - d_H(a, b)(N_c - N_d))(-1)^{e_b}, \quad " \end{aligned}$$

where $e_a, e_b \in \mathbb{F}_2$, $e_a = 0$ (resp. 1) if a is self dual (resp. anti self dual), $e_b = 0$ (resp. 1) if b is self dual (resp. anti self dual).

Proof. From Lemma 2 and the (anti) self dual properties of a and b , we obtain

$$\tilde{f} = a + \tilde{d} + (a + b + e_a + e_b)(\tilde{c} + \tilde{d}) + e_a.$$

Therefore

$$f + \tilde{f} = d + \tilde{d} + (a + b + e_a + e_b)(c + \tilde{c} + d + \tilde{d}) + e_a.$$

Consider the case where $e_a = e_b$. For $x \in \mathbb{F}_2^m$ such that $a(x) + b(x) = 0$ (resp. 1), the previous equation then reduces to $f + \tilde{f} = d + \tilde{d} + e_a$ (resp. $f + \tilde{f} = c + \tilde{c} + e_a$). From Proposition 1. we see that an n -variable bent Boolean function, h , has Rayleigh quotient given by $N_h = 2^n - 2d_H(h, \tilde{h})$. Plugging this back into the previous equations, we obtain

$$(-1)^{e_a} N_f = 2^{n+m} - 2 \left((2^m - d_H(a, b))d_H(d, \tilde{d}) + d_H(a, b)d_H(c, \tilde{c}) \right)$$

which, after some re-arrangements, gives the expression for N_f when $e_a = e_b$ in the Proposition. Similar arguments can be used to obtain the expression for N_f when $e_a \neq e_b$, and, likewise, one obtains similar expressions for N_g . \square

It follows immediately from the proposition that, if $a+b$ is balanced, and $e_a = e_b$, then $N_f = N_g = (-1)^{e_a} 2^{m-1}(N_c + N_d)$. If, further to this, $N_c = -N_d$, e.g. if c is self dual and d is anti self dual, then $N_f = N_g = 0$. Also, from the proposition,

$$N_f + N_g = 2^m(N_c + (-1)^{e_a+e_b} N_d)(-1)^{e_b}.$$

References

1. Canteaut, A., Charpin, P.: Decomposing bent functions. IEEE Trans. Inform. Theory. 49, 2004–2019 (2003)
2. Carlet, C.: Boolean Functions for Cryptography and Error Correcting Codes. In: Hammer, P., Crama, Y. (eds.) Boolean methods and models. Cambridge University Press, Cambridge (to appear)

3. Carlet, C.: On the secondary constructions of resilient and bent functions. In: Feng, K., Niederreiter, H., Xing, C. (eds.) *Proceedings of the Workshop on Coding, Cryptography and Combinatorics 2003. Progress in Comp. Sc. and Appl. Logic*, pp. 3–28. Birkhäuser, Basel (2004)
4. Carlet, C., Danielsen, L.E., Parker, M.G., Solé, P.: Self dual bent functions (submitted 2009)
5. Dillon, J.F.: Elementary Hadamard Difference Sets. Ph.D. thesis, Univ. of Maryland (1974)
6. Hou, X.-D.: New constructions of bent functions. *J. Combin. Inform. System Sci.* 25(1–4), 173–189 (2000)
7. Janusz, G.J.: Parametrization of self-dual codes by orthogonal matrices. *Finite Fields and Their Applications* 13(3), 450–491 (2007)
8. Zheng, Y., Zhang, X.M.: Relationships between bent functions and complementary plateaued functions. In: Song, J.S. (ed.) *ICISC 1999. LNCS*, vol. 1787, pp. 60–75. Springer, Heidelberg (2000)
9. Zheng, Y., Zhang, X.-M.: On plateaued functions. *IEEE Trans. Inform. Theory* 47, 1215–1223 (2001)

Cache Timing Analysis of LFSR-Based Stream Ciphers

Gregor Leander¹, Erik Zenner¹, and Philip Hawkes²

¹ Technical University of Denmark

Department of Mathematics

{g.leander,e.zenner}@mat.dtu.dk

² Qualcomm Incorporated

phawkes@qualcomm.com

Abstract. Cache timing attacks are a class of side-channel attacks that is applicable against certain software implementations. They have generated significant interest when demonstrated against the Advanced Encryption Standard (AES), but have more recently also been applied against other cryptographic primitives.

In this paper, we give a cache timing cryptanalysis of stream ciphers using word-based linear feedback shift registers (LFSRs), such as Snow, Sober, Turing, or Sosemanuk. Fast implementations of such ciphers use tables that can be the target for a cache timing attack. Assuming that a small number of noise-free cache timing measurements are possible, we describe a general framework showing how the LFSR state for any such cipher can be recovered using very little computational effort. For the ciphers mentioned above, we show how this knowledge can be turned into efficient cache-timing attacks against the full ciphers.

1 Introduction

Cache Timing Attacks [24,19,23] are a class of side-channel attacks. They assume that the adversary can use timing measurements to learn something about the cache accesses of a legitimate party, which turns out to be the case in some practical applications. In 2005, Bernstein [2] and Osvik, Shamir, and Tromer [17,18] showed in independent work that the Advanced Encryption Standard (AES) is particularly vulnerable to this type of side-channel attack, generating a lot of attention for the field. Subsequent work dealt with verifying the findings [16,15,14,22,7], improving the attack [20,3,13,5], and devising and analyzing countermeasures [6,4,25].

However, the cryptanalytic attention was mainly focussed on AES, while other ciphers were treated only handwavingly. For example, the eStream report on side-channel attacks [10] simply categorizes all stream ciphers that use tables in their implementations as vulnerable, independent of whether or not a cache timing attack was actually feasible. The cache timing analysis of the HC-256 stream cipher [26] presented at SAC 2008 was the first paper to actually analyze the cache timing resistance of a stream cipher. It also provided a model for the design and analysis of stream ciphers with regards to cache timing attacks.

In this paper, we discuss a different class of stream ciphers, namely those using tables to speed up software implementations of word-based linear feedback shift registers (LFSRs). The technique was introduced around the year 2000 and is used by ciphers such as Snow [8,9], Sober-128 [11], Turing [21], or Sosemanuk [1].

1.1 Organisation of the Paper

The paper is organized as follows. In Section 2, we review the cache timing model that is used for the analysis and discuss its goal and practical relevance. Section 3 gives the general framework for the attack and describes its applicability to a wide range of stream cipher, including but not limited to the concrete examples being discussed in Section 4. Finally, in Section 5, we summarize our findings.

1.2 Notation

All ciphers discussed below are word-based, where one word consists of 32 bits. On a 32-bit value, we denote by \oplus the bitwise addition in $\mathbb{F}_{2^{32}}$, by \boxplus the addition modulo 2^{32} . The notations $\ll n$ and $\gg n$ define left and right shifts by n bits (modulo 2^{32}), and $\lll n$ the left rotation by n bits. For any vector $x = (x_0, \dots, x_{n-1})$ in \mathbb{F}_2^n and integers $0 \leq a < b < n$ we denote by $x^{(a, \dots, b)}$ the vector (x_a, \dots, x_b) .

2 Cache Timing Model

As with all side-channel attacks, cache timing attacks are not inherently attacks against the algorithm, but against its implementation¹. Thus, there are basically two ways of analyzing the cache timing resistance of a cipher. One can either consider a concrete implementation of the cipher, or do a general analysis in the framework of a model that gives the adversary certain rights, which can be modeled as oracle accesses. In the latter case, a “break” within the model does not necessarily imply a break of all practical implementations, but it can indicate that extra care has to be taken when implementing the cipher.

The model that we want to use for our analysis is the one proposed in [26]. It models a synchronous cache attacker, i.e. an adversary who can only access (and thus perform measurements on) the cache after certain elementary operations by the legitimate users have finished. In particular, a synchronous cache adversary can do cache measurements before and after a full update of the stream cipher’s inner state, but not while the update is in progress.

¹ For readers not familiar with cache timing attacks, Appendix A gives a short introduction.

Formally, the adversary uses two oracles:

- **KEYSTREAM(i):** He requests the cipher to return the i -th keystream block to him.
- **SCA_KEYSTREAM(i):** He obtains a noise-free list of all cache accesses made by KEYSTREAM(i), but does not learn anything about their order. These cache accesses give him information about the actual table entries as described in the paragraph on “Handling Cache Line Sizes” below.

Model vs. Real World: Note that the KEYSTREAM(i) oracle is considered “standard” for stream cipher analysis, i.e. it is also available to an adversary in a non-side-channel setting. The SCA_KEYSTREAM(i) oracle, on the other hand, gives the adversary more information than will typically be available in a real-world side-channel setting, since it assumes that his measurements are undisturbed by noise. Thus, the results obtained in this paper are for an idealized cache measurement setting. Whether attacks under this model also constitute attacks in the real world depends on the implementation details and has to be verified on a case-by-case basis.

In the real world, the adversary usually needs the ability to repeat his measurements several times in order to remove the noise from the list of cache accesses. This corresponds to running the stream cipher under the same key and initialization vector (IV) for a given number of times, each time measuring the cache lines accessed and intersecting the resulting tables. Note that while encrypting different plaintexts under the same IV should not be possible for most implementations, one can imagine applications where the adversary can ask the system to decrypt the *same* ciphertext several times under the same IV, thus forcing the stream cipher to execute an identical sequence of operations.

Also note that there may be “wrong” cache accesses that occur very frequently, i.e. that do not disappear when repeating the measurement and intersecting several cache access lists. These accesses may originate with external processes such as applications or the operating system. In some cases, it may be possible to identify these wrong accesses by repeatedly running the stream cipher for a number *different* IVs – the cache accesses that stay constant are the ones that are independent of the cipher and can thus be ignored.

In all cases, if the noise can not be eliminated completely, discarding the set of measurements is always an option. As will be described in Section 3, our attack technique already works if noise-free measurements are possible only once in a while.

Handling Cache Line Sizes: In theory, addresses of cache lines translate into information about the table indices used. However, in real-world cache timing attacks, a cache line can hold several table entries, i.e. a cache line represents several table indices. Thus, even if the adversary could do noise-free measurements, he would still not learn the *exact* table indices used, but only obtain b bits of *partial* information about the table index.

Consider the case of the popular Pentium 4 processors as an example. All LFSR lookup tables used by the ciphers in this paper contain 256 32-bit entries,

i.e. each table entry fills 4 bytes. A cache line in a Pentium 4 processor is 64 bytes broad, meaning that it can contain 16 table entries. Thus, given the correct cache line, the adversary learns only a subset of 16 table entries which contains the right one. Since these table entries are typically aligned², this corresponds to learning the $b = 4$ most significant bits of the table index, while the adversary obtains no information whatsoever about the 4 least significant ones.

More generally, if a table contains 2^c entries of d bytes each, and if the processor has a cache line size of λ bytes, then each cache line will hold λ/d table entries. Thus, even in a completely noise-free scenario, the attacker can not reconstruct more than the uppermost $b = c - \log_2(\lambda/d)$ bits of the table index (i.e. c bits in the best and 0 bits in the worst case).

3 General Framework of the Attack

In this section we present the general framework of our cache timing attack. All ciphers that make use of a LFSR for which clocking the LFSR involves table lookups are covered by this framework. This includes the stream ciphers Sosemanuk, Snow, Sober and Turing, which we will analyze separately below.

3.1 Basic Idea

Observing inner state bits: Given a stream cipher with an LFSR of length n defined over \mathbb{F}_{2^m} we denote by

$$s = (s_0, \dots, s_{n-1}) \in \mathbb{F}_{2^m}^n$$

the initial state of the LFSR after initialization and by

$$(s_t, \dots, s_{t+n-1}) \in \mathbb{F}_{2^m}^n$$

its internal state at time t . Our model assumes that for clocking the LFSR the implementation makes use of table lookups. This is the case for almost all stream ciphers using an LFSR defined over \mathbb{F}_{2^m} , the reason being that this usually is the fastest manner to implement multiplication by one fixed element in \mathbb{F}_{2^m} . At time t this table lookup uses some bits of one of the elements s_{t+i} for $0 \leq i < n$ (in the case of Sosemanuk, Snow, Sober and Turing it involves 8 bits). Depending on the cache line size (cf. the discussion above) the cache timing measurements will reveal b of those bits. The trivial, but important, observation is that all those bits can be expressed as linear combinations of bits of the initial state s . Moreover, computing the actual linear combination can easily be done as follows.

Transforming into initial state bits: Elements in \mathbb{F}_{2^m} can be identified with m -bit words (i.e. elements of \mathbb{F}_2^m) via a vectorspace isomorphism

$$\psi : \mathbb{F}_{2^m} \rightarrow \mathbb{F}_2^m.$$

² If they aren't, the attack gets easier, since certain lines leak more information than assumed here.

Using the isomorphism ψ we can consider the state of the LFSR as an element in \mathbb{F}_2^{nm} instead of an element of $\mathbb{F}_{2^m}^n$ via

$$(s_t, \dots, s_{t+n-1}) \rightarrow (\psi(s_t), \dots, \psi(s_{t+n-1})).$$

Then, clocking the LFSR can be described by applying an invertible $nm \times nm$ matrix over \mathbb{F}_2 to the current state. This is true simply because updating the LFSR is certainly a \mathbb{F}_{2^m} -linear operation and therefore in particular \mathbb{F}_2 -linear. We denote the matrix that corresponds to updating the LFSR by M . Moreover, the matrix M can be easily computed given the feedback polynomial of the LFSR.

Each table lookup reveals some bits of s_{t+i} , i.e. $\psi(s_{t+i})^{(a..a+b-1)}$ for some $a \in \{0, \dots, m-b-1\}$. Writing $\psi(s_{t+i})$ as $M^t \psi(s)$ we see that

$$\psi(s_{t+i})^{(a..a+b-1)} = [M^t \psi(s)]^{(a'..a'+b-1)}$$

for some $a' \in \{0, \dots, nm-b-1\}$ and thus linear in the bits of the initial state as claimed. In this way, each bit observed in a cache timing measurement yields a linear equation in the initial state bits, and once sufficiently many equations have been collected, the initial state can be retrieved by solving the equation system.

3.2 Number of Required Noise-Free Measurements

Practical problems: It remains to discuss the number of measurements that are required to obtain a solvable equation systems. We start by pointing out that in practice, we can not expect to obtain all cache measurements that seem possible in theory. The following problems can occur, but as it turns out, they can be overcome using the linearity of the above equations:

1. If the cipher clocks the LFSR several (say c) times for each call to $\text{KEYSTREAM}(i)$, or if the table is accessed several times for each clock, then a synchronous adversary can not measure each single table access separately. Instead, he learns c indices at a time (see e.g. Sosemanuk, where $c=4$). This implies that the attacker gets to know the values

$$\psi(s_{t+i})^{(a..a+b-1)}, \dots, \psi(s_{t+i+c-1})^{(a..a+b-1)}$$

but not the order of those.

This problem can be dealt with by forming a linear equation using the sum of all the observed values. This way, the adversary obtains only 1 observation (instead of c) for each round, but otherwise, there is no effect on the overall effort of the attack.

2. Also in the case where a table is accessed several times for each call to $\text{KEYSTREAM}(i)$, there is the problem of collisions. If two or more table accesses use the same cache line, then above trick no longer works. For example, if he measures accesses to cache lines L_1, L_2, L_3 for Sosemanuk (4 accesses

per call), then he does not know which cache line information he has to use twice. Guessing is usually not a good strategy, since it rapidly increases the overall work effort for the attack. Instead, if such collisions are not too frequent, simply discarding the measurements and trying again next round gives much better results.

3. As described in Section 2, the number of bits available for analysis depends on the cache line size. In particular, for processors with large cache lines, the information available decreases. Nonetheless, as long as at least one bit of information leaks, the attack still works as described above.

Number of Noise-Free Measurements: Assume we are attacking an LFSR with an internal state of n elements of \mathbb{F}_{2^m} and the cache measurement reveals only a fixed linear combination of internal state bits at each k -th iteration. In this case, nmk iterations suffice to completely reveal the initial state of the LFSR. The easiest way to see that is to interpret the initial state s of the LFSR as an element of $\mathbb{F}_{2^{nm}}$ (see for example [12, Chapter 8] for details). Clocking the LFSR corresponds to multiplying the internal state by a fixed element $\alpha \in \mathbb{F}_{2^{nm}}$ and the information leaking at time $T = tk$ can be written as $u_T = \text{Tr}(\theta(\alpha^k)^t s)$ for an fixed element $\theta \in \mathbb{F}_{2^{nm}}$ depending on the linear combination of bits that are leaked. Here Tr denotes the trace function

$$\begin{aligned} \text{Tr} : \mathbb{F}_{2^{nm}} &\rightarrow \mathbb{F}_2 \\ \text{Tr}(x) &= \sum_{i=0}^{nm-1} x^{2^i}. \end{aligned}$$

The usual requirement that the LFSR should have maximal period corresponds to α being a primitive element in $\mathbb{F}_{2^{nm}}$. Thus, for reasonably small k , the element α^k is not in a proper subfield of $\mathbb{F}_{2^{nm}}$ and

$$(\alpha^k)^0, (\alpha^k)^1, \dots, (\alpha^k)^{nm-1}$$

form an \mathbb{F}_2 basis of $\mathbb{F}_{2^{nm}}$. Therefore, all the linear equations $u_T = \text{Tr}(\theta(\alpha^k)^t s)$ for $T = tk$, $t \leq nm - 1$ are linearly independent and the initial state can be uniquely recovered by solving the corresponding system of linear equations.

Note that this result only holds if all known keystream bits are exactly equidistant. For the ciphers discussed in this paper, this effect can be achieved by measuring only 1 bit for each clocking of the LFSR (where in principle, we could measure b or even $2b$ bits). If, however, some measurements have to be discarded due to collisions or noise, then the result can not be applied. Nonetheless, it is still very likely that we need only a very small overhead of noise-free measurements to get a uniquely solvable system. Namely, under the assumption that it behaves like a random system of linear equations, the probability that after $nm + \delta$ noise-free measurements the resulting system has full rank is given by

$$p = \prod_{j=0}^{nm-1} \frac{2^{nm+\delta} - 2^j}{2^{nm+\delta}} \approx 1 - 2^{-\delta}.$$

For example, when using the Sosemanuk parameters $n = 10$ and $m = 32$, the equation system will have rank nm with probability > 0.969 after only $\delta = 5$ additional noise-free measurements.

4 Analyzing Specific Ciphers

In this section we apply our general framework to the ciphers Sosemanuk, Snow, Sober and Turing. All ciphers are described briefly, giving only the details required to understand the attack.

For all those ciphers it turns out that once the internal state of the LFSR is recovered, the remaining bits of the internal state (if any) can be determined very efficiently. We explain the attack on Sosemanuk in detail; the other ciphers are discussed much shorter as the general approach is always the same.

In order to simplify notation in this section, we do not explicitly write the isomorphism operator ψ but assume that the reader is aware of whether a given vector is in \mathbb{F}_{2^m} or in \mathbb{F}_2^m .

4.1 Analysis of Sosemanuk

The Sosemanuk cipher was proposed by Berbain et al. in 2005 as an eStream candidate [1]. Sosemanuk consists of a 10-word LFSR over $\mathbb{F}_{2^{32}}$, a finite-state machine (FSM) with two 32-bit words, and an output function combining LFSR and FSM output into the keystream. No valid attacks against the cipher have been proposed so far.

LFSR: The linear recursion of the LFSR is defined by

$$s_{t+10} = s_{t+9} \oplus \alpha^{-1} s_{t+3} \oplus \alpha s_t,$$

where α is a fixed element in $\mathbb{F}_{2^{32}}$. An optimized implementation of the multiplications by α and α^{-1} uses 8x32-bit lookup tables T_1 and T_2 . Ignoring the isomorphism operator ψ , the multiplications can be implemented as follows:

$$\begin{aligned} x \cdot \alpha &= \left((x \ll 8) \oplus T_1[x^{(24..31)}] \right) \\ x \cdot \alpha^{-1} &= \left((x \gg 8) \oplus T_2[x^{(0..7)}] \right). \end{aligned}$$

FSM: In addition, we need a simplified description of the FSM. If we denote the two 32-bit state words by $R1$ and $R2$ and the inner state words of the LFSR by s_t as above, we can describe the production of an intermediate value f_t as follows:

$$\begin{aligned} R1_t &= \text{Update1}(R1_{t-1}, R2_{t-1}, s_{t+1}, s_{t+8}) \\ R2_t &= \text{Update2}(R1_{t-1}) \\ f_t &= (s_{t+9} \boxplus R1_t) \oplus R2_t \end{aligned}$$

We don't need the internals of the functions *Update1* and *Update2* for the analysis.

Output: Before producing output, Sosemanuk will clock the LFSR 4 times and generate four intermediate values. Then the keystream output is generated as

$$(z_{t+3}, z_{t+2}, z_{t+1}, z_t) = \text{Serpent1}(f_{t+3}, f_{t+2}, f_{t+1}, f_t) \oplus (s_{t+3}, s_{t+2}, s_{t+1}, s_t),$$

where we don't have to know more about the *Serpent1* function than that it is a permutation that is easily invertible.

Cache Timing Attack

Cache Measurements: Assume for simplicity that $b = 8$, i.e. that the adversary observes all 8 bits of the table indices for T_1 and T_2 after any output block of his choice. This means that he obtains 4 measurements of accesses to T_1 and T_2 each. In the case of T_1 , he knows that they correspond to inner state bytes

$$s_t^{(24..31)}, s_{t+1}^{(24..31)}, s_{t+2}^{(24..31)}, s_{t+3}^{(24..31)},$$

but he does not know the correct order. The same also holds for the accesses to T_2 , which give information about

$$s_t^{(0..7)}, s_{t+1}^{(0..7)}, s_{t+2}^{(0..7)}, s_{t+3}^{(0..7)}$$

without revealing the proper ordering.

Reconstructing the LFSR State: The attacker knows at each time $T = 4t$ the values

$$u_T = s_t^{(24..31)} \oplus s_{t+1}^{(24..31)} \oplus s_{t+2}^{(24..31)} \oplus s_{t+3}^{(24..31)} \quad (1)$$

and

$$v_T = s_t^{(0..7)} \oplus s_{t+1}^{(0..7)} \oplus s_{t+2}^{(0..7)} \oplus s_{t+3}^{(0..7)}, \quad (2)$$

as those sums do not depend on the ordering anymore. As both Eq. 1 and 2 yield $b = 8$ linear equations over \mathbb{F}_2 , we get a total of 16 linear equations for the initial state s at each time T . It turns out that after 20 LFSR clockings the resulting 320 equations already have full rank and therefore the initial state can easily be computed given u_T, \dots, u_{T+19} and v_T, \dots, v_{T+19} . Note that if $b < 8$, then the number of LFSR clockings whose timings have to be observed is $320/2b = 160/b$.

Reconstructing the FSM State: Given the correct inner state of the LFSR, the inner state of the FSM is easily reconstructed. To this end, we proceed as follows:

1. Given output words z_t, \dots, z_{t+3} , the adversary can subtract s_t, \dots, s_{t+3} and obtains the output of the *Serpent1* S-box.
2. The S-box is invertible, yielding the values f_t, \dots, f_{t+3} .
3. The adversary guesses the state of $R1_t$ (32 bits), which allows him to compute the state of $R2_t$ from the equation $f_t = (s_{t+9} \boxplus R1_t) \oplus R2_t$.
4. The adversary updates the state of the FSM once to obtain $R1_{t+1}$ and $R2_{t+1}$. He checks whether the output matches the observed f_{t+1} .

Normally, after this step, only the correct guess for $R1_t$ should have survived. If more guesses survive, one simply continues updating the inner state and checking against the output, until the state $(R1, R2)$ is uniquely determined. In total, this step requires not more than 2^{32} simple guess-and-determine steps. Note that algorithmically more elegant ways of reconstructing the FSM state might exist, but since 2^{32} guess-and-determine steps are easily computable on e.g. a standard PC, we did not search for such attacks.

Resources: The overall effort for the attack is dominated by reconstructing the FSM state and therefore has an overall complexity of 2^{32} guess-and-determine steps, and the memory consumption is dominated by the space for storing the 320×320 -bit equation system. Thus, assuming the availability of noise-free timing observations for $\approx 160/b$ LFSR clockings, we have an efficient cache timing attack against Sosemanuk.

4.2 Analysis of Snow 2.0

Snow 2.0 was proposed by Ekdahl and Johansson in 2002 [9]. It uses an LFSR over $\mathbb{F}_{2^{32}}$ of length 16, an FSM with two 32-bit words, and an output function combining LFSR and FSM output into the keystream. No valid attacks against the cipher have been proposed so far.

LFSR: The linear recursion of the LFSR is defined by

$$s_{t+16} = \alpha^{-1} s_{t+11} \oplus s_{t+2} \oplus \alpha s_t,$$

where α is a fixed element in $\mathbb{F}_{2^{32}}$. Just as for Sosemanuk, two 8×32 -tables T_1 and T_2 are involved for the multiplication by α and α^{-1} . Each of them is called exactly once for each state update.

FSM: Snow uses a FSM consisting of two 32-bit words $R1_t$ and $R2_t$. The exact specification is not important for our attack. It suffices to know that given the internal state, the output stream and one of the two words of the FSM the remaining word is uniquely determined.

Cache Timing Attack

Cache Measurements: Assuming precise measurements, the adversary obtains the uppermost b bits of the table indices for T_1 and T_2 . In each round, he obtains one measurement for T_1 and T_2 each. In the case of T_1 , he knows that they match to inner state bytes $s_t^{(31-b+1..31)}$ and in the case of T_2 the attacker learns $s_t^{(7-b+1..7)}$. As both observations yield b linear equations over \mathbb{F}_2 , we get a total of $2b$ linear equations for the initial state s for each LFSR clocking. Thus, after approximately $512/2b = 256/b$ LFSR clockings, the equation system can be solved.

Table 1. Attack parameters against various ciphers

	Size of eq. system	Guessing Steps	# Cache Measurements		Known Keystream
			General	Pentium 4	
Sosemanuk	320	2^{32}	160/b clks	40 clks	16 bytes
Snow 2.0	512	2^{32}	256/b clks	64 clks	8 bytes
Sober-128	544	-	544/b clks	136 clks	4 bytes
Turing	544	-	544/b clks	136 clks	-

Reconstructing the FSM State: As mentioned above, the inner state of the FSM consists of two words. As for Sosemanuk, by guessing one word (e.g. $R1_t$), the adversary can derive the other. He can then update the inner state and verify whether his guess was correct. Again, the expected workload is not more than 2^{32} simple guess-and-determine steps.

4.3 Analysis of Sober-128

Sober-128 was proposed by Hawkes and Rose in 2003 [11]. It makes use of an LFSR over $\mathbb{F}_{2^{32}}$ of length 17, a key dependent 32-bit constant K , and a nonlinear output function combining the LFSR and the constant K into the keystream.

LFSR: The linear recursion of the LFSR is defined by

$$s_{t+17} = s_{t+15} \oplus s_{t+4} \oplus \alpha s_t,$$

where α is a fixed element in $\mathbb{F}_{2^{32}}$. Just as for Sosemanuk and Snow 2.0, multiplication by α is done using a lookup table.

Cache Timing Attack. Assuming precise measurements, the adversary observes b bits for each round, i.e. $s_t^{(7-b+1..7)}$. Thus, after $\approx 544/b$ rounds, we expect to be able to reconstruct the LFSR state. Given this state, the constant K can trivially be computed given one keystream word.

4.4 Analysis of Turing

Turing was introduced by Rose and Hawkes in 2002 [21]. It is based on the same LFSR as Sober-128 and uses a fixed non-linear filter function on the internal state of the LFSR to generate the keystream. In particular the cache timing part is exactly the same as for Sober-128. As the internal state of Turing consists only of the LFSR state, no additional bits have to be recovered and in particular no keystream bits are needed to perform the attack.

5 Conclusions

We have shown how to mount cache timing attacks against all word-based LFSR implementations that use lookup tables to speed up multiplications. As described

above such ciphers are especially vulnerable to cache timing attacks. All of them we are aware of can be broken very efficiently within our theoretical model.

What is more, our attack is tolerant with respect to noisy measurements: the information delivered by the cache timings may be few (1 bit once in a while is enough) and far between (the distances between the bits can be arbitrary). This is due to the fact that a noise-free measurement always reveals a linear equation for the internal state. Thus, as long as we can detect errors, we can simply discard noisy measurements without significantly increasing the complexity of the equation system to be solved.

Clearly, implementing cache timing attacks on real life systems is a difficult and cumbersome task that requires dedicated skills. However, due to the reasons outlined above, we anticipate that it is significantly easier to implement our attack on the above mentioned stream ciphers than most other cache timing attacks.

Countermeasures. A possible countermeasure is to split the lookup table into several, smaller tables such that each table fits into one cache line. Here is a short example using C notation. Suppose `LinTab[256]` is a lookup table with 32-bit entries and our processor has cache lines of 64 bytes, so each cache line can contain at most 16 table entries. The linearity of the table can be exploited to compute `LinTab[x]` using two smaller tables `LinTabUpper[16]` and `LinTabLower[16]` with entries defined as:

```
for(y=0; y<16; y++) LinTabUpper[ y ] = LinTab[ y << 4 ];
for(y=0; y<16; y++) LinTabLower[ y ] = LinTab[ y ];
```

The linearity of `LinTab[]` means that we can generate the value of `LinTab[x]` at compute time using five operations:

```
LinTabUpper[ x >> 4 ] ^ LinTabLower[ x & 0xF ];
```

This way, no information about the cache entries can be obtained by timing measurements because `LinTabUpper[]` and `LinTabLower[]` each fit within one cache line. It should be noted that there is a performance penalty since the one table-lookup operation is replaced by five operations, and this performance penalty might be prohibitively large for the speed-sensitive stream ciphers.

References

1. Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: SOSEMANUK, a fast software-oriented stream cipher. eStream submission (2005), <http://www.ecrypt.eu.org/stream/sosemanuk.html>
2. Bernstein, D.: Cache timing attacks on AES (2005), <http://cr.yp.to/papers.html#cachetiming>
3. Bertoni, G., Zaccaria, V., Breveglieri, L., Monchiero, M., Palermo, G.: AES power attack based on induced cache miss and countermeasure. In: International Symposium on Information Technology: Coding and Computing (ITCC 2005), vol. 1, pp. 586–591. IEEE Computer Society, Los Alamitos (2005)

4. Blömer, J., Krummel, V.: Analysis of countermeasures against access driven cache attacks on AES. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 96–109. Springer, Heidelberg (2007)
5. Boneau, J., Mironov, I.: Cache-collision timing attacks against AES. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 201–215. Springer, Heidelberg (2006)
6. Brickell, E., Graunke, G., Neve, M., Seifert, S.: Software mitigations to hedge AES against cache-based software side-channel vulnerabilities (2006), <http://eprint.iacr.org/2006/052.pdf>
7. Canteaut, A., Lauradoux, C., Seznec, A.: Understanding cache attacks. Technical Report 5881, INRIA (2006)
8. Ekdahl, P., Johansson, T.: SNOW - a new stream cipher. NESSIE project submission, <http://www.it.lth.se/cryptology/snow/>
9. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 47–61. Springer, Heidelberg (2003)
10. Gierlichs, B., Batina, L., Clavier, C., Eisenbarth, T., Gouget, A., Handschuh, H., Kasper, T., Lemke-Rust, K., Mangard, S., Moradi, A., Oswald, E.: Susceptibility of eSTREAM candidates towards side channel analysis. In: de Cannière, C., Dunkelmann, O. (eds.) SASC 2008 Workshop Record, pp. 123–150 (2008)
11. Hawkes, P., Rose, G.: Primitive specification for Sober-128, <http://www.qualcomm.com.au/Sober128.html>
12. Lidl, R., Niederreiter, H.: Finite Fields. Cambridge University Press, Cambridge (1997)
13. Neve, M., Seifert, J.: Advances on access-driven cache attacks on AES. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 147–162. Springer, Heidelberg (2007)
14. Neve, M., Seifert, J., Wang, Z.: Cache time-behavior analysis on AES (2006), http://www.cryptologie.be/document/Publications/AsiaCSS_full_06.pdf
15. Neve, M., Seifert, J., Wang, Z.: A refined look at bernstein's AES side-channel analysis. In: Proc. AsiaCSS 2006, p. 369. ACM, New York (2006)
16. O'Hanlon, M., Tonge, A.: Investigation of cache-timing attacks on AES (2005), <http://wwwcomputing.dcu.ie/research/papers/2005/0105.pdf>
17. Osvik, D., Shamir, A., Tromer, E.: Cache attacks and countermeasures: The case of AES (2005), <http://eprint.iacr.org/2005/271.pdf>
18. Osvik, D., Shamir, A., Tromer, E.: Cache attacks and countermeasures: The case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)
19. Page, D.: Theoretical use of cache memory as a cryptanalytic side-channel. Technical Report CSTR-02-003, University of Bristol (June 2002), http://www.cs.bris.ac.uk/Publications/pub_info.jsp?id=1000625
20. Percival, C.: Cache missing for fun and profit. Paper accompanying a talk at BSDCan 2005 (2005), <http://www.daemonology.net/papers/htt.pdf>
21. Rose, G., Hawkes, P.: Turing: A fast stream cipher. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 290–306. Springer, Heidelberg (2003)
22. Salembier, R.: Analysis of cache timing attacks against AES. Scholarly Paper, ECE Department, George Mason University, Virginia (May 2006), http://ece.gmu.edu/courses/ECE746/project/F06_Project_resources/Salembier_Cache_Timing_Attack.pdf

23. Tsunoo, Y., Saito, T., Suzaki, T., Shigeri, M., Miayuchi, H.: Cryptanalysis of DES implemented on computers with cache. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 62–76. Springer, Heidelberg (2003)
24. Tsunoo, Y., Tsujihara, E., Minematsu, K., Miayuchi, H.: Cryptanalysis of block ciphers implemented on computers with cache. In: Proc. ISITA 2002 (2002)
25. Wang, Z., Lee, R.: New cache designs for thwarting software cache-based side channel attacks. In: Proc. ISCA 2007, June 2007, pp. 494–505. ACM, New York (2007)
26. Zenner, E.: A cache timing analysis of HC-256. In: Avanzi, R., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 199–213. Springer, Heidelberg (2009)

A Cache Timing Attack Basics

In the following, we give a short introduction to cache timing attacks. For more detailed information, the reader is referred to the introductory papers by Bernstein [2] and Osvik, Shamir, and Tromer [17,18].

Cache motivation: Modern processors store data in different types of storage. Data that is currently being processed is stored in the so-called registers; however, only few of these are available. Instead, reasonably large amounts of data (such as look-up tables or S-boxes) are stored in RAM. Since access to RAM is relatively slow compared to executing an arithmetic operation, frequently used data is also stored in an intermediate type of memory, the so-called cache.

Cache workings: The CPU cache of modern processors is organised into blocks – so-called *lines* – of λ bytes. Correspondingly, RAM is considered to be (virtually) divided into λ -byte lines. When loading data from RAM into a CPU register, the system first checks whether the corresponding RAM line already lies in cache. If yes, it is loaded directly from cache, which is very fast. If not, it is first loaded from RAM to cache, which takes longer. Mapping from RAM to cache is typically by a simple modulo operation, i.e. if the cache can hold n lines and if the data lies in RAM line a , then it is loaded into cache block $a \bmod n$. This means that neighbouring data in RAM (e.g. tables) stays clustered in cache.

A simple attack: As an example, consider the **prime-then-probe** method presented in [18]. The adversary starts by filling all the cache with his own data. Then the legitimate user U gets the read/write token. U loads the data required for his own computations into cache, where it evicts the adversary's data. When the adversary reobtains the read/write token, he tries to reload his own data from cache. For each cache line, if this takes long, it means that U has evicted the corresponding data.

From this analysis, the adversary obtains a profile of cache blocks that have been used by U . This profile is a noisy version of the cache lines that have been used for the encryption. By repeating the experiment a number of times, a good approximation of the real cache access profile can be obtained.

Note that the adversary does not learn the *data* that was written in the cache by U – he learns something about the *addresses* of the data that was used. In the case of an LFSR, this corresponds to the indices of the LFSR cells that were accessed.

Optimal Recovery of Secret Keys from Weak Side Channel Traces

Werner Schindler¹ and Colin D. Walter²

¹ Bundesamt für Sicherheit in der Informationstechnik (BSI)

Godesberger Allee 185-189, 53175 Bonn, Germany

Werner.Schindler@bsi.bund.de

² Royal Holloway, University of London,

Egham, Surrey, TW20 0EX, United Kingdom

Colin.Walter@rhul.ac.uk

Abstract. It should be difficult to extract secret keys using weak side channel leakage from embedded crypto-systems which employ standard counter-measures. Here we consider the case of key re-use with randomised exponent recoding. An optimum strategy is presented and proved, but it has the disadvantage of impracticality for realistic key sizes. Developed from the basis of an optimal decision strategy, some modified, computationally feasible versions are studied for effectiveness. This shows how to modify existing algorithms and pick their parameters for the best results.

Keywords: Side channel leakage, power analysis, optimal strategy.

1 Introduction

The academic parents of this work are the optimal strategies of Schindler [10] and the algorithms of Walter [13] for deducing secret keys using weak side channel leakage from an exponentially based public key crypto-system in which the key is re-used a number of times in some form. The optimal decision strategy is practically feasible if the key can be guessed in small parts as in [10], for instance, while under the present conditions it remains feasible only for artificially small key sizes. However, the algorithms discussed in [13] still remain feasible for typical cryptographic key sizes. Here a comparison of the two approaches leads to i) the determination of the best parameters to choose in existing computationally feasible algorithms and ii) the identification of points in the development of the algorithm where it seems impossible to derive a computationally feasible method from the optimal algorithm. Although standard hardware counter-measures nowadays make side channels extremely weak in embedded systems, the methods here were used in simulations to recover keys using substantially weaker leakage than has been reported in the past.

The earliest published work on such secret key recovery is that of Kocher *et al.* [6,7]. For RSA and similar crypto-systems, (unprotected) classical exponentiation algorithms employ the same sequence of multiplicative operations every

time the key is re-used, and this allows leakage (in a certain sense) to be averaged over many traces to guess the key in small portions, such as bit by bit. With enough data, the operation types can be determined as squarings or multiplications, and this is sufficient to yield the key for the binary exponentiation algorithm. Randomised recoding of the exponent causes operations corresponding to the same key bit to be mis-aligned and variable for different uses of the key. A number of such re-codings exist [9,8,4,15] and they were generally believed to lead to much more secure systems – attacks on such re-codings seemed to require significantly stronger side channel leakage to succeed than is the case where Kocher’s attack applies. However, this no longer seems to be the case. Karlof & Wagner [5], Green *et al.* [3] and Walter [13] provide increasingly robust details for attacking such systems without using such strong leakage, thereby emphasising the need to combine a number of counter-measures rather than relying on just one or two to defeat the opponent. None of these works provides justification for the efficacy of their algorithms. Thus there is a gap between what has been achieved at a computationally feasible level and what has been derived theoretically. Here we describe a search to bridge that gap by explaining the computationally efficient choices in terms of the optimal strategies described by Schindler in [10]. As a result, much more powerful means of exploiting the leakage are now identified.

2 The Leakage Model

The context of the side channel attack is the repeated use of a randomised exponentiation algorithm for computing C^K in any cryptographic group where K is a fixed secret key which is not blinded by a random multiple of the group order¹, and C is an unknown ciphertext (or unknown plaintext) which may vary and may be blinded². Of course, all the following considerations apply equally well to randomised scalar multiplications in additive groups (as in ECC).

The adversary is assumed to know all the details of the exponentiation algorithm. Use of the key provides him with a side channel trace for the exponentiation itself, but no further information is assumed: in particular, he is not expected to be able to choose or see any direct input to the exponentiation, nor view any output, nor usefully observe any pre- or post-processing.

It is assumed that occurrences of multiplicative operations in the exponentiation can be identified accurately from the corresponding side channel trace, but that their identities as squares or multiplications (and, in the case of methods with pre-computed tables, multiplications by particular table entries) can only be determined with a substantial degree of inaccuracy [1]. The adversary’s aim is to discover K using computationally feasible resources. The multiplicative operations are assigned probabilities that they represent squares or multiplications as a result of previous experience by the adversary. For this he uses knowledge

¹ Another standard counter-measure to Kocher’s averaging of side channel traces.

² The base in the exponentiation is frequently unknown due, for example, to “Rivest” blinding [2] or because of an unknown modular reduction when applying the CRT.

of the stochastic behaviour of the operations in the side channel, and the extent to which this behaviour varies.

In order to model noisy measurement data in the simulations we assumed that these probabilities were distributed binomially and independently for all multiplications with mean probabilities depending on the true type of the operation, so that some were known correctly with high probability, but most were known with little confidence. However, Theorem 1 also covers more general leakage scenarios.

3 The Randomised Exponentiation

Examples of the randomised exponentiation algorithms which can be attacked in the way described here include those of Liardet-Smart [8], Oswald-Aigner [9] and Ha-Moon [4,15]. Their common, underlying basis is a recoding of the binary representation of the key K into a form

$$R = ((\dots(r_{m-1}2^{m_{m-2}} + r_{m-2})2^{m_{m-3}} + \dots + r_2)2^{m_1} + r_1)2^{m_0} + r_0$$

for digits r_i and exponents m_i in some fixed, pre-determined sets $\mathcal{D} \subseteq \mathbb{Z}$ (which, for convenience, contains 0³) and $\mathcal{M} \subseteq \mathbb{N}^+$ respectively. In this *recoding*, both r_i and m_i are selected according to some finite state automaton (FA) which has the bits of K and the output from a random number generator (RNG) as inputs. For convenience, we assume the bits of K are consumed by the FA from least to most significant. Different bit streams from the RNG result in different recodings R of K .

The exponentiation C^R begins with the pre-calculation of the table $\{C^d \mid d \in \mathcal{D}, d \neq 0\}$. Then for $i = m-2, m-3, \dots, 2, 1, 0$ the main iterative step of the exponentiation consists of m_i squarings followed by a multiplication by the table entry C^{r_i} when $r_i \neq 0$. This results in a sequence of multiplicative operations which is most easily presented using r_i to denote multiplication by C^{r_i} and m_i copies of 0 to denote the m_i squarings. We call this the *recoding sequence* for R , and it belongs to \mathcal{D}^* . For example, the exponent $K = 13_{10} = 1101_2$ may have a recoding $R = (1.2^2+3)2^1+\bar{1}$ which gives the operation sequence 100301, the recoding sequence associated with the recoding R . For convenience (e.g. in Section 7), the leading recoded digit is translated in the same way as the others into multiplicative operations of the recoding sequence even when that digit is 0; alternatives in processing the leading bits are ignored. (This matches the situation where the exponentiation algorithm begins with 1 instead of C .) However, the leading digits are invariably treated differently in practice, and appropriate modifications need to be made in the methods here to handle them properly.

Thus, we are using the same set \mathcal{D} to represent both the set of recoding digits and the set of corresponding recoding operations. In our examples, we will make the distinction clearer by using, for example, ‘S’ for the squaring operation and

³ Recodings which do not allow 0 in representations are not excluded here, but we want to include it for another use, namely to represent a squaring operation.

reserving ‘0’ for the digit. We will rarely be working with recodings. It will be much more frequently be with recoding sequences.

The exponentiation algorithms of interest are assumed to have the property that perfect knowledge of the multiplication/squaring sequences (without necessarily knowledge of the choice of respective table entries in the case of multiplications) for a small number of recodings R of K yields enough information to reconstruct the secret key K with at most a small number of ambiguities. This is the case for the algorithms listed above: attacks on them using such information are described in [11], [12] and [15] respectively. The theory here, however, allows for the possibility of distinguishing between the use of different table entries in the multiplications, and this enables one to deal with recodings for which only non-zero digits are used.

4 The Optimal Decision Strategy

The optimal decision strategy for discovering the secret key K begins by identifying a key K^* with the highest probability of having generated the observed side channel leakage. If the most likely key candidate fails the attacker tries the key candidates that are ranked 2, 3, This maximises the use of known information about K and hence minimises the effort in searching for K .

For each use of the secret key K , the side channel leakage leads to a best guess G at the recoding that was used: the locations of multiplicative operations are identified, and the most likely digit values are selected for those operations. Associated with a set of these recoding guesses there are optimal choices K^* for the key value – those which maximise how well the key collectively matches the guessed recodings.

Remark 1. To be successful in real-world attacks we must assume there is enough leakage to ensure that the correct key is among the best, i.e. most probable, fits to the recoding guesses for otherwise it will be computationally infeasible to find it. If not correct, the most plausible keys typically are at least related to the correct one, which means that their bit representations or their recodings are similar to that of the correct key in some sense. E.g., long sequences of bits in these most probable keys may either be identical to those in the correct key or, depending on the recoding scheme, related to them in very specific, predictable ways⁴. Hence the errors in using a most probable key to predict the correct key should also be relatively few in number, generally isolated, and effectively independent. Consequently, virtually all errors will be equally easy to correct although finding them may not be so easy.

Definition 1

(i) Let $\mathcal{K} \subseteq \mathbb{F}_2^*$ denote the set of all admissible keys in binary representation and $\mathcal{R} \subseteq \mathcal{D}^*$ the set of all possible recoding sequences of keys for the chosen recoding

⁴ E.g. two keys which are bit-wise complements of each other can have almost identical recodings, as may two keys of which one is almost the same as a shift of the other.

scheme. Recoding to an operation sequence is defined by a map $\phi: \mathcal{K} \times \mathcal{Z} \rightarrow \mathcal{R}$ where $z \in \mathcal{Z}$ denotes a random number in a finite set \mathcal{Z} . The set $\phi(K, \mathcal{Z}) \subseteq \mathcal{R}$ of possible recoding sequences of K is denoted $\mathcal{R}(K)$ or \mathcal{R}_K .

(ii) The set of possible recoding sequence guesses which can be deduced from side channel leakage is denoted by \mathcal{G} where $\mathcal{R} \subseteq \mathcal{G} \subseteq \mathcal{D}^*$, and $\text{len}(G)$ is the number of elements in a guess $G \in \mathcal{G}$ when viewed as a sequence over \mathcal{D} . The i^{th} element of G is g_i where the index runs from $\text{len}(G)-1$ (the most significant operation) down to 0 (the least significant operation).

(iii) The random variable $X_{\mathcal{Q}}$ assumes values in the set \mathcal{Q} .

Clearly the subsets $\mathcal{R}(K)$ are disjoint for different K 's – each recoding sequence specifies exponentiation to a particular power, and that power is K . We seek to determine $K \in \mathcal{K}$ from guesses $G_1, \dots, G_N \in \mathcal{G}$ based on side channel leakage with the disadvantage that the G_j are probably inconsistent because of erroneous operation deductions, i.e. they may not all represent the same key K ; some G_j may even represent ‘impossible’ recoding sequences (i.e. not belonging to any K).

Starting with key $K \in \mathcal{K}$ a guessed recoding sequence G may be interpreted as the result of two consecutive random experiments. The first step (recode to an operation sequence) is determined by a random number z (and K , of course). The second step (adding noise), namely *recode operation sequence \rightarrow recoding guess*, is determined by a hidden parameter $y \in \mathcal{Y}$ which represents the influence of noise of various forms such as that arising from the measurement process itself or from implemented countermeasures. Formally, the guessing step $R \mapsto G$ can be expressed by a function $\psi: \mathcal{R} \times \mathcal{Y} \rightarrow \mathcal{G}$, $(R, y) \mapsto \psi(R, y)$ since the guessed recoding sequence G depends on the actual recoding sequence $R = \phi(K, z) \in \mathcal{R}$ but not directly on $K \in \mathcal{K}$ or z .

The random variable $X_{\mathcal{K}}$ describes the selection of the key K during initialisation of the attacked cryptosystem. Without loss of generality we may assume the probability $\eta(K)$ of each key is non-zero:

$$\eta(K) \stackrel{\text{def}}{=} \text{Prob}(X_{\mathcal{K}} = K) > 0 \quad \text{for all } K \in \mathcal{K}. \quad (1)$$

The distributions of $X_{\mathcal{K}}$ and $X_{\mathcal{Z}}$ and, of course, the applied recoding scheme determine the distribution ν of the random variable $X_{\mathcal{R}} \stackrel{\text{def}}{=} \phi(X_{\mathcal{K}}, X_{\mathcal{Z}})$. The random variable $X_{\mathcal{G}}$ quantifies the distribution of random recoding sequences that are guessed by the attacker. Theorem 1 considers the situation where an attacker observes N re-uses of the key K . We assume that the associated random variables $X_{\mathcal{K}}, X_{\mathcal{Z},1}, \dots, X_{\mathcal{Z},N}, X_{\mathcal{Y},1}, \dots, X_{\mathcal{Y},N}$ are independent.

Theorem 1. (i) Given recoding sequence guesses $G_1, \dots, G_N \in \mathcal{G}$, the optimal decision strategy $\tau_*: \mathcal{G}^N \rightarrow \mathcal{K}$ selects a key $K^* \in \mathcal{K}$ that maximises the expression

$$\sum_{j=1}^N \log \left(\sum_{R \in \mathcal{R}(K)} \nu(R) \text{Prob}(X_{\mathcal{G},j} = G_j \mid X_{\mathcal{R}} = R) \right) - (N-1) \log(\eta(K)) = \quad (2)$$

$$\sum_{j=1}^N \log \left(\sum_{R \in \mathcal{R}(K)} \nu(R \mid X_{\mathcal{K}}=K) \text{Prob}(X_{\mathcal{G},j}=G_j \mid X_{\mathcal{R}}=R) \right) + \log(\eta(K)). \quad (3)$$

If this maximum is attained for several keys the first key is chosen under any pre-selected order on \mathcal{K} .

(ii) Assume that the adversary is able to detect whenever an operation of the recoded sequence R is carried out and guesses the types of these operations independently to obtain $G \in \mathcal{G}$. Assume also that the conditional probabilities $p(g|r) \stackrel{\text{def}}{=} \text{Prob}(\text{guessed op}^n \text{ type is } g \text{ given the true op}^n \text{ type is } r)$ for guessed operations g do not depend on the position of the operation r but only on the operation types $g, r \in \mathcal{D}$. Then $\text{len}(R) = \text{len}(G)$, and the optimal decision K^* with respect to the N such guesses G_1, \dots, G_N maximises

$$\sum_{j=1}^N \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \text{len}(R)=\text{len}(G_j)}} \nu(R) \prod_{i=0}^{\text{len}(G_j)-1} p(g_{j,i} \mid r_i) \right) - (N-1) \log(\eta(K)) = \quad (4)$$

$$\sum_{j=1}^N \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \text{len}(R)=\text{len}(G_j)}} \nu(R \mid X_{\mathcal{K}}=K) \prod_{i=0}^{\text{len}(G_j)-1} p(g_{j,i} \mid r_i) \right) + \log(\eta(K)). \quad (5)$$

(iii) Assume that $\mathcal{D} = \{0, 1, \overline{1}\}$. Suppose also that (ii) holds with $p(0|0) = 1$, $p(1|\overline{1}) = p(\overline{1}|1) = q \in [0, 0.5]$ and $p(0|1) = p(0|\overline{1}) = 0$. Then the optimal decision K^* maximises

$$\sum_{j=1}^N \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \text{len}(R)=\text{len}(G_j), \\ \text{supp}_0(R)=\text{supp}_0(G_j)}} \nu(R) \left(\frac{q}{1-q} \right)^{\text{Ham}(R, G_j)} \right) - (N-1) \log(\eta(K)) = \quad (6)$$

$$\sum_{j=1}^N \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \text{len}(R)=\text{len}(G_j), \\ \text{supp}_0(R)=\text{supp}_0(G_j)}} \nu(R \mid X_{\mathcal{K}}=K) \left(\frac{q}{1-q} \right)^{\text{Ham}(R, G_j)} \right) + \log(\eta(K)) \quad (7)$$

where $\text{supp}_0(R)$ and $\text{supp}_0(G_j)$ are the sets of positions in $R = (r_{\text{len}(R)-1}, \dots, r_0)$ and $G_j = (g_{j,\text{len}(R)-1}, \dots, g_{j,0})$ for which the type of operation is a squaring, i.e. $r_i = 0$. Further, $\text{Ham}(R, G_j)$ denotes Hamming distance, viz. the number of positions for which the operations in R and G_j differ.

(iv) If $X_{\mathcal{K}}$ is uniformly distributed on \mathcal{K} the terms “ $(N-1) \log(\eta(K))$ ” in (2), (4), (6) and “ $\log(\eta(K))$ ” in (3), (5), (7) may be omitted.

(v) For any constant $c > 0$, multiplying the probability $\nu(R)$ in (2), (4) or (6) (resp. the conditional probability $\nu(R \mid X_{\mathcal{K}}=K)$ in (3), (5) or (7)) by c for all

$R \in \mathcal{R}$ does not change the optimal decision strategy in (i), (ii), (iii) or (iv), respectively, but may simplify concrete calculations.

Proof. To prove (2) we apply Theorem 3(i) from the Appendix (originally proved as Theorem 1(i) in [10]) with $\Theta = \mathcal{K}$, $\Omega = \mathcal{G}^N$, $\mu =$ the counting measure on \mathcal{G}^N , and the loss function $s(\theta, a) \stackrel{\text{def}}{=} 0$ if $\theta = a$ and $s(\theta, a) \stackrel{\text{def}}{=} 1$ otherwise since, as noted at the start of this section, any kind of false key guess is equally harmful.

Let $\text{Prob}_K(A)$ be the probability of the event A if K is the correct key. Then the optimal decision strategy $\tau^*: \mathcal{G}^N \rightarrow \mathcal{K}$ assigns to the guess tuple $\omega = (G_1, \dots, G_N)$ a key $K^* \in \mathcal{K}$ which minimises

$$\begin{aligned} & \sum_{K' \in \mathcal{K}} s(K', K^*) \eta(K') \text{Prob}_{K'}((X_{\mathcal{G},1}, \dots, X_{\mathcal{G},N}) = \omega) = \\ & \sum_{K' \in \mathcal{K}} \eta(K') \text{Prob}_{K'}((X_{\mathcal{G},1}, \dots, X_{\mathcal{G},N}) = \omega) - \eta(K^*) \text{Prob}_{K^*}((X_{\mathcal{G},1}, \dots, X_{\mathcal{G},N}) = \omega) \end{aligned}$$

or, equivalently, maximises $\eta(K^*) \text{Prob}_{K^*}((X_{\mathcal{G},1}, \dots, X_{\mathcal{G},N}) = \omega)$. Recall that $X_{\mathcal{K}}, X_{\mathcal{Z},1}, \dots, X_{\mathcal{Z},N}, Y_{\mathcal{Z},1}, \dots, Y_{\mathcal{Z},N}$ are independent and $X_{\mathcal{G},j} = \psi(\phi(X_{\mathcal{K}}, X_{\mathcal{Z},j}), X_{\mathcal{Y},j})$. For each fixed $K \in \mathcal{K}$ and $\omega = (G_1, \dots, G_N) \in \mathcal{G}^N$ we have

$$\begin{aligned} \text{Prob}_K((X_{\mathcal{G},1}, \dots, X_{\mathcal{G},N}) = \omega) &= \prod_{j=1}^N \text{Prob}_K(X_{\mathcal{G},j} = G_j) \\ &= \prod_{j=1}^N \sum_{z_j \in \mathcal{Z}} \text{Prob}(X_{\mathcal{Z},j} = z_j) \text{Prob}(X_{\mathcal{G},j} = G_j \mid X_{\mathcal{Z},j} = z_j, X_{\mathcal{K}} = K) \\ &= \prod_{j=1}^N \sum_{z_j \in \mathcal{Z}} \text{Prob}(X_{\mathcal{Z},j} = z_j) \text{Prob}(X_{\mathcal{G},j} = G_j \mid X_{\mathcal{R}} = \phi(K, z_j)) \\ &= \eta(K)^{-N} \prod_{j=1}^N \sum_{z_j \in \mathcal{Z}} \text{Prob}(X_{\mathcal{K}} = K, X_{\mathcal{Z},j} = z_j) \text{Prob}(X_{\mathcal{G},j} = G_j \mid X_{\mathcal{R}} = \phi(K, z_j)) \\ &= \eta(K)^{-N} \prod_{j=1}^N \sum_{R \in \mathcal{R}(K)} \nu(R) \text{Prob}(X_{\mathcal{G},j} = G_j \mid X_{\mathcal{R}} = R) \end{aligned}$$

since $\phi(K, \mathcal{Z}) = \mathcal{R}(K)$ and the sets $\mathcal{R}(K')$ are mutually disjoint for different keys K' . As the logarithm function is strictly increasing, the last formula implies (2). Moreover, since the sets $\mathcal{R}(K')$ are mutually disjoint, for any $R \in \mathcal{R}(K)$ we have $\nu(R \mid X_{\mathcal{K}} = K) = \nu(R)/\eta(K)$, which proves (3). Substituting the additional conditions from (ii) into (2) and (3) yields (4) and (5).

Assertion (iii) follows from (ii) since, when $\text{supp}_0(R) = \text{supp}_0(G_j)$,

$$\begin{aligned} \prod_{i=0}^{\text{len}(G_j)-1} p(g_{j,i}, r_i) &= q^{\text{Ham}(R, G_j)} (1-q)^{\text{len}(G_j) - |\text{supp}_0(G_j)| - \text{Ham}(R, G_j)} \\ &= \left(\frac{q}{1-q} \right)^{\text{Ham}(R, G_j)} (1-q)^{\text{len}(G_j) - |\text{supp}_0(G_j)|}. \end{aligned}$$

Otherwise the product is zero. Since the last factor only depends on G_j (fixed) but not on R it can be factored out in (4) and (5). In particular, it has no impact on the location of the maximum, and thus may be dropped. Similarly, in (iv) $\eta(K)$ is assumed to be the same for all $K \in \mathcal{K}$ and thus may be dropped. Assertion (v) follows immediately from the functional property $\log(ab) = \log(a) + \log(b)$ of the logarithm function. ■

Remark 2. (i) Theorem 1(i) is very general. It covers any recoding scheme as long as the random variables $X_{\mathcal{K}}, X_{\mathcal{Z},1}, \dots, X_{\mathcal{Z},N}, X_{\mathcal{Y},1}, \dots, X_{\mathcal{Y},N}$ are independent. In particular, there are no restrictions on maps $\phi: \mathcal{K} \times \mathcal{Z} \rightarrow \mathcal{R}$ or $\psi: \mathcal{R} \times \mathcal{Y} \rightarrow \mathcal{G}$. (ii) In the case of several best candidates in Theorem 1 (which should be a rare event in practice) the rule that one of them is selected according to some pre-defined order has ‘technical reasons’, namely, to ensure the measurability of the decision strategy. Alternatively, one could pick one of the best key candidates at random (defining a randomised decision strategy).

Definition 2. With regard to expression (2), define the (weighted) “credibility” function $\text{cred}: \mathcal{G} \times \mathcal{K} \rightarrow R$ by

$$\text{cred}(G, K) \stackrel{\text{def}}{=} \sum_{R \in \mathcal{R}(K)} \nu(R) \text{Prob}(X_{\mathcal{G}} = G \mid X_{\mathcal{R}} = R). \quad (8)$$

A large value for this credibility function implies that G is a likely recoding sequence guess for key K .

Corollary 1. If η is uniformly distributed on \mathcal{K} (i.e. if all keys are equally likely) for recoding sequence guesses G_1, \dots, G_N the optimal decision strategy selects a key $K^* \in \mathcal{K}$ that maximises $\sum_{j=1}^N \log(\text{cred}(G_j, K))$ for $K \in \mathcal{K}$.

As in theorem Theorem 1(ii), suppose that $\text{Prob}(X_{\mathcal{G}} = G \mid X_{\mathcal{R}} = R) = \prod_{0 \leq i < \text{len}(G)} p(g_i | r_i)$. Then the computation of $\text{cred}(G, K)$ is computationally feasible because recodings are performed by a finite automaton (FA) which typically has very few states. The state of the finite automaton incorporates the difference between the key suffix which has been read and the recoded key suffix which has been generated. The finite automaton reads the next key bit and uses a random input to decide the next recoded digit to output and which transition to make to its next state. For each state s of the automaton, define

$$\text{cred}(G, K', s) \stackrel{\text{def}}{=} \sum_{R \in \mathcal{R}(K')} \nu(R, s) \text{Prob}(X_{\mathcal{G}} = G \mid X_{\mathcal{R}} = R) \quad (9)$$

where $\nu(R, s)$ corresponds to the FA reaching state s after generating the recoding R of key suffix K' . All the component functions can be evaluated iteratively by processing the bits of K sequentially (here from right to left): $\nu(R, s)$ is given by the product of the state transition probabilities for the path through the FA to s which yields R , and, if $K'' = d||K'$ has leading (i.e. most significant) bit

d , then $\text{cred}(G, K'', s')$ can be expressed easily as a linear combination of values $\text{cred}(G, K', s)$ for the suffix K'' of K . The work is then proportional to the length of K and the number of FA states. Algorithm 1 (in the Appendix) provides a concrete example for the Ha-Moon recoding scheme.

5 Traces

The rest of this paper investigates the extent to which the main theorem (Theorem 1) applies to real-world scenarios and, in subsequent sections, enables the computationally feasible recovery of a secret key K . From here on, it is assumed that side channel leakage is sufficient to identify the trace sections which correspond to individual long integer multiplicative operations (or elliptic curve point operations) when the key is used. Given that a standard counter-measure to timing attacks is to ensure that these operations always take the same number of clock cycles, it should be relatively straight-forward to divide the trace correctly from knowledge of the expected number of operations.

The decision about whether a section of side channel leakage represents a particular type of multiplicative operation is not clear cut. As a result of noise, the decision can only be made with a certain probability of correctness. Initial processing of the leakage from each operation uses templates of the expected leakage from each operation type and takes account of the relative probabilities with which the particular digits of \mathcal{D} occur in the given recoding scheme. This yields a set of probabilities, one for each $r \in \mathcal{D}$ and summing to 1, that the operation involved digit r . For convenience, we define a *trace* to be the result of this pre-processing:

Definition 3. $\mathcal{T} \subseteq ([0, 1]^{|\mathcal{D}|})^*$ denotes the set of traces considered by an attacker. The i^{th} element t_i of $T \in \mathcal{T}$ is a list of probabilities, one for each $r \in \mathcal{D}$, that the corresponding side channel measurement represents the operation r .

Formally, for power attacks for example, each operation $r \in \mathcal{D}$ induces a probability distribution on the set of all possible power traces. The exact probability vector t_i follows from the convex combination of these probability distributions with regard to the probabilities with which the particular operations occur (generally) in the recoding scheme. In real-world scenarios the probability vectors t_i can hardly be determined exactly, but roughly estimated.

Using the above definition of a trace, we need to convert each trace $T_j \in \mathcal{T}$ into a guess $G_j \in \mathcal{G}$ before applying Theorem 1(ii) to this situation. The straightforward strategy is to treat each operation separately and to select the most likely candidate $g_{j,i} \in \mathcal{D}$, i.e. the operation that maximises $t_{j,i}$. Formally, the conditional probabilities $p(g_{\dots} | r_{\dots})$ used in Theorem 1 are given by first averaging over sections of side channel traces (such as those described in the previous paragraph) which correspond to the execution of the same multiplicative operation $r_{\dots} \in \mathcal{D}$ at the same position within the recoding sequence. Then, secondly, these conditional probabilities are averaged over the different positions in the recoding sequence. Practically, these conditional probabilities can simply be estimated by

applying this procedure (assigning trace sections, deciding on the most probable operation etc.) to a sample of side channel traces with known recodings.

For many applications this strategy should be appropriate. However, depending on the concrete scenario it may have two difficulties. The first is that the probabilities in Theorem 1(ii) neither depend on the particular side channel trace nor (which may be of less importance) on the position i . In contrast to Theorem 2 below, Theorem 1(ii) applies averaged probabilities. An advantage of Theorem 1(ii) and, in particular, of Theorem 1(iii) is their simple formulae. But using Theorem 2 and traces avoids the second difficulty, namely that of explicitly determining the best values G_j to use in advance⁵.

Formally, for $N = 1$, Theorem 2 may be viewed as a special case of Theorem 1(i), which considers the whole recoding sequence R_1 and the guess G_1 at the same time. (Note that each t_{j,i,r_i} may formally be replaced by a trace- and position-dependent conditional probability $p_{j,i}(g_{j,i} | r_i)$ with some guess $g_{j,i}$.) A straight-forward generalisation of Theorem 1(i), allowing trace-dependent conditional probabilities $\text{Prob}_j(X_{\mathcal{G},j}=G_j | X_{\mathcal{R}}=R)$, comprises the general case $N \geq 1$. In particular, the proof of Theorem 1 can easily be adapted to Theorem 2. Theorem 1(iv) and (v) also remain valid for traces.

Theorem 2. *Given traces $T_1, \dots, T_N \in \mathcal{T}$, the optimal decision strategy $\tau_* : \mathcal{T}^N \rightarrow \mathcal{K}$ selects a key $K^* \in \mathcal{K}$ that maximises the expression*

$$\sum_{j=1}^N \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \text{len}(R)=\text{len}(T_j)}} \nu(R) \prod_{i=0}^{\text{len}(T_j)-1} t_{j,i,r_i} \right) - (N-1) \log(\eta(K)). \quad (10)$$

If this maximum is attained for several keys the first is chosen under any pre-selected order on \mathcal{K} .

6 Application of the Main Theorem

In Sections 4 and 5 we determined optimal decision strategies. To demonstrate the power of its optimal decision strategy, Theorem 1(ii) was applied to small key sizes with the Ha-Moon recoding scheme using Algorithm 1 of the Appendix. Algorithm 1 allows one to compute efficiently the credibility function $\text{cred}(G, K)$ for any required (G, K) and key size. The Ha-Moon recoding scheme maps the binary representation of key K into a representation that uses the digits 0, 1 and -1 . Rather than using $\mathcal{D} = \{0, 1, \bar{1}\}$ also for the recoding sequences, to avoid confusion we will write these operation sequences using ‘ S ’ to denote a squaring, ‘ M ’ a multiplication by the base C , and ‘ \bar{M} ’ a multiplication by C^{-1} . Then digits 0, 1, and -1 in the recoding correspond to the operation sequences ‘ S ’, ‘ SM ’ and ‘ \bar{SM} ’ respectively.

⁵ Indeed, choosing the $g_{j,i}$ first and independently for all positions may lead to impossible guesses which are not recoding sequences of any key.

A large number of stochastic simulations were performed. In each simulation we generated: a random key $K \in \mathcal{K} \stackrel{\text{def}}{=} \{0,1\}^n \setminus \{(0,\dots,0)\}$ where n denotes the length of K , N random recodings which delivered operation sequences R_1, \dots, R_N and some related recoding guesses G_1, \dots, G_N . More precisely, the guess G_j was derived from the operation sequence R_j by replacing the correct operations independently with the conditional probabilities $p('M'|S') = 0.2$, $p('M'|S) = 0.1$, $p('M'|S') = 0.2$, $p('S'|M') = 0.1$, $p('M'|M') = 0.2$ and $p('S'|M) = 0.1$, which corresponds to noise in the side channel traces. Hence $p('Y'|Y) = 0.7$ for each operation ' Y '. Given G_1, \dots, G_N we applied Algorithm 1 exhaustively to all $2^n - 1$ admissible keys K' . For each row in Table 1 we performed 100 simulations. The integers in the column entitled "The correct key was ranked 2", for instance, give the number of simulations out of 100 for which the correct key was ranked second for the given parameter set (Key length, N), e.g. 5 out of 100 for key length 15 and $N = 10$.

Table 1. Ha-Moon recoding scheme: simulation results for short keys

Key length	N	The correct key was ranked					
		1	2	3-9	10-99	100-999	> 1000
15	2	9	2	19	38	27	7
15	3	10	7	26	36	19	2
15	5	45	12	20	18	4	1
15	10	84	5	9	1	1	0
20	10	57	20	20	2	1	0

These results clearly underline the strength of the optimal decision strategy. However, a major problem with the optimal decision strategy is that it is computationally infeasible for keys of cryptographic size – there are too many keys to search for the best one. Unlike in, e.g., [6,7] or in several examples in [10] we cannot handle single key bits or small blocks of key bits either independently or at least sequentially. Instead, the optimal strategy does not describe how to recover *part* of a key, only how to recover the *whole* key. In general it is not possible to associate the key bits independently yet correctly with recoding operations. Where such an association is possible, the key bits might be recovered independently and in parallel or sequentially with effort which is linear rather than exponential in the key length.

7 Incremental Key Construction

General complexity and other issues (see [13], §1) seem to dictate that the best fit (i.e. most probable) key that it is feasible to find has to be generated sequentially bit by bit. We take this approach here, following [13], and this enables the processing of partial keys to be closely related to the theory already presented for full length keys.

It makes sense to determine the bits in the order that recoding takes place, which is assumed here, for convenience, to start at the least significant (right-most) end. So key suffixes of increasing length will be generated. This is done using corresponding suffixes of ‘traces’ in the sense of Definition 3. In order to maximise the expression in Theorem 2, each key suffix must normally have, for each trace, a very good fit between one (or more) of its recoding sequences and the suffix of the trace with the same length. In subsequent sections we try to justify this reasonable assumption while working out its implications. If a sufficiently large set of good key suffixes are processed the best fit key suffix should always be included, so that the best key emerges at the end of the algorithm. However, the longer the key the more likely there is to be a suffix which is too poor a fit to be retained, and our algorithm will fail more frequently in these cases.

Suppose we select a key suffix $K^{(n)}$ of length n which yields the best match to all the leakage from that point onwards, i.e. the best match to trace suffixes. Formally, under the optimal decision strategy model $K^{(n)}$ provides the maximum of

$$\sum_{j=1}^N \log \left(\sum_{\substack{R \in \mathcal{R}(K'): \\ \text{len}(R) \leq \text{len}(T_j)}} \nu(R) \prod_{i=0}^{\text{len}(R)-1} t_{j,i,r_i} \right) - (N-1) \log(\eta(K')) \quad (11)$$

over all keys K' of n bits. (There are some minor issues with end conditions which we will ignore.) If all bits of $K^{(n)}$ are accepted as belonging to the near best-fit full length key K then a decision is being made on the initial, i.e. leftmost, bits of $K^{(n)}$ which does not take into account all available information. Recoding choices are not made independently of previous input; they depend on the state of the recoding finite automaton arising from the previous input, and the influence can persist measurably for several bits. Hence we should ignore the first λ bits, say, of $K^{(n)}$ and choose only its last $n-\lambda$ bits. Thus, the obvious algorithm is to compute iteratively for $n = 1, 2, 3, \dots$ the fitness of every $K^{(n)}$ which extends the previously chosen $K^{(n-\lambda-1)}$, select the best, and use that to determine $K^{(n-\lambda)}$.

This algorithm with the above formula was applied to the (first) Ha-Moon recoding scheme [4], but so far without success. Even worse, under the leakage model described in [13], the resulting “best fit” full length key was indistinguishable from a randomly chosen key – on average half the bits were incorrect. This spurred an investigation into what modifications are necessary to obtain useful results as it is known (e.g. from [3,5,13]) that it is possible to recover the secret key when leakage is weak. It was hoped that the optimal decision strategy would lead to a much more powerful algorithm.

8 Simulation Experiments

With the Ha-Moon scheme [4] as a test case, the formula (11) was modified in a large number of ways to discover what choices lead to a useful, computationally

feasible algorithm. By Theorem 1(iv) choice of a uniform key space \mathcal{K} allowed the formula to be simplified by ignoring the $\log(\eta(K))$ term.

Of all the modifications attempted, only one seemed essential to obtain anything better than how a random key choice would perform, and that was substituting Max for the second \sum in (11). Thus we became interested in iteratively finding the n -bit suffix $K^{(n)}$ which closely maximises

$$\sum_{j=1}^N \log \left(\text{Max} \left\{ \nu(R) \prod_{i=0}^{\text{len}(R)-1} t_{j,i,r_i} \mid R \in \mathcal{R}(K^{(n)}), \text{len}(R) \leq \text{len}(T_j) \right\} \right). \quad (12)$$

(Note that (12) moves the search for a maximum within subsets of recoding sequences to individual recoding sequences. This is computationally more easy.) With sufficiently strong leakage and enough traces, this change results in a significant portion of the predicted bits of K being correct. However, it exposes the factor $\nu(R)$ which gives undue weight to shorter recoding sequences. In the original formula (10) recoding sequences which were too short could not arise because of the constraint $\text{len}(R) = \text{len}(T_j)$. Now, with only an upper bound on the length, some re-balancing is necessary. Normalising using $\text{len}(R)$ yields the much more frequently successful formula

$$\sum_{j=1}^N \log \left(\text{Max} \left\{ \prod_{i=0}^{\text{len}(R)-1} t_{j,i,r_i}^{1/\text{len}(R)} \mid R \in \mathcal{R}(K^{(n)}), \text{len}(R) \leq \text{len}(T_j) \right\} \right). \quad (13)$$

We could have added a weighting that accounted more for the expected length of a recoding sequence for an n -bit key which has given trace length for the full key, but did not do so.

It was noted above that only the final $n-\lambda$ bits should be chosen from $K^{(n)}$. This provides the possibility of treating the first λ bits differently from the later ones. In particular, the two modifications described for (11) can be applied only to the recoding operations corresponding to the last $n-\lambda$ bits or only to the operations corresponding to the first λ bits. However, this different treatment of the two sequences of bits did not give better results.

The formula (13) and related algorithm are now close to what was used in [13]. The main difference is that here we have a product of probabilities to maximise rather than the following sum of distances to minimise:

$$\sum_{j=1}^N \text{Min} \left\{ \frac{1}{\text{len}(R)} \sum_{i=0}^{\text{len}(R)-1} (1-t_{j,i,r_i}) \mid R \in \mathcal{R}(K^{(n)}), \text{len}(R) \leq \text{len}(T_j) \right\}. \quad (14)$$

However, the absence of the “log” from this formula suggests that the contributions from each trace might be either multiplied or added together. So a promising alternative to (13) might be found in

$$\frac{1}{N} \sum_{j=1}^N \text{Max} \left\{ \prod_{i=0}^{\text{len}(R)-1} t_{j,i,r_i}^{1/\text{len}(R)} \mid R \in \mathcal{R}(K^{(n)}), \text{len}(R) \leq \text{len}(T_j) \right\} \quad (15)$$

which is normalised with respect to the number of traces N . In practice, the simulations did work a little better for this formula than for (13), so that it became our final choice. Moreover, this “most probable” version proved to be significantly better than the “best fit” version (14) which appears in [13] unless a very specific choice of parameters is made in [13].

9 An Ordered Search

The algorithm of the previous section rarely identifies the correct key although, with enough traces and sufficiently strong leakage, the resulting key $K^* = K^{(\text{len}(K))}$ should yield a close to maximal value for (2), as should the correct key. The remaining problem is therefore to organise a computationally feasible ordered search of the key space to find the correct key K from K^* .

In a simulation, a quick scan of this close-to-most-probable key K^* shows that, unless the strategy failed, most of the key bits are correct, or differ from the correct ones according to specific patterns which make no difference to the likelihood of the key (such as a sequence of one or more inverted bits). Thus, the correct key might be found by allowing for all the related patterns of equal probability and by looking at each bit and obtaining an estimate of confidence in its correctness. This estimate is naturally based on the values returned by the expression (15) which was used to choose the bit initially. Then the search of the key space is performed by changing more and more of those bits for which confidence is lowest (and any subsequent related pattern) until the correct key is found.

With the last $n-\lambda-1$ bits of K^* already chosen, there are $2^{\lambda+1}$ choices for $K^{(n)}$ which are supplied to (15). There are 2^λ cases for which bit $n-\lambda$ is 0, and 2^λ cases for which bit $n-\lambda$ is 1. Let $\text{cred}(0, n-\lambda)$ and $\text{cred}(1, n-\lambda)$ denote the maximum values returned by (15) when evaluated over these two partial key subsets. Whichever is larger determines the choice of bit $n-\lambda$, and so it is natural to use some measure of their proximity to provide a confidence value for the bit decision. Out of several possibilities,

- an effective discriminant was found to be the larger of the two ratios $\text{cred}(1, n-\lambda)/\text{cred}(0, n-\lambda)$ or $\text{cred}(0, n-\lambda)/\text{cred}(1, n-\lambda)$.

The larger this ratio, the greater the confidence that the bit is chosen correctly.

When evaluating (15) over the 2^λ keys of interest, the variance of $\text{cred}(b, n-\lambda)$ decreases as n increases. This implies that, on average, the confidence function will return smaller values with increasing key size. Of course, this is what happens, and it corresponds to the fact that bits are predicted with decreasing accuracy as n becomes larger. A re-scaling to compensate for this is the modification to (15) which results in the following expression for maximising:

$$\frac{1}{N} \sum_{j=1}^N \text{Max}\left\{ \prod_{i=0}^{\text{len}(R)-1} t_{j,i,r_i}^{n/\text{len}(R)} \mid R \in \mathcal{R}(K^{(n)}), \text{len}(R) \leq \text{len}(T_j) \right\} \quad (16)$$

This re-scaling may give some benefit when applied, but was usually marginally poorer than (15) over all the choices of parameters we tried, such as variations in the level of leakage and in the key lengths.

10 Complexity

This section considers the space and time complexity for obtaining a near best-fit key K^* using expression (15). We are assuming that all necessary pre-processing has already been performed to derive traces in the sense of Definition 3 from the side channel measurements and template information.

Our model assumes 1 unit of time for any arithmetic operation involving one or two bits. It also assumes one unit of time for moving (reading, writing etc.) or copying a single bit and one unit of space to store a single bit. This is not quite realistic since address sizes and wire connect grow as the logarithm of the volume of data, but is adequate for the quantities of data under consideration. Moreover, since most bits are effectively random and we avoid storing multiple copies of data, there are no data compression techniques which are likely to be useful.

For convenience, we further assume that all probabilities can be stored with sufficient accuracy using $O(1)$ bits. In practice our simulations worked using 32-bit real arithmetic without any hint of problems. Overall this complexity model enables one to run small trial cases first, and scale up using the complexity results to obtain good approximations to the space and time requirements for an attack on secret keys of cryptographic size.

Following earlier notation, we have a key K^* of n_K bits which is constructed bit by bit, N traces, and λ leading bits of the suffixes which we ignore when selecting the next bit of K^* . At any one time we have decided the least significant $n - \lambda - 1$ bits of K^* . We have $2^{\lambda+1}$ possibilities for the remaining bits of the n -bit suffix and hence that number of keys to consider when determining the next bit of K^* .

Recall our assumption that recodings are converted into sequences of operations with digit 0 generating one operation (a squaring) and non-zero digits generating two (a squaring and a multiplication). Consequently, when the recoding finite automaton (FA) has read a suffix of n bits, the resulting recoding sequences may have any length from n up to $2n$. Suppose also that the recoding FA has F possible states. These states include, but are not limited to, storing the shifted difference (i.e. carry or borrow) between the value of the key suffix read by the FA and the recoding output by it. (For the Ha-Moon scheme this is always 0 or 1, and the corresponding FA has two states.) Whenever two recoding sequences have the same length and left the FA in the same state, we can ignore the one with the smaller value of the product of trace values in (15). It cannot give the maximum, nor contribute to any maximum when the FA recodes more bits. Hence there are only up to $F(n+1)$ sequences which need to be maintained in order to select the best recoding.

The iterative step starts at $n = \lambda + 1$ in order to determine the bit of index 0. If there were just one trace then, after incrementing n , the general induction

step would start with a set of best recoding sequences for the least significant $n - \lambda - 1$ bits of K^* and the corresponding values for the product of traces values needed in (15). There are up to $(n - \lambda)F$ of these, so $O((n - \lambda)F)$ space is required for this. In fact, this is needed for all N traces, so $O((n - \lambda)FN)$ space is used to hold the data required by the induction step. A three dimensional array is used for this in order to have direct access in unit time to the data elements. The total space order is also enough to include the decided bits of K^* .

Extending the recoding sequences from representing a suffix of length $n - \lambda - 1$ to a suffix of length n just means generating the same set of data incrementally for longer suffixes. A depth first traversal is made of the binary tree of depth $\lambda + 1$ representing the remaining choices for the n -bit suffix of K^* . Along each branch of the tree one such data set needs to be stored at each node. So $O(nFN\lambda)$ space is required for data storage during the induction step. At each node in the tree, each recoding sequence of the parent node's data set is extended by processing the key bit value which labels the current node. The FA generates $O(F)$ choices which are used to create or update the items in the data set of this node. This means $O(F)$ time per recoding sequence, and a total of $O(nF^2N)$ per node of the tree. With $2^{\lambda+2} - 2$ nodes to treat, the time order is $O(2^\lambda nF^2N)$.

If, instead, we are able to hold all the data sets for all the nodes in the above tree simultaneously, then we do not need to regenerate the same data for up to λ consecutive values of n . Instead we traverse the leaves of the tree from the preceding value of n and generate the data for the leaves of the tree for the current value of n . However, as this still means traversing $2^{\lambda+1}$ nodes, the time order is not reduced although there will be a speed-up by a factor of about 2.

Whenever the data set for a leaf of the tree has been generated, we have the value of the product term in (15) for all the $O(Fn)$ recoding sequences associated with each trace. Hence the maximum value can be obtained for each trace, and the sum over all traces calculated. This does not add to the time complexity as it requires $O(1)$ time per sequence. Finally, we must obtain the maximum value of (15) over the $2^{\lambda+1}$ key choices in order to determine bit $n - \lambda$ of K^* . Again, this does not add to the time order. If we want to rank the bits in order of certainty, then the ratio of credibility values is obtained at this point by taking the maximums over the two sub-trees corresponding to the two choices for bit $n - \lambda$.

Thus, neglecting special processing for the most significant λ bits of K^* (when we must decrease λ rather than increase n), the iterative process to compute K^* takes $O(n_K FN)$ space and $O(2^\lambda n_K F^2 N)$ time.

11 Numerical Results

Direct comparison of the improved algorithms here with the results of [13] is made difficult by the fact that [13] limits the key search to a fixed maximum number of recoding states⁶ whereas here the number is not limited. Specifically, in (15) the maximum is given by incrementally extending a set of best recoding

⁶ The recoding state is a pair consisting of the state of the recoding finite automaton and the number of operations it has generated.

Table 2. Fraction of key guesses with all bit errors in the 24 most dubious positions for Ha-Moon recoding [4]. ¹= original for 10 recodings, ²= improved version (see text).

Method	Leakage Level L	Key Length	No. of Traces	Av. No. Bit Errors	Fraction with all errors in worst 24	Fraction for 10% best-fitting Keys
[13] ¹	0.7	192	5	—	0.0027	—
[13] ²	0.7	192	5	23.2	0.011	0.020
Here	0.7	192	5	20.7	0.013	0.014
[13] ²	0.7	384	10	19.0	0.003	0.002
Here	0.7	384	10	18.2	0.003	0.010
[13] ²	0.7	1024	40	13.5	0.3	0.3
Here	0.7	1024	40	13.4	0.28	0.28
"	0.6	192	64	30.9	0.19	0.43
"	0.6	384	128	39.4	0.12	0.24

sequences whenever another key bit is decided, and there must be a best recoding sequence for every recoding state that might have been reached so far. For an n -bit key suffix, the Ha-Moon algorithm could have generated a recoding sequence of up to $2n$ operations and left the recoding finite automaton in one of two states. Hence there are up to $4n$ best recoding sequences per trace which need to be extended in order to extract the maximum, and restricting this number turns out to be detrimental. Hence, in order to generate comparative values we modified our simulation to use Walter's metric but without his bound on recoding states. On its own, this modification arising from the optimal strategy approach led to the majority of the improvement in performance tabulated below – witness the first two rows in Table 2.

Unlike the example in Section 6 we assume a side-channel leakage which only allows us to distinguish between squarings and multiplications with some certainty but not between the two multiplicative operations ' M ' and ' \bar{M} ' corresponding to digits 1 and -1 . This is easy to model. The i^{th} element t_i of a trace T is a set of three probabilities, one for each operation type: $t_i = \{p_{i0}, p_{i1}, p_{i\bar{1}}\}$ in which we ensure that $p_{i1} = p_{i\bar{1}}$. We selected this scenario to have a fair comparison with the results in [13].

Let L denote the average level of side channel leakage, that is, the fraction of square or multiply operations which are independently guessed correctly. So $L = \frac{1}{2}$ means no leakage, when blind guessing makes half the bits correct, and $L = 1$ corresponds to full leakage, when all operations and hence all bits are completely determined⁷. For several realistic cryptographic key lengths and numbers of traces, Table 2 gives the probability that all the incorrectly guessed bits of the most likely key are among the 24 bits which the credibility metric shows are most likely to be in error (see Section 9). It is computationally feasible to correct all errors in

⁷ Guessing all operations to be squarings in the binary exponentiation algorithm makes two thirds of the operations correct (on average), not a half. Here we are modelling the noise which is added to the actual recoding sequence, changing the average correctness of the decision between squaring or multiplication from 1 to L .

such guessed keys and thereby recover the true key. The last column explores the possibility that the most probable key guesses are those for which the bit errors are confined to the 24 most dubious positions. This new measure is not in previous literature, and clearly indicates that the adversary can select cases for which the method is more likely to succeed. He collects sets of side channel data for a number of different keys, computes the best-fitting key in each case, and then selects the 10%, say, which yield the highest value for (15). Then, in parallel for the selected keys, he modifies more and more of the most dubious bit values in decreasing order of likelihood until a correct key is found.

12 Conclusion

A computationally feasible algorithm has been presented for determining the secret key used repeatedly in exponentiations where there is weak side channel leakage and randomised recoding has been employed in an attempt to nullify the effect of that leakage. The algorithm was derived from an optimal decision strategy and is an improvement over prior techniques. Using it, it is easy both to determine which results have few bit errors, and to locate the potential bit errors. Hence it is frequently possible to recover the key using much weaker leaked data than before. The derivation also highlights points where significant modifications had to be made to the optimal strategy to obtain a computationally feasible algorithm.

References

1. Brier, E., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 335–345. Springer, Heidelberg (2002)
2. Chaum, D.: Blind Signatures for Untraceable Payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology – Crypto 1982, pp. 199–203. Plenum Press, New York (1983)
3. Green, P.J., Noad, R., Smart, N.: Further Hidden Markov Model Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 61–74. Springer, Heidelberg (2005)
4. Ha, J.C., Moon, S.J.: Randomized signed-scalar multiplication of ECC to resist power attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 551–563. Springer, Heidelberg (2003)
5. Karlof, C., Wagner, D.: Hidden Markov Model Cryptanalysis. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 17–34. Springer, Heidelberg (2003)
6. Kocher, P.C.: Timing Attack 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)
7. 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)

8. Liardet, P.-Y., Smart, N.P.: Preventing SPA/DPA in ECC Systems using the Jacobi Form. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 391–401. Springer, Heidelberg (2001)
9. Oswald, E., Aigner, M.: Randomized Addition-Subtraction Chains as a Counter-measure against Power Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 39–50. Springer, Heidelberg (2001)
10. Schindler, W.: On the Optimization of Side-Channel Attacks by Advanced Stochastic Methods. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 85–103. Springer, Heidelberg (2005)
11. Walter, C.D.: Breaking the Liardet-Smart Randomized Exponentiation Algorithm. In: Proc. Cardis 2002, San José, November 2002, pp. 59–68. Usenix Association, Berkeley (2002)
12. Walter, C.D.: Issues of Security with the Oswald-Aigner Exponentiation Algorithm. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 208–221. Springer, Heidelberg (2004)
13. Walter, C.D.: Recovering Secret Keys from Weak Side Channel Traces of Differing Lengths. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 214–227. Springer, Heidelberg (2008)
14. Witting, H.: Mathematische Statistik I, Teubner, Stuttgart (1985)
15. Yen, S.-M., Chen, C.-N., Moon, S.J., Ha, J.C.: Improvement on Ha-Moon Randomized Exponentiation Algorithm. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 154–167. Springer, Heidelberg (2005)

A Appendix

A.1 Algorithmic Treatment of Ha-Moon Recodings

In this subsection we provide an efficient algorithm to compute $\text{cred}(G, K)$, defined by (8), for the Ha-Moon recoding scheme [4] for any fixed $G \in \mathcal{G}$ and $K \in \mathcal{K}$. In the recoding step the binary representation K is mapped onto a recoding which is a sequence of digits in $\{-1, 0, 1\}$ with length $\text{len}(K)$ or $\text{len}(K)+1$. Again, we follow a standard convention in which the letters ‘ S ’, ‘ M ’, and ‘ \overline{M} ’ denote a squaring, a multiplication by the base C , and a multiplication by C^{-1} , respectively. So, to obtain the corresponding operation sequence, apart from the most significant digit ($= 1$), each digit of the recoding is substituted as follows: $0 \mapsto 'S'$, $1 \mapsto ('S', 'M')$, and $-1 \mapsto ('S', \overline{M}')$, yielding an element $R \in \mathcal{R}(K)$. In particular, the recoding sequence has length $\leq 2\text{len}(K)$. Note that the probabilities $\nu(R)$ are not identical for all $R \in \mathcal{R}(K)$ in the Ha-Moon recoding scheme.

Assumption 1: We consider the scenario from Theorem 1(ii). We assume further that $K \in \mathcal{K} \stackrel{\text{def}}{=} \{0, 1\}^n$, and that $X_{\mathcal{K}}$ is uniformly distributed on \mathcal{K} , i.e., each admissible key is equally likely. If the next recoding step is not unique the recoding algorithm decides with probability $\frac{1}{2}$ for one of the two alternatives.

Definition 4. (*Ha-Moon recoding scheme [4].*) For $0 \leq v \leq \text{len}(K)$ and $K \in \mathcal{K}$ the term $\mathcal{R}(K, v, \text{len}, c_v) \subseteq \mathcal{R}(K)$ denotes the subset of recoding sequences that require exactly len operations (‘ S ’, ‘ M ’, ‘ \overline{M} ’) and the carry bit c_v to encode the v least significant bits of K . Guesses of recoding sequences $G = (g_j)_{0 \leq j < \text{len}(G)}$

and recoding sequences $R = (r_j)_{0 \leq j < \text{len}(R)}$ are extended to length $2n$ by defining a further symbol, ∞ , and setting $g_j \stackrel{\text{def}}{=} \infty$ for $j \geq \text{len}(G)$ and $r_j \stackrel{\text{def}}{=} \infty$ for $j \geq \text{len}(R)$. Further, the conditional probability function $p(\cdot | \cdot)$ is extended so that $p(\infty | \infty) \stackrel{\text{def}}{=} 1$ and $p(a | \infty) \stackrel{\text{def}}{=} p(\infty | a) \stackrel{\text{def}}{=} 0$ for $a \in \{\text{'S'}, \text{'M'}, \overline{\text{M}}\}$. This allows one to increase the upper bound of the product in (4) beyond $\text{len}(R) - 1$ to $2n - 1$. For $v \geq 0$ the ‘intermediate’ credibility function associated with the guess G and any subset $M \subseteq \mathcal{R}(K) = \bigcup_{v \leq \text{len} \leq 2v; c \in \{0,1\}} \mathcal{R}(K, v, \text{len}, c)$ is

$$\text{cred}_v(G, M) \stackrel{\text{def}}{=} \sum_{R \in M} \nu(R) \prod_{i=0}^{\text{len}_v(R)-1} p(g_i | r_i). \quad (17)$$

The term $\text{len}_v(R)$ denotes the number of operations in R which are used to encode the key bits k_{v-1}, \dots, k_0 . (In particular, for all $R \in \mathcal{R}(K, v, \text{len}, c)$ it is $\text{len}_v(R) = \text{len}$.)

During recoding, the Ha-Moon scheme generates either one or two operations per bit of K , and a carry bit equal to 0 or 1. Hence for given v the length of a recoding sequence ranges from v to $2v$, and so $\mathcal{R}(K)$ is the disjoint union of $2(v+1)$ (possibly empty) subsets $\mathcal{R}(K, v, \text{len}, c)$ with $v \leq \text{len} \leq 2v$ and $c \in \{0,1\}$. Algorithm 1 below computes sequentially the $\text{cred}_v(G, \cdot)$ -values for increasing parameters v and subsets $\mathcal{R}(K, v, \text{len}, c) \subseteq \mathcal{R}(K)$, which finally yields the desired value $\text{cred}(G, K)$. The definition of the subsets $\mathcal{R}(K, v, \text{len}, c)$ and the $\text{cred}_v(\cdot, \cdot)$ -function are closely related to the components of definition (9) used in the generic description of an algorithm for the efficient computation of $\text{cred}(G, K)$. Indeed, both approaches are essentially equivalent. In definition (9) ‘truncated’ recoding sequences are considered, and the product in (17) only considers these operations. The recoding sequences are elements of $\mathcal{R}(K)$ but the probabilities of all recoding sequences with a fixed suffix add up to the probability of the truncated recoding sequence. However, the approach chosen in the appendix is more convenient for Algorithm 1 which splits and merges subsets of $\mathcal{R}(K)$.

The goal of Algorithm 1 is the efficient computation of $\text{cred}(G, K)$ for arbitrary but fixed $G \in \mathcal{G}$ and $K \in \mathcal{K}$.

Remark 3. Our implementation of Algorithm 1 requires the binary representation of key K to contain at least one ‘1’, while Theorems 1 and 2 clearly also cover the zero key. In our simulation experiments presented in Section 6 we restricted the key space to $\{0, 1\}^n \setminus \{(0, \dots, 0)\}$ for simplicity.

Algorithm 1. Initialise by setting $\text{cred}_0(G, \mathcal{R}(K, 0, 0, 0)) \stackrel{\text{def}}{=} \text{Prob}(X_K = K)$. (Note that $\mathcal{R}(K) = \mathcal{R}(K, 0, 0, 0)$.) Assume that we know the ‘truncated’ credibilities $\text{cred}_v(G, \mathcal{R}(K, v, u, c_v))$ for all $u \in \{v, \dots, 2v\}$ and $c_v \in \{0, 1\}$. The next task is therefore to consider key bit k_v and to determine the values $\text{cred}_{v+1}(G, \mathcal{R}(K, v+1, u, c_{v+1}))$ for $v+1 \leq u \leq 2(v+1)$ and $c_{v+1} \in \{0, 1\}$.

Induction Step (over v):

For each $len \in \{v, \dots, 2v\}$ and $c_v \in \{0, 1\}$:

Case A: If $(k_v, c_v) = (0, 0)$ the next digit (viewed from right to left) in any recoding is 0 and thus ‘S’ is the next operation in the recoding sequence. In particular, $\text{cred}_{v+1}(G, \mathcal{R}(K, v+1, len+1, 0) \cap \mathcal{R}(K, v, len, 0)) = p(g_{len} | 'S') \text{cred}_v(G, \mathcal{R}(K, v, len, 0))$.

Case B: If $(k_v, c_v) = (1, 0)$ the next digit of the recoding is 1 or -1 (and thus $c_{v+1} = 0$ or $c_{v+1} = 1$, respectively), each with probability $\frac{1}{2}$. Consequently, $\text{cred}_{v+1}(G, \mathcal{R}(K, v+1, len+2, 0) \cap \mathcal{R}(K, v, len, 0)) = \frac{1}{2}p(g_{len+1} | 'S')p(g_{len} | 'M') \times \text{cred}_v(G, \mathcal{R}(K, v, len, 0))$ and $\text{cred}_{v+1}(G, \mathcal{R}(K, v+1, len+2, 1) \cap \mathcal{R}(K, v, len, 0)) = \frac{1}{2}p(g_{len+1} | 'S')p(g_{len} | 'M') \text{cred}_v(G, \mathcal{R}(K, v, len, 0))$.

Cases C and D: The cases $(k_v, c_v) = (0, 1)$ and $(k_v, c_v) = (1, 1)$ are treated analogously with $\mathcal{R}(K, v, len, 1)$ in place of $\mathcal{R}(K, v, len, 0)$. As in Case B above the set $\mathcal{R}(K, v, len, 1)$ splits into two subsets if $(k_v, c_v) = (0, 1)$, i.e. in Case C.

When this has been completed for all values $len \in \{v, \dots, 2v\}$ and $c_v \in \{0, 1\}$, the credibility values for the subsets $\mathcal{R}(K, v+1, len', c_{v+1})$ are easily computed. For each $v+1 \leq len' \leq 2(v+1)$ and $c_{v+1} \in \{0, 1\}$ ‘related’ subsets are merged to give: $\text{cred}_{v+1}(G, \mathcal{R}(K, v+1, len', c_{v+1})) = \sum \text{cred}_{v+1}(G, \mathcal{R}(K, v+1, len', c_{v+1}) \cap \mathcal{R}(K, v, len'', c''))$, where the sum extends over all $(len'', c'') \in \{len'-1, len'-2\} \times \{0, 1\}$. This equality holds because the subsets on the right side of the equation have a disjoint union equal to the set on the left side. Clearly, $\text{cred}_v(G, \emptyset) = 0$ and $\text{cred}_{v+1}(G, \emptyset) = 0$, and this enables instances of $\text{cred}_{v+1}(G, \mathcal{R}(K, v+1, len', c'))$ to be evaluated without taking a sum over subsets when the second parameter is the empty set.

This completes the induction step from v to $v+1$.

This procedure is continued until $v = \text{len}(K) - 2$ (inclusively). The most significant bit of K then needs special treatment because different operations are used to initialise the exponentiation. Finally, $\text{cred}(G, K) = \text{cred}_{\text{len}(K)}(G, \mathcal{R}(K, \text{len}(K), \text{len}(G), 0))$. \square

Note that for $R \in \mathcal{R}(K)$ we have $\nu(R) = \text{Prob}(X_K = K)2^{-\text{bif}(R)}$ where $\text{bif}(R)$ stands for the number of ‘bifurcations’ in generating the recoding sequence R from K , i.e. the number of positions where two recoding choices are possible, i.e. when $(k_v, c_v) = (0, 1)$ or $(1, 0)$. This causes the factor $\frac{1}{2}$ for Cases B and C.

A.2 Statistical Decision Theory

This subsection provides a brief introduction to statistical decision theory as far as is relevant to understand the concepts of Theorems 1 and 2. We omit all mathematical details. For a more comprehensive treatment we refer the interested reader to [10], Section 2, or to textbooks in this field (e.g. [14]). Reference [10] illustrates the merits of statistical decision theory for side-channel analysis by several examples.

Our focus is side-channel analysis. We interpret side-channel measurements as *realisations* of random variables, i.e. as values assumed by these random variables. The relevant part of the information is covered by noise but an attacker

clearly aims to exploit the available information in an optimal way. Statistical decision theory quantifies the impact of the particular pieces of information on the strength of a decision strategy, and so the search for the optimal decision strategy can be formalised.

Formally, a statistical decision problem is given by a 5-tuple $(\Theta, \Omega, s, \mathcal{DST}, A)$. The statistician (in our context the attacker) observes a sample $\omega \in \Omega$ that he interprets as a realisation of a random variable X with unknown distribution p_θ . On basis of this observation he guesses the parameter $\theta \in \Theta$ where Θ denotes the *parameter space*, i.e., the set of all admissible hypotheses (= possible parameters). Further, the set Ω is called the *observation space*, and the letter A denotes the set of all admissible alternatives the statistician can choose. In the following we assume $\Theta = A$ with finite sets Θ and A .

Example 1. ([10], Example 1)

- (i) Assume that the attacker guesses a particular (single) RSA key bit and that his decision is based upon N measurements. Then $\Theta = A = \{0, 1\}$. For timing attacks, we may assume $\Omega = \mathbb{R}^N$ while $\Omega = \mathbb{R}^{TN}$ for power attacks where T is the number of relevant measurement points per power trace.
- (ii) Consider a power attack on a DES implementation where the attacker guesses a particular 6-bit subkey that affects a single S-box in the first round. Then $\Theta = A = \{0, 1\}^6$.

The term \mathcal{DST} denotes the set of all decision strategies between which the statistician can choose. A (*deterministic*) *decision strategy* is given by a mapping $\tau: \Omega \rightarrow A$. This means that if the statistician applies decision strategy τ he decides on $\tau(\omega) \in A = \Theta$ whenever he observes $\omega \in \Omega$. (It may be noted that for certain statistical applications it is reasonable to consider the more general class of randomised decision strategies ([10], Remark 1(i)). For our purposes we need only concentrate on deterministic decision strategies.)

Finally, the *loss function* $s: \Theta \times A \rightarrow [0, \infty)$ quantifies the harm of a wrong decision, i.e., $s(\theta, a)$ gives the loss if the statistician decides on $a \in A$ although $\theta \in \Theta = A$ is the correct parameter. In the context of side-channel attacks the loss function quantifies the efforts to detect, to localise and to correct a wrong decision, usually a wrong guess of a key part. Clearly, $s(\theta, \theta) \stackrel{\text{def}}{=} 0$ since a correct guess does not cause any loss. We point out that for some side-channel attacks certain types of errors are easier to detect and correct than others ([10], Sect. 6). The optimal decision strategy takes such phenomena into account.

Assume that the statistician uses the deterministic decision strategy $\tau: \Omega \rightarrow A$ and that θ is the correct parameter. The expected loss (= average loss if the hypothesis θ is true) is given by the *risk function*

$$r(\theta, \tau) \stackrel{\text{def}}{=} \int_{\Omega} s(\theta, \tau(\omega)) p_{\theta}(d\omega). \quad (18)$$

In the context of side-channel attacks one can usually determine (at least approximate) probabilities with which the particular parameters occur. This is quantified by the so-called *a priori distribution* η , a probability measure on the parameter space Θ .

Example 2. ([10], Example 2)

(i) (Continuation of Example 1(i).) Assume that k exponent bits remain to be guessed and that the attacker knows that r of them are 1. If the secret key was selected randomly it is reasonable to assume that a particular exponent bit is 1 with probability $\eta(1) = r/k$.

(ii) (Continuation of Example 1(ii).) Here $\eta(x) = 2^{-6}$ for all $x \in \{0, 1\}^6$.

Assume that η denotes the a priori distribution. If the statistician applies the deterministic decision strategy $\tau: \Omega \rightarrow A$ the expected loss is given by

$$R(\eta, \tau) \stackrel{\text{def}}{=} \sum_{\theta \in \Theta} r(\theta, \tau) \eta(\theta) = \sum_{\theta \in \Theta} \int_{\Omega} s(\theta, \tau(\omega)) p_{\theta}(d\omega) \eta(\theta). \quad (19)$$

A decision strategy τ' is *optimal against* η if it minimises the right-hand term. Such a decision strategy is also called a *Bayes strategy* against η .

Theorem 3. ([10], Theorem 1(i), (iii))

Assume that $(\Theta, \Omega, s, DST, A)$ defines a statistical decision problem with finite parameter space $\Theta = \{\theta_1, \dots, \theta_t\} = A$ where DST contains all deterministic decision strategies. Further, let μ denote a σ -finite measure on Ω with $p_{\theta_i} = f_{\theta_i} \cdot \mu$, i.e. p_{θ_i} has μ -density f_{θ_i} , for each $i \leq t$.

(i) The deterministic decision strategy $\tau: \Omega \rightarrow A$,

$$\tau(\omega) \stackrel{\text{def}}{=} a \quad \text{if } \sum_{i=1}^t s(\theta_i, a) \eta(\theta_i) f_{\theta_i}(\omega) = \min_{a' \in A} \left\{ \sum_{i=1}^t s(\theta_i, a') \eta(\theta_i) f_{\theta_i}(\omega) \right\} \quad (20)$$

is optimal against the a priori distribution η . (If the minimum is attained for several decisions, we chose $a \in A$ according to any pre-selected order on A .)

(ii) Assume that $C \subseteq \Omega$ with $p_{\theta_i}(C) = p > 0$ for all $\theta_i \in \Theta$. Then (i) and (ii) remain valid if f_{θ} is replaced by the conditional density $f_{\theta|C}$.

Remark 4. ([10], Remark 2)

(i) A σ -finite measure μ on Ω with the properties claimed in Theorem 3 always exists (e.g. $\mu = p_{\theta_1} + \dots + p_{\theta_t}$).

(ii) For $\Omega = \mathbb{R}^n$ the well-known Lebesgue measure λ_n is σ -finite. (The Lebesgue measure on \mathbb{R}^n is given by $\lambda_n([a_1, b_1] \times \dots \times [a_n, b_n]) = \prod_{i=1}^n (b_i - a_i)$ if $b_i \geq a_i$ for all $i \leq n$.) If Ω is finite or countable the counting measure μ_C is σ -finite. The counting measure is given by $\mu_C(\omega) = 1$ for all $\omega \in \Omega$. In particular, the probabilities $\text{Prob}_{\theta}(X = \omega) = p_{\theta}(\omega)$ can be interpreted as densities with respect to μ_C .

(iii) The examples mentioned in (ii) and combinations thereof cover the cases that are relevant in the context of side-channel analysis.

The probability densities f_{θ} , the a priori distribution η , and the loss function s have an impact on the optimal decision strategy. The probability densities f_{θ} clearly have most influence, and usually their determination is the tough part of the work.

Practical Zero-Knowledge Proofs for Circuit Evaluation

Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi

Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom

{ghadfi,nigel,bogdan}@cs.bris.ac.uk

Abstract. Showing that a circuit is satisfiable without revealing information is a key problem in modern cryptography. The related (and more general) problem of showing that a circuit evaluates to a particular value if executed on the input contained in a public commitment has potentially multiple practical applications. Although numerous solutions for the problem had been proposed, their practical applicability is poorly understood.

In this paper, we take an important step towards moving existent solutions to practice. We implement and evaluate four solutions for the problem. We investigate solutions both in the common reference string model and the random oracle model. In particular, in the CRS model we use the recent techniques of Groth–Sahai for proofs that use bilinear groups in the asymmetric pairings environment. We provide various optimizations to the different solutions we investigate. We present timing results for two circuits the larger of which is an implementation of AES that uses about 30000 gates.

1 Introduction

BACKGROUND. Consider the problem where given a function f , a value y and a commitment $C(x)$ to x , some party A wants to prove to a party B that $f(x) = y$. The proofs should be convincing, but no additional information on x (but $f(x)$) should be revealed. For the case when f is given by an arithmetic circuit, Cramer and Damgård call the problem the *Arithmetic Circuit Problem* [10]. Since in this paper we consider the case when f is given by a binary circuit, we refer to the problem described above as the *Binary Circuit Problem* (**BCP** in short). Note, that **BCP** and the standard problem of circuit satisfiability are similar; in circuit satisfiability the problem is to present the satisfying input x such that $f(x) = y$ from which the validity of the statement can be proved “in the open”; in the **BCP** problem the inputs are given as commitments and then the proof of the validity of the statements needs to be presented so that it reveals no information on the underlying value of x .

There are countless practical and theoretical applications of solutions to this problem. For example, if y is a bit and f is described by a circuit, then a solution for the above problem would allow A to prove that f is satisfiable without revealing information about the (private) satisfying assignment x . Another possible application is the hedge funds scenario where it is often the case that managers need to convince investors that their portfolio has certain properties (e.g. that it meets a certain risk profile) [28]. All this needs to be done without disclosing sensitive information about the particularities of the investments. This problem is again an instance of the above generic problem for appropriate choices of f , x and y . Another way to look at the problem is as a particular case of two-party secure function evaluation where only one party has the input x to the function f and the second party has to learn the output $y = f(x)$ but nothing more; yet unlike standard two-party function evaluation we require a non-interactive solution in this situation.

Due to its importance, the problem has received a lot of attention especially in the case of proofs for circuit satisfiability and numerous solutions exist for both the interactive and non-interactive cases [4,6,10,12,13,14,15,20,23]. In the case of non-interactive proofs, which is the setting that we focus on in this paper, the first definition and solution was proposed by Blum, Feldman and Micalli [4]. Their construction assumes some specific number-theoretic assumption. The *theoretical* efficiency of protocols based on number theoretic assumptions has subsequently been improved [6,12]. Feige, Lapidot and Shamir [15] and later Kilian and Petrank [23] give solutions based on general assumptions. All of the above solutions are in the CRS model. A set of techniques that use the CRS in a very different way than the works above has been developed by Groth et al [20,21]. Their approach is based on various assumptions in bilinear groups and is currently the state-of-the-art for solutions not based on random oracles.

All of the work mentioned above concerns possibility results with only limited interest in the actual practical efficiency of the solutions. Complexity is always stated in terms of big-Oh notation, and determining the precise hidden constants is almost never of interest. An important first step towards the use of solutions for **BCP** (and circuit satisfiability) in practice is to understand to what extent the proposed solutions are actually feasible and this is precisely the goal of this paper. We emphasize that we are concerned with the **BCP** problem in its full generality, and that we do not investigate implementations for particular classes of functions f . When f comes from a certain class, solutions can exploit particular structures and can therefore be made more efficient. For example Szyldo [28] gives a solution for the above fund manager problem when the function is quite particular. Groth and Sahai [21] give proofs for the case when the witness satisfies a set of equations of a certain form. Although we are not explicitly concerned with this case, our results do shed some light on the efficiency of the techniques of [21].

THIS PAPER. We implement and evaluate a general solution that goes back at least to the work of Cramer and Damgård [10]. The idea is that given binary circuit f , and commitments to the values on the input wires, the prover computes the values on all of the remaining wires of the circuit. The prover then sends

commitments to all of those values to the verifier. Furthermore, for each individual gate in the circuit, the prover computes a non-interactive zero-knowledge *gate consistency* proof: the proof should convince the verifier that the commitments to the input and the output wires are consistent with the description of the gate. That is, the value on the output wire has been honestly computed by the prover. Finally, the prover decommits the values on the output wires of the circuit.

We consider two methods for computing the gate consistency proofs. These methods correspond to alternative characterizations of boolean gates. One characterization is based on linear relations between the inputs and the outputs; this is essentially the technique underlying the method of [20]. The second method is based on quadratic equations, and is essentially the “classical” technique underlying the work of [10] and others. The equations and their use are detailed later in the paper. We refer to the resulting proof as the **LEq** method and the **QEeq** method respectively. For each method in turn we consider gate consistency proofs in the random oracle model and in the common reference string model. Note that the **LEq** method was originally considered only in the context of pairing based NIZK proofs in the CRS, whereas the **QEeq** method was originally considered in the context of ROM based proofs derived from Σ -protocols; we however look at both techniques in both settings. The proofs in the random oracle model are obtained through a transform attributed to Blum (via [3]) from standard Σ -protocols for the relations associated to the gate equations. The proofs in the CRS model use the techniques of Groth et al. [20,21]. We note that although the techniques for which Groth et al. give full details in their papers are for the case of symmetric pairings we choose an implementation based on asymmetric pairings. These pairings are more efficient and are amenable to a series of known optimizations.

We present a number of optimizations of the basic proof constructions for the gate consistency proofs, which apply in both the ROM and the CRS implementations, and we provide a practical performance comparison of the four resulting algorithms. Also by implementing the witness-indistinguishable non-interactive proofs recently introduced by Groth and Sahai [21] our evaluation should be of interest independent of the particular application to **BCP**, since these proofs are used in other settings as well. In particular we notice that constructing the Groth–Sahai proofs in the CRS model is only 7-9 times less efficient than the equivalent proofs derived from the Blum transform being applied to a Σ -protocol in the random oracle model. However, the relative performance of the verifier is much worse, being around 10-20 times less efficient for the CRS based proofs compared to the ROM based proofs, when batch verification is used. This lack of performance is due to the verification algorithm in the Groth–Sahai situation requiring many pairing evaluation.

We evaluate the efficiency of our implementation in the case of two circuits with potential practical applications. The first circuit compares two 32-bit committed integers (a functionality needed for example in proving a desired relation between two committed bids in auctions). The second circuit takes as input a

128-bit key K and a 128-bit public message M and computes $\text{AES}(K, M)$. The associated **BCP** is thus to show that a particular ciphertext C is obtained by enciphering a public message M , with a key K that is held by a publicly available commitment. The first circuit is in the range of hundreds of gates, whereas the second one is in the range of tens of thousands of gates. To cope with this large size we have developed a series of optimizations that reduce the overall cost of the computation.

The paper is organized as follows. We start with an overview of binary and arithmetic circuits in Section 2. We describe the two solutions based on different characterizations for the boolean gates of a circuit in Section 3. The different ways of implementing the gate consistency proofs for both the CRS and the ROM, together with theoretical estimates for the resulting running time when these proofs are plugged in the protocols of Section 3 are in Section 4. We present batch verification techniques for the different proofs in Section 5. We conclude with the practical experimental results in Section 6.

2 Binary and Arithmetic Circuits

To facilitate the description of the proofs that we consider, we introduce some notation regarding binary and arithmetic circuits.

BINARY CIRCUITS. We write $W = \{w_i\}$ for the set of wires of the circuit. These are variables with values in $\{0, 1\}$. We write $G = \{g_i\}$ for the set of gates of the circuit. We only consider binary gates (gates with two inputs and one output), so each gate $g_i \in G$ is given by a triple

$$g_i = (I_i, o_i, T_i)$$

with $I_i \in W \times W$ a pair of elements of W that define the input wires of g_i and $o_i \in W$ is an element of $W \setminus I_i$ that defines the output wire. T_i is the table of a mapping from $\{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$, that defines the output of the gate for each possible input. By a slight abuse of notation we may occasionally write $T_i(w_i, w_j)$ for the output of the gate for specific values for the input wires.

A *binary circuit* $C = \{W, G\}$ consists of a set of wires and of a set of gates. We identify special subsets of wires $O \subset W$ that defines the output wires of the circuit and $I \subset W$ that defines the input wires of the circuit. We require that for any $w_i \in I$ it holds that $w_i \neq o_j$ whenever $g_j = (I_j, o_j, T_j) \in G$ for some g_j . Furthermore, we insist that the circuit can be evaluated in the sense that there are no cyclic dependencies. In particular this requirement implies that the circuit C can be topologically ordered: there exists an order on the gates in G such that evaluation of a gate g only depends on the output wires of the gates before g . By this we mean that we order the set G such that for each $i \in \{0, \dots, |G|\}$ we can define a set A_i such that

- $A_0 = I$.
- $A_{|G|} = W$.
- For all $g_j = (I_j, o_j, T_j) \in G$ we have $I_j \subset A_{j-1}$.
- For all $g_j = (I_j, o_j, T_j) \in G$ we set $A_j = A_{j-1} \cup \{o_j\}$.

The intuition in the above description is that set A_j is the set of wires for which their associated value had already been determined, before gate g_j is evaluated.

An important observation [20] useful in proving statements about gates (and therefore circuits) is that any binary gate $g = ((w_i, w_j), w_k, T)$ can be represented by a linear arithmetic equation $L(w_i, w_j, w_k)$ over the input and output wires in the sense that $L(w_i, w_j, w_k) \in \{0, 1\}$ if and only if $T(w_i, w_j) = w_k$. For example, for the NAND gate, i.e. the gate that outputs 1 if and only if at least one of its input wires is 0, the associated linear equation is:

$$w_i + w_j + 2(w_k - 1) \in \{0, 1\}.$$

In the remainder of the paper we write $L_r(w_i, w_j, w_k)$ for the linear equation associated to gate g_r .

ARITHMETIC CIRCUITS. An arithmetic circuit over a finite field \mathbb{F}_q is defined in a similar way. The difference is that wires assume values in \mathbb{F}_q instead of $\{0, 1\}$ and the gates can only evaluate either addition/subtraction or multiplication (over \mathbb{F}_q) of their input values. In other words, the truth table of the gate is replaced by a polynomial function of the input wires.

It is clear that any arithmetic circuit can be converted into a binary circuit by replacing the arithmetic gates with their binary circuit equivalents. We will now show that every binary circuit can also be represented as an arithmetic circuit. Suppose we have a binary gate with two binary inputs w_i, w_j and one binary output w_k . Then there is always a quadratic equation such that

$$w_k = Q(w_i, w_j).$$

For example a NAND gate is described by the equation:

$$w_k = 1 - w_i \cdot w_j.$$

Therefore, every binary gate on two inputs can be represented by a quadratic equation in the inputs. In the remainder of the paper we write Q_j for the quadratic equation that describes gate g_j .

The two methods for representing gates yield two basic methods for proving **BCP**. Both methods also require commitments schemes. For message space \mathcal{M} and randomness space \mathcal{R} , we write $\text{comm}_*(x, r) \in \mathcal{Comm}$, for the commitment to $x \in \mathcal{M}$ using randomness $r \in \mathcal{R}$. We require that sets \mathcal{M}, \mathcal{R} , and \mathcal{Comm} be abelian groups and that the commitment scheme when regarded as a function $\text{comm}_* : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{Comm}$ be homomorphic:

$$\text{comm}_*(m_1 + m_2; r_1 + r_2) = \text{comm}_*(m_1, r_1) + \text{comm}_*(m_2, r_2).$$

We consider several instantiations of the commitment scheme and we differentiate them by replacing $*$ with representative mnemonics.

3 Two Solutions for the Binary Circuit Problem

In this section, we describe two methods for **BCP**. Both methods are based on the idea presented in the introduction: the prover computes commitments to

the values assumed by the wires of the circuits (given the value of the input x) and then, for each gate produces a *gate consistency proof* which shows that the commitments to the values of the input wires and the output wires are in the desired relation. The difference between the methods lies in the method used to prove the consistency of the commitments to the wires of a gate. We use two methods, one based on the characterization of gates via linear equations and the other one based on quadratic equations, presented in the previous section.

We describe the methods at a high level; we do not specialize the commitment schemes that are used and we do not detail how the prover computes the proofs for the relation between the committed values on the wires. In the next section we specify the precise methods through which the prover ensures gate consistency, and for each case we determine the computational requirements for the prover and the verifier.

3.1 Linear Equation Based Method (LEq)

As already mentioned this method underlies the techniques in [20], however we present it in generality as we will be examining its efficiency in the ROM as well as the CRS. Recall that to each gate $g_r = ((w_i, w_j), w_k, T_r)$ one can associate a linear equation $L_r(w_i, w_j, w_k) = aw_i + bw_j + cw_k + d$ such that if w_i, w_j are in $\{0, 1\}$, then $L_r(w_i, w_j, w_k) \in \{0, 1\}$ if and only if $T_r(w_i, w_j) = w_k$. Proving consistency of committed values for the wires w_i, w_j and w_k can be ensured in two steps: show that each of the commitments is a bit commitment, and then show that the desired linear equation between the committed bits holds. The last step can be implemented when the commitment scheme is homomorphic via another proof that a particular commitment is to a bit. The algorithm of the prover that we give below uses a commitment $\text{comm}_{\star 0} = \text{comm}_{\star}(1, 0)$ to the value 1.

1. Given circuit $C = (W, G)$, and a given input x (which determines the values for the input wires in I) the prover first determines the value of each wire $w_i \in W$.
2. The prover computes a commitment $\text{comm}_{\star i} = \text{comm}_{\star}(w_i, r_i)$ to each of the value of $w_i \in W$.
3. For each $w_i \in W$, the prover produces a proof π_i that each commitment $\text{comm}_{\star i}$ opens to an element in the set $\{0, 1\}$.
4. If $L_r(w_i, w_j, w_k) = aw_i + bw_j + cw_k + d$ is the linear equation associated with gate g_r , the prover computes

$$\text{comm}_{\star r}' = a \cdot \text{comm}_{\star i} + b \cdot \text{comm}_{\star j} + c \cdot \text{comm}_{\star k} + d \cdot \text{comm}_{\star 0},$$

for each gate g_r . Here the prover uses the homomorphic property of the commitment scheme to compute the randomness underlying $\text{comm}_{\star r}'$. For each $g_r \in G$ the prover produces a proof π_r' that commitment $\text{comm}_{\star r}'$ opens to an element in $\{0, 1\}$.

5. Finally, the prover outputs the decommitment values for the circuit's output wires.

The proof that the prover outputs consists of commitments $\text{comm}_{\star i}$ for $i = 0, \dots, |W|$, the proofs π_i (for $i \in W$) that the commitments associated to wires are to valid values (i.e. 0,1), the proofs π'_r (for $r \in G$) that the gates satisfy their associated linear equation, and the decommitments to the wires in O . The verifier computes the commitments $\text{comm}'_{\star r}$ on his own (for each $1, \dots, |G|$), verifies all the individual proofs, and correctness of the decommitted values. The total communication required is therefore $|W|$ commitments, $|W| + |G|$ proofs and $|O|$ decommitment information.

3.2 The Quadratic Equation Based Method (QE_Q)

The second method that we investigate uses the representation of the output of a gate as a quadratic polynomial of its input wires. This is the classical method underlying the ROM methods in many papers such as [10]. However, our presentation involves a number of optimizations; in particular note below our division of the sets of quadratic monomials into subsets. When we instantiate the proofs this forms an important optimization. However, in the ROM this turns out to only produce an optimization when batch verification techniques are not desirable (such techniques produce more efficient verification, but less efficient provers and also increase the size of the underlying proof).

At a high level here the prover starts out with commitments to the bits on the input wires of the circuits and uses the homomorphic property of the commitment scheme to compute commitments to the remaining ones. Notice that since the equation is quadratic (and thus not sufficiently compatible with the commitment scheme to use the method of the previous section) one also needs to supply commitments to the monomials that occur in the gate equation and a proof that these commitments are well formed. Given these, the verifier can check the validity of the computation on its own.

We implemented this method with a number of significant optimizations. To give the details we need a little more notation. We first identify the set of quadratic monomials that occur in the quadratic equations that define the gates of the circuit. We define

$$\mathcal{M}on = \{(i, j) : g_k = ((x_i, x_j), x_k, T_k) \in G, \}\,.$$

The set $\mathcal{M}on$ represents the set of products of two terms which are needed to derive the output of a gate.

We divide $\mathcal{M}on$ into t disjoint subsets $\mathcal{M}on_1, \dots, \mathcal{M}on_t$, where each subset $\mathcal{M}on_k$ is indexed by a value s_k such that

$$\forall (i, j) \in \mathcal{M}on_k \text{ either } s_k = i \text{ or } s_k = j.$$

This is done in a way so as to heuristically minimize the number of disjoint subsets. Each subset of $\mathcal{M}on$ represents a combination of proofs which can be executed together. Thus by minimizing the number of disjoint subsets, we minimize the number of proofs which need to be executed.

We can now give the second method for solving **BCP**. We use a commitment $\text{comm}_{\star 0} = \text{comm}_{\star}(1, 0)$ to the value 1.

1. Given circuit $C = (W, G)$, and a given input x (which determines the values for the input wires in I) the prover first determines the value of each wire $w_i \in W$.
2. The prover computes a commitment $\text{comm}_{\star i} = \text{comm}_{\star}(w_i, r_i)$ to each of the value of $w_i \in I$.
3. For each $w_i \in I$, the prover produces a proof π_i that each commitment $\text{comm}_{\star i}$ opens to an element in the set $\{0, 1\}$.
4. For every element $(i, j) \in \mathcal{M}on$, the prover commits to the product $w_i w_j$, via $\text{comm}_{\star i,j} = \text{comm}_{\star}(w_i \cdot w_j, r_{i,j})$.
5. For each gate g represented by the quadratic equation $w_k = Q(w_i, w_j) = aw_i \cdot w_j + bw_i + cw_j + d$, he computes a commitment $\text{comm}_{\star k}$ to w_k via

$$\text{comm}_{\star}(w_k, r_k) = a \cdot \text{comm}_{\star i,j} + b \cdot \text{comm}_{\star i} + c \cdot \text{comm}_{\star j} + d \cdot \text{comm}_{\star 0}$$

6. For $l = 1, \dots, t$ the prover generates a proof π'_l that the commitments $\text{comm}_{\star i,j}$ to the products in $\mathcal{M}on_l$ are consistent with the wires' commitments.
7. Finally, the prover outputs the decommitment values for the circuit's output wires.

The output of the prover consists of the commitments $\text{comm}_{\star i}$ for $i = 0, \dots, |I|$, the commitments to the products used, the proofs π_i for $i \in I$ and the proofs π'_l for $l \in 1, \dots, t$, as well as decommitments to the output wires. The verifier can then compute $\text{comm}_{\star k}$ himself for $k = |I|, \dots, |W|$ and verify all the proofs. We are thus required to transmit $|I| + |\mathcal{M}on|$ commitments, $|I|$ proofs that each commitment is to a bit, t monomial consistency proofs, and the $|O|$ decommitments.

4 Proofs for Relations between Committed Values

The methods described in Section 3 make use of various subproofs: proofs that a committed value is either 0 or 1, that a committed value is product of two other committed values, etc. In this section, we detail the proofs and the particular commitment schemes that we use in our implementation. We then estimate the efficiency of the protocol(s) that result by plugging in these particular proofs in the general solutions of the previous section.

4.1 Relations between Committed Values

In this subsection, we summarize the statements for which we require zero-knowledge proofs. These proofs essentially allow us to prove that certain commitments hold a bit value, and that certain relations between committed bits hold. Using a commitment scheme that is homomorphic, we can then prove that each gate is computed correctly and reveal the values on the output wires. Below, we use $\text{comm}_{\star}(\cdot, \cdot)$ for the commitment algorithm. The statements for which we need proofs are as follows:

The first kind of statement specifies that the prover knows the opening of a commitment A , and the value committed to is 0 or 1. In Camenisch–Stadler notation [9], this statement is

$$\text{POK} \{(x) : \exists r_x \in \mathbb{Z}_q \wedge A = \text{comm}_\star(x, r_x) \wedge x \in \{0, 1\}\},$$

Since we use such proofs to show that commitments to values for wires are well-formed, we refer to this statement as **WFComm**. The second kind of statements involves a commitment A to some x , and $2 \cdot s$ commitments B_i and C_i to some values $y^{(i)}$ and $z^{(i)}$ respectively, for $i = 1, 2, \dots, s$. These commitments are known to both the prover and the verifier. The prover wants then to convince the verifier that he knows $x, r_x, y^{(i)}, r_y^{(i)}, z^{(i)}$ and $r_z^{(i)}$, such that

$$A = \text{comm}_\star(x, r_x) \bigwedge_{i=1}^s S_i$$

where S_i is the statement

$$B_i = \text{comm}_\star(y^{(i)}, r_y^{(i)}) \quad \wedge \quad C_i = \text{comm}_\star(z^{(i)}, r_z^{(i)}) \quad \wedge \quad z^{(i)} = x \cdot y^{(i)}.$$

That is, for each i , the value committed in C_i is the product of the values committed in A and B_i . Since the statement regards the well-formedness of certain products we refer to this statement as **WFProd**.

4.2 Non-interactive Proofs in the Random Oracle Model

The first instantiation that we investigate uses Pedersen commitments and non-interactive proofs in the random oracle model. The proofs are obtained from standard Σ -protocols for the relations that we prove by using a transform proposed by Blum. All of the following proofs use standard techniques, and many are essentially “folklore”.

Let $(\mathbb{G}, +)$ be a finite abelian group of prime order p , P a generator of \mathbb{G} and Q an arbitrary element in \mathbb{G} such that no party knows discrete logarithm of Q with respect to P . Also, assume that $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ is a random oracle. Our random oracle implementation uses the Pedersen commitment scheme that allows to commit to an element $x \in \mathbb{Z}_p$, using randomness $r \in \mathbb{Z}_p$ as follows:

$$\text{comm}_{\text{ROM}} : \begin{cases} \mathbb{Z}_p \times \mathbb{Z}_p \longrightarrow \mathbb{G} \\ (x, r) \longmapsto x \cdot P + r \cdot Q \end{cases}$$

Proofs for WFComm in the RO Model: To prove knowledge of a committed value that is either 0 or 1, we use the standard OR-proofs [11] obtained from Schnorr proofs [26]. The computation of the prover is summarized below.

- If $x = 0$, the prover proceeds as follows:
 - Select $w, z_1, e_1 \in_R \mathbb{Z}_p$ and compute $A_0 = w \cdot Q$ and $A_1 = z_1 \cdot Q - e_1 \cdot (A - P)$.

- Compute $e = H(A_0 \| A_1)$.
- Compute $e_0 = e - e_1$ and $z_0 = w - r_x \cdot e_0$.
- If $x = 1$, the prover proceeds as follows:
 - Select $w, z_0, e_0 \in_R \mathbb{Z}_p$ and compute $A_0 = z_0 \cdot Q + e_0 \cdot A$ and $A_1 = w \cdot Q$.
 - Compute $e = H(A_0 \| A_1)$.
 - Compute $e_1 = e - e_0$ and $z_1 = w + r_x \cdot e_1$.

The proof in both cases is given by the tuple (e, e_0, e_1, z_0, z_1) . To verify the proof, the verifier computes the two values

$$A'_0 = z_0 \cdot Q + e_0 \cdot A \text{ and } A'_1 = z_1 \cdot Q - e_1 \cdot (A - P)$$

and then verifies that $e = e_0 + e_1$ and $e = H(A'_0 \| A'_1)$.

In the above description, proof generation requires one multiplication and one “double multiplication”. We write this as: $1 \cdot \mathbb{G} + 1 \cdot \mathbb{G}^2$. We preserve this convention in the remainder of the paper. Proof verification requires $2 \cdot \mathbb{G}^2$ operations, i.e. two double multiplications.

Proof for WFProd in the RO Model: Next we give the protocol that we use in order to show the relation

$$A = \text{comm}_{\text{ROM}}(x, r_x) \bigwedge_{i=1}^s S_i$$

where S_i is the statement

$$B_i = \text{comm}_{\text{ROM}}(y^{(i)}, r_y^{(i)}) \quad \wedge \quad C_i = \text{comm}_{\text{ROM}}(z^{(i)}, r_z^{(i)}) \quad \wedge \quad z^{(i)} = x \cdot y^{(i)}.$$

Our proof is based on executing, s -times in parallel, the serial proof protocol from [24], which is itself based on Okamoto’s identification protocol [25]. Hence, the prover executes the following statements:

- Select $a, b \in_R \mathbb{Z}_p$ and compute $A_0 = a \cdot P + b \cdot Q$ and for $i = 1, \dots, s$ $A_i = a \cdot B_i + b \cdot Q$
- Compute $e = H(A_0 \| A_1 \| \dots \| A_n)$.
- Computes $s_0 = a + e \cdot x$, $s_1 = b + e \cdot r_x$ and for $i = 1, \dots, s$, $s_{i+1} = b + e \cdot (r_z^{(i)} - r_y^{(i)} \cdot x)$.

The proof is given by the tuple $(e, s_0, s_1, \dots, s_{n+1})$. To verify the proof the verifier computes

$$\begin{aligned} A'_0 &= s_0 \cdot P + s_1 \cdot Q - e \cdot A \\ A'_i &= s_0 \cdot B_i + s_{i+1} \cdot Q - e \cdot C_i \text{ for } i = 1, \dots, s. \end{aligned}$$

Then the verifier accepts the proof if $e = H(A'_0 \| A'_1 \| \dots \| A'_{n+1})$.

Proof generation requires $(s+1) \cdot \mathbb{G}^2$ operations (although it might be simpler to compute this using $(s+3) \cdot \mathbb{G}$ operations.) Verification requires $(s+1) \cdot \mathbb{G}^3$ operations.

Complexity of LEq in the RO Model: Using the above proofs for gates in the general protocol of Section 3.1 leads to the following computational effort for the prover.

1. Negligible
2. $(|W| + 1) \cdot \mathbb{G}^2$.
3. Here we execute $|W|$ times the proof for **WFComm**. The computation time is $|W| \cdot (\mathbb{G} + \mathbb{G}^2)$.
4. The prover computes the values of $\text{comm}_{\text{ROM}}'$; since a, b, c, d are “small” this computation is relatively cheap so we ignore this cost. The prover computes $|G|$ proofs for **WFComm** which requires $|G| \cdot (\mathbb{G} + \mathbb{G}^2)$ computation time.
5. Negligible.

Hence, the total cost for the prover (obtained by summing the cost for each step above) is

$$(|W| + |G|) \cdot \mathbb{G} + (2 \cdot |W| + |G| + 1) \cdot \mathbb{G}^2.$$

In a similar manner we obtain that the verifier performs

$$(2 \cdot |W| + 2 \cdot |G| + |O| + 1) \cdot \mathbb{G}^2$$

operations.

Complexity of QEeq in the RO Model: When we use the proofs developed in this section in conjunction with the general protocol of Section 3.2 the computation cost for the prover can be estimated as follows:

1. Negligible.
2. $|I| \cdot \mathbb{G}^2$.
3. This requires $|I|$ invocations of **WFComm**. This requires $|I| \cdot (\mathbb{G} + \mathbb{G}^2)$ operations.
4. $|\mathcal{M}on| \cdot \mathbb{G}^2$.
5. This is negligible cost as the values of a, b, c and d are very small.
6. For each $i \in [1, \dots, t]$ this requires an invocation of **WFProd** with $s = |\mathcal{M}on_i|$. Thus the cost of this step for the prover is $\sum_{i=1}^t (|\mathcal{M}on_t| + 3) \cdot \mathbb{G} = (|\mathcal{M}on| + 3 \cdot t) \cdot \mathbb{G}$.
7. Negligible.

Thus the total cost for the prover is

$$(|\mathcal{M}on| + |I| + 3 \cdot t) \cdot \mathbb{G} + (|\mathcal{M}on| + 2 \cdot |I| + 1) \cdot \mathbb{G}^2,$$

and the cost for the verifier is

$$(|I| + |O|) \cdot \mathbb{G}^2 + (|\mathcal{M}on| + t) \cdot \mathbb{G}^3.$$

4.3 Non-interactive Proofs in the Common Reference String Model

We also study an efficient implementation in the CRS model. The implementation is based on the techniques proposed by Groth et al. [20,21]. Their techniques crucially rely on groups that benefit from cryptographic pairings and are presented in three different settings. Each setting is based on a particular assumption which translates in particular types of elliptic curves. The assumptions and the pairing functions that they consider are as follows.

- **Subgroup Decision Problem:** The implementation uses a symmetric pairing over groups of unknown order, such as those in [5].
- **Decision Linear Problem:** The implementation uses a symmetric pairing over a group of known prime order q . These are often called Type-1 pairings [16].
- **Symmetric External Diffie–Hellman Problem:** The implementation uses an asymmetric pairing over a group of known prime order q in which the external DDH problem is hard in both domain groups. These are often called Type-3 pairings [16].

We note that although [20] already presents a NIZK proof of circuit satisfiability for the first two cases above, it does not treat the third case. Clearly however, it is this third case that should lead to more practical result since the pairings used in this setting can benefit from techniques that lead to significantly improved implementations such as sextic twists, the ate-pairing [22]. Furthermore, security scales better in the asymmetric pairing setting [16]. The proofs that we implement use this setting and we give the details below.

We comment that our presentation departs from that in [21] in terms of notation. The notation of [21] is sufficiently comprehensive to deal with all of the three cases considered. Instead, in this paper we use notation tailored for the third case. The result is that our presentation exhibits the structure of their proofs in much clearer way for the case of asymmetric pairings, which are the pairings that we use in our implementation. In addition, in [21] the proofs are presented for four possible types of equations, we will only be interested in equations between commitments to integer values, which again allows us to simplify the notation from [21] somewhat. In the sequel we detail the proofs that we have implemented and, when appropriate, we highlight how our presentation departs from the one in the original paper.

We consider a Type-3 pairing, $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, between finite groups of prime order p , such that the Diffie–Hellman problem is hard in both \mathbb{G}_1 and \mathbb{G}_2 . We assume generators P_i of \mathbb{G}_i are given and we set

$$\mathbb{B}_1 = \mathbb{G}_1 \times \mathbb{G}_1, \quad \mathbb{B}_2 = \mathbb{G}_2 \times \mathbb{G}_2, \quad \mathbb{B}_T = \mathbb{G}_T^4.$$

We denote integer variables by lower case letters, elements in \mathbb{G}_i by upper case letters, elements in \mathbb{G}_T by lower case Greek letters, and elements of \mathbb{B}_i by calligraphic letters \mathcal{U}, \mathcal{V} . We keep Groth/Sahai’s convention that a proof is given by a pair $(\Pi, \Theta) \in \mathbb{B}_2 \times \mathbb{B}_1$, but we use capital Greek letters for these two elements,

the other exception relating to namely of functions. In these exceptional cases type inference should be clear from the context. We index by subscripts 1 and 2 elements of \mathbb{G}_1 or \mathbb{G}_2 , this is the major benefit of our notation as it is now simpler to see to which of the two groups in the asymmetric pairing situation an element belongs to. Vectors are indicated by underlying them, as in \underline{a} and we write $\underline{a}^{(i)}$ for the i 'th position of vector \underline{a} . Sets/modules/groups etc will be denoted by blackboard font, as in $\mathbb{B}, \mathbb{Z}, \mathbb{G}$.

We can now describe the basic Groth and Sahai proofs which we use.

Setup

Generate $a_1, a_2, t_1, t_2 \in \mathbb{Z}_p$ and for $i = 1, 2$ define

$$Q_i = [a_i]P_i, \quad U_i = [t_i]P_i, \quad V_i = [t_i]Q_i.$$

We now set

$$\mathcal{U}_i = (P_i, Q_i) \in \mathbb{B}_i, \quad \mathcal{V}_i = t_i \cdot \mathcal{U}_i = (U_i, V_i) \in \mathbb{B}_i, \quad \mathcal{W}_i = \mathcal{V}_i + (\mathcal{O}, P_i) \in \mathbb{B}_i.$$

The functions that we define next extend naturally to vectors of input values in a component-wise manner. We let

$$F : \left\{ \begin{array}{ccc} \mathbb{B}_1 \times \mathbb{B}_2 & \longrightarrow & \mathbb{B}_T \\ (X_1, Y_1), (X_2, Y_2) & \longmapsto & (\hat{t}(X_1, X_2), \hat{t}(X_1, Y_2), \hat{t}(Y_1, X_2), \hat{t}(Y_1, Y_2)) \end{array} \right.$$

Note that unlike in [21] we present elements of \mathbb{B}_T as vectors of length four, as opposed to 2×2 matrices. This emphasizes that the group operation in \mathbb{B}_T is component wise multiplication and not matrix multiplication. Since the underlying pairing \hat{t} is bilinear, it follows that map F is also bilinear. We also define the maps

$$\iota'_i : \left\{ \begin{array}{ccc} \mathbb{Z}_p & \longrightarrow & \mathbb{B}_i \\ x & \longmapsto & x \cdot \mathcal{W}_i, \end{array} \right.$$

$$\iota'_T : \left\{ \begin{array}{ccc} \mathbb{Z}_p & \longrightarrow & \mathbb{B}_T \\ z & \longmapsto & (1, 1, 1, \hat{t}(P_1, P_2)^z) \end{array} \right.$$

The Commitment Scheme

We now define a commitment scheme to elements $x \in \mathbb{Z}_p$, using randomness $r \in \mathbb{Z}_p$ as follows:

$$\text{comm}_{\text{CRS}_i} : \left\{ \begin{array}{ccc} \mathbb{Z}_p \times \mathbb{Z}_p & \longrightarrow & \mathbb{B}_i \\ (x, r) & \longmapsto & x \cdot \mathcal{W}_i + r \cdot \mathcal{U}_i \end{array} \right.$$

Note that we have $\text{comm}_{\text{CRS}_i}(x; r) = (xU_i + rP_i, x(V_i + P_i) + rQ_i)$. We will also need to make use of the “combined” commitment which we define as

$$\text{comm}_{\text{CRS}} : \left\{ \begin{array}{ccc} \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p & \longrightarrow & \mathbb{B}_1 \times \mathbb{B}_2 \\ (x, r_1, r_2) & \longmapsto & (\text{comm}_{\text{CRS}_1}(x, r_1), \text{comm}_{\text{CRS}_2}(x, r_2)) \end{array} \right.$$

One can think of these commitments schemes as analogues of Pederson commitments in the groups \mathbb{B}_i and $\mathbb{B}_1 \times \mathbb{B}_2$, with appropriately chosen generators, due to the form $x \cdot \mathcal{W}_i + r \cdot \mathcal{U}_i$.

The Proof

Assume that values x_i are committed to by selecting random vectors $\underline{r}_i \in \mathbb{Z}_p^{n_i}$ and then forming the vector of commitments $\underline{\mathcal{C}}_i$ via

$$\underline{\mathcal{C}}_i^{(j)} = \text{comm}_{\text{CRS}_i}(\underline{x}_i^{(j)}; \underline{r}_i^{(j)}) \text{ for } j = 1, \dots, n_i, i = 1, 2.$$

Groth and Sahai allow one to prove that given commitments as above, the prover knows an opening of the commitments as variables $\underline{x}_1 \in \mathbb{Z}_p^{n_1}$, $\underline{x}_2 \in \mathbb{Z}_p^{n_2}$ such that the following equation holds:

$$\underline{a}_2 \cdot \underline{x}_2^t + \underline{x}_1 \cdot \underline{a}_1^t + \underline{x}_1 \cdot \Gamma \cdot \underline{x}_2^t = t$$

where $\underline{a}_i \in \mathbb{Z}_p^{n_i}$, $t \in \mathbb{Z}_p$ and $\Gamma \in \mathbb{Z}_p^{n_1 \times n_2}$.

A proof is simply given by two elements $(\Theta, \Pi) \in \mathbb{B}_1 \times \mathbb{B}_2$. These are formed by computing, for some random value $s \in \mathbb{Z}_p$.

$$\begin{aligned} \Theta &= \underline{r}_2 \cdot \iota'_1(\underline{a}_2)^t + \underline{r}_2 \cdot \Gamma^t \cdot \iota'_1(\underline{x}_1)^t + s \cdot \mathcal{U}_1, \\ \Pi &= \underline{r}_1 \cdot \iota'_2(\underline{a}_1)^t + \underline{r}_1 \cdot \Gamma \cdot \iota'_2(\underline{x}_2)^t + (\underline{r}_1 \cdot \Gamma \cdot \underline{r}_2^t - s) \cdot \mathcal{U}_2. \end{aligned}$$

Verification

To verify a proof the prover checks whether

$$\begin{aligned} \prod_{i=1}^{n_1} F \left(\underline{C}_1^{(j)}, \iota'_2(\underline{a}_1^{(j)}) + \sum_{k=1}^{n_2} \Gamma_{j,k} \cdot \underline{C}_2^{(k)} \right) \cdot \prod_{j=1}^{n_2} F \left(\iota'_1(\underline{a}_2^{(j)}), \underline{C}_2^{(j)} \right) \\ = \iota'_T(t) \cdot F(\mathcal{U}_1, \Pi) \cdot F(\Theta, \mathcal{U}_2). \end{aligned}$$

We now examine the communication and computational cost of the above method for the equation specified above. The size of the commitments is equal to $2 \cdot (n_1 \cdot \mathbb{G}_1 + n_2 \cdot \mathbb{G}_2)$. To compute the commitments requires $4 \cdot (n_1 \cdot \mathbb{G}_1 + n_2 \cdot \mathbb{G}_2)$ operations. Whilst, the size of a proof for one equation is equal to $2 \cdot (\mathbb{G}_1 + \mathbb{G}_2)$. To compute a proof depends on the number of non-zero entries in $\underline{a}_1, \underline{a}_2$ and Γ . An upper bound is given by

$$2 \cdot (n_1 + n_2 + 1) \cdot (\mathbb{G}_1 + \mathbb{G}_2).$$

To evaluate the cost for verifying a proof we assume that the most expensive cost of the basic operations, i.e. group scalar multiplication or pairing, is the pairing operation. This might not always be true in practice, but it should give a good idea. Each F operation requires four pairings. However, each product of F values can be implemented using the standard ‘‘product of pairings’’ trick[17], the cost of evaluating a product of t application of F (resp. pairings) we shall denote by F^t (resp. P^t). Thus to verify a proof we require at most $F^{n_1+n_2+2}$ operations, or equivalently $4 \cdot P^{2 \cdot n_1+n_2+2}$. Clearly, the exact number depends on the number of non-zero entries in $\underline{a}_1, \underline{a}_2$ and Γ .

Particular Instantiations

Our solution uses proofs for the following three instantiations for the equation above (the proofs are presented in the appendix). For all of these proofs there are publicly known commitments to three values $\underline{x}_1^{(1)}$, $\underline{x}_1^{(2)}$ and $\underline{x}_2^{(1)}$. The commitments are computed via:

$$\begin{aligned}\underline{C}_1^{(1)} &= \text{comm}_{\text{CRS}1}(\underline{x}_1^{(1)}, \underline{r}_1^{(1)}) = \underline{x}_1^{(1)} \cdot \mathcal{W}_1 + \underline{r}_1^{(1)} \cdot \mathcal{U}_1 \in \mathbb{B}_1, \\ \underline{C}_1^{(2)} &= \text{comm}_{\text{CRS}1}(\underline{x}_1^{(2)}, \underline{r}_1^{(2)}) = \underline{x}_1^{(2)} \cdot \mathcal{W}_1 + \underline{r}_1^{(2)} \cdot \mathcal{U}_1 \in \mathbb{B}_1, \\ \underline{C}_2^{(1)} &= \text{comm}_{\text{CRS}2}(\underline{x}_2^{(1)}, \underline{r}_2^{(1)}) = \underline{x}_2^{(1)} \cdot \mathcal{W}_2 + \underline{r}_2^{(1)} \cdot \mathcal{U}_2 \in \mathbb{B}_2.\end{aligned}$$

Recall our convention that a subscript on a group element denotes which of the two domains of the pairing it belongs to. The proofs are for the following equations:

- **Product:** $x_1^{(2)} = x_1^{(1)} \cdot x_2^{(1)}$.
- **BitProof:** $x_1^{(1)} = x_1^{(1)} \cdot x_2^{(1)}$.
- **Equality:** $x_1^{(1)} = x_2^{(1)}$.

Using proofs for the above equations, to be found in the Appendix, we develop instantiations for **WFComm** and **WFProd** as follows:

Proofs of WFComm in the CRS Model: We actually require two such proofs, the first is to prove the following statement:

$$\text{POK} \{(x) : \exists r_x \in \mathbb{Z}_q \wedge C_1 = \text{comm}_{\text{CRS}1}(x, r_x) \wedge x \in \{0, 1\}\}.$$

Here we execute the following steps:

- Compute $C_2 = \text{comm}_{\text{CRS}2}(x, r'_x)$.
- Show that $x \cdot (x - 1) = 0$ by using a proof for **BitProof**
- Show that C_1 and C_2 are commitments to the same value via a proof for **Equality**.

The cost of producing this proof is therefore equal to $2 \cdot \mathbb{B}_1^2 + 3 \cdot \mathbb{B}_2^2 = 4 \cdot \mathbb{G}_1^2 + 6 \cdot \mathbb{G}_2^2$, whereas the cost of verification requires $8 \cdot P^4$. It may at first sight appear, from examining the verification equations associated to **BitProof** and **Equality** that a speed-up may be possible by exploiting that one of the F terms is the same in the two equations. However, because we are using products of pairings to perform the verification, this observation does not provide us with any efficiency gain.

For the commitment scheme comm_{CRS} things are a bit different. For the same statement

$$\text{POK} \{(x) : \exists r_x, r'_x \in \mathbb{Z}_q \wedge C = \text{comm}_{\text{CRS}}(x, r_x, r'_x) \wedge x \in \{0, 1\}\}.$$

if we can ensure somehow that the commitment is well-formed it is sufficient to only execute a proof for **BitProof**. One way to ensure that the commitment is

well formed is through a proof for **Equality**. This is an important observation for justifying the optimizations that we propose later in the paper. For evaluating efficiency in the case of commitment scheme comm_{CRS} we refer to **WFComm** as the proof which only executes a proof for **BitProof**. We ensure that through separate proofs that the commitments involved are well formed (if this cannot be deduced from other information). In this case, each of these proofs require $\mathbb{B}_1^2 + \mathbb{B}_2^2 = 2\mathbb{G}_1^2 + 2\mathbb{G}_2^2$ operations to produce them and $4 \cdot P^4$ operations to verify them.

Proofs for WFProd in the CRS Model: We implement a proof for the statement

$$\text{POK} \left\{ (\cdot) : A = \text{comm}_{\text{CRS}_1}(x, r_x) \bigwedge_{i=1}^s S_i \right\}$$

where S_i is the statement

$$B_i = \text{comm}_{\text{CRS}_1}(y^{(i)}, r_y^{(i)}) \quad \wedge \quad C_i = \text{comm}_{\text{CRS}_1}(z^{(i)}, r_z^{(i)}) \quad \wedge \quad z^{(i)} = x \cdot y^{(i)}.$$

via the following steps:

- Compute $A' = \text{comm}_{\text{CRS}_2}(x, r'_x)$.
- Show that A and A' are commitments to the same value by using the proof for **Equality**.
- Execute $z^{(i)} = x \cdot y^{(i)}$ for $i = 1, \dots, s$ by executing s proofs for **Product**.

The total cost for producing the proof is $2 \cdot (1 + s) \cdot \mathbb{G}_1^2 + 2 \cdot (2 + s) \cdot \mathbb{G}_2^2$ and the cost to verify it is $4 \cdot (s + 1) \cdot P^4$.

We can now put the above together and estimate the computational costs for the proofs obtained via the two methods discussed above in the CRS model.

Complexity of LEq in the CRS Model: We use comm_{CRS} , i.e. the one which commits into $\mathbb{B}_1 \times \mathbb{B}_2$, as the commitment scheme. We perform the same analysis as in the ROM case for each stage;

1. Negligible
2. $2 \cdot (|W| + 1) \cdot (\mathbb{G}_1^2 + \mathbb{G}_2^2)$.
3. Here we execute $|W|$ times a proof for **WFComm** plus $|W|$ operations which show that the commitments in the first stage are well formed. Thus the prover computes $4 \cdot |W| \cdot (\mathbb{G}_1^2 + \mathbb{G}_2^2)$.
4. As before computing $\text{comm}_{\text{CRS}}'$ is cheap. The prover then needs to execute $|G|$ versions of **WFComm** for $\text{comm}_{\text{CRS}}'$, but does not need to prove that the values of $\text{comm}_{\text{CRS}}'$ are well formed, since the verifier will already know this. This step therefore requires a cost of $2 \cdot |G| \cdot (\mathbb{G}_1^2 + \mathbb{G}_2^2)$.
5. Negligible.

The total cost for the prover is therefore

$$2 \cdot (3 \cdot |W| + |G| + 1) \cdot (\mathbb{G}_1^2 + \mathbb{G}_2^2),$$

and the cost for the verifier is

$$4 \cdot (2 \cdot |W| + |G|) \cdot P^4 + 2 \cdot |O| \cdot (\mathbb{G}_1^2 + \mathbb{G}_2^2).$$

Complexity of QEq in the CRS Model: In this method we take as our basic commitment scheme the commitment $\text{comm}_{\text{CRS}1}$, which represents commitments in \mathbb{B}_1 . Performing an analysis as above we find at each step the prover needs to compute

1. Negligible.
2. $2 \cdot |I| \cdot \mathbb{G}_1^2$.
3. To execute this proof the prover needs to first make the same commitments in \mathbb{B}_2 , prove they are equivalent and then execute the proof that the committed value is in $\{0, 1\}$. This in total requires $|I| \cdot (4 \cdot \mathbb{G}_1^2 + 6 \cdot \mathbb{G}_2^2)$.
4. $2 \cdot |\mathcal{M}on| \cdot \mathbb{G}_1^2$.
5. This is negligible cost as the values of a, b, c and d are very small.
6. Here we execute **WFProd** in this format which has a cost of $2 \cdot (t + |\mathcal{M}on|) \cdot \mathbb{G}_1^2 + 2 \cdot (2 \cdot t + |\mathcal{M}on|) \cdot \mathbb{G}_2^2$.
7. Negligible.

Thus the total cost for the prover is

$$2 \cdot (3 \cdot |I| + 2 \cdot |\mathcal{M}on| + t + 1) \cdot \mathbb{G}_1^2 + 2 \cdot (3 \cdot |I| + 2 \cdot t + |\mathcal{M}on|) \cdot \mathbb{G}_2^2,$$

and the cost for the verifier is

$$4 \cdot (t + |\mathcal{M}on| + 2 \cdot |I|) \cdot P^4 + 2 \cdot |O| \cdot \mathbb{G}_1^2.$$

5 Batch Verification

In order to speed up the times the different verifiers take, we can use batch techniques and in particular the small exponents test from [2] in the ROM model; In the CRS while a similar batch verification to the one used by Camenish et al. [8], for batch verifying BLS signature scheme, can be used to batch verify our CRS proofs.

5.1 Small Exponent Test

Choose $\gamma_1, \dots, \gamma_n$ at random where l is the bit length of γ_i . We then compute $x = \sum_{i=1}^n (x_i \cdot \gamma_i)$ and $y = \prod_{i=1}^n y_i^{\gamma_i}$. The verification is done by checking that $g^x = y$. We set the small exponent test's error rate at 2^{-l} by choosing exponents such that $|\gamma_i| = l$. To compute y , there are fast methods that can compute product of powers like the one in [7] which requires a total of $\frac{l(n+2)}{2}$ on average. The precise value depending of the hamming weight of the exponents.

5.2 Batch Verification in the ROM Model

Since **WFComm** and **WFProd** are the main building blocks in our proofs in the ROM model, we will introduce the batch verification in respect to them.

WFComm

As we have seen earlier, to verify a single **WFComm** proof, the verifier needs to check that the following two equations hold:

$$\begin{aligned} A_0 &= z_0 \cdot Q + e_0 \cdot A \\ A_1 &= z_1 \cdot Q - e_1 \cdot (A - P) \end{aligned}$$

So if we have n **WFComm** proofs, then we need to verify n pairs of the above equations. However, if we use the small exponent test to batch verify those n **WFComm** proofs, the verification reduces to checking the following equation only:

$$\begin{aligned} \sum_{i=1}^n (\gamma_{1,i} \cdot A_{0,i}) + \sum_{i=1}^n (\gamma_{2,i} \cdot A_{1,i}) &= \left(\sum_{i=1}^n (\gamma_{1,i} \cdot z_{0,i}) + \sum_{i=1}^n (\gamma_{2,i} \cdot z_{1,i}) \right) \cdot Q \\ &\quad + \sum_{i=1}^n ((e_{0,i} \cdot \gamma_{1,i} - e_{1,i} \cdot \gamma_{2,i}) \cdot A_i) \\ &\quad + \left(\sum_{i=1}^n (e_{1,i} \cdot \gamma_{2,i}) \right) \cdot P \end{aligned}$$

Again here we can use the product of powers algorithm to efficiently compute $\sum_{i=1}^n ((e_{0,i} \cdot \gamma_{1,i} - e_{1,i} \cdot \gamma_{2,i}) \cdot A_i)$ and $(\sum_{i=1}^n (\gamma_{1,i} \cdot A_{0,i}) + \sum_{i=1}^n (\gamma_{2,i} \cdot A_{1,i}))$. Thus batch verification of n **WFComm** would in total require $\frac{l(2n+2)+q(n+2)}{2}$ point additions and 2 scalar multiplications (or one double multiplication), where q is the bit length of the group order.

WFProd

We note that the optimization we used in 3.2 in which we generate a proof for all the monomials in each subset Mon_i in one go would not really help taking advantage of the batch verification as n (i.e. the size of the batch) in this case is very small and batch verification in general is meant to save time as n grows. This is because (in terms of the underlying Σ -protocol) we send the challenge value, and get the verifier to recompute the individual commitments, then a proof is verified by checking the challenge is obtained from hashing the commitments. This saves bandwidth, but stops one being able to perform batch verification techniques efficiently. If we instead send the commitments, the verifier then computes the challenge (via the hash), and verifies the equations, as one would for an interactive Σ -protocol, one can apply batch verification techniques. However, this not only increases the size of the proof significantly, we obtain no benefit in treating each subset Mon_i in a separate manner.

We therefore ran another experiment in which we did not divide the set Mon into t disjoint subsets and instead generated a proof for each element in the big set Mon individually, using the above technique for transmitting the proof. As we will see from the results in Section 6 it is actually a matter of a trade off between the speed of the prover and the verifier and choosing the appropriate method accordingly.

Since we adapted the way the quadratic equations method are evaluated to verify a single product proof, the verifier needs to check the following two equations for each element in the set \mathcal{M}_{on} :

$$\begin{aligned} A_0 &= s_0 \cdot P + s_1 \cdot Q - e \cdot A \\ A_1 &= s_0 \cdot B + s_2 \cdot Q - e \cdot C \end{aligned}$$

Applying the batch verification to verify n **WFProd** proofs then requires checking the following equation only:

$$\begin{aligned} \sum_{i=1}^n (\gamma_{1,i} \cdot A_{0,i}) + \sum_{i=1}^n (\gamma_{2,i} \cdot A_{1,i}) &= \left(\sum_{i=1}^n (\gamma_{1,i} \cdot s_{0,i}) \right) \cdot P \\ &\quad + \left(\sum_{i=1}^n (\gamma_{1,i} \cdot s_{1,i}) + \sum_{i=1}^n (\gamma_{2,i} \cdot s_{2,i}) \right) \cdot Q \\ &\quad - \sum_{i=1}^n ((\gamma_{1,i} \cdot e_i) \cdot A_i) + \sum_{i=1}^n ((\gamma_{2,i} \cdot s_{0,i}) \cdot B_i) \\ &\quad - \sum_{i=1}^n ((\gamma_{2,i} \cdot e_i) \cdot C_i) \end{aligned}$$

Thus batch verification of n **WFProd** would in total require $\frac{l(2n+2)+q(3n+2)}{2}$ point additions and 2 scalar multiplications (or one double multiplication), where q is the bit length of the group order.

5.3 Batch Verification in the CRS Model

As we see from the experiments' results in Section 6, the times of the verifiers of the two methods in the CRS model are slower than their counterparts in the ROM model and that comes as no surprise because pairing, which is the main operation in the CRS model, is an expensive one and hence using batch verification to reduce the number of pairings required would greatly reduce the time needed. We will explain our batch verification in respect to the three main proofs we use in the CRS model which are:

Product

As we see in A.1, to verify a single *Product* proof, the verifier needs to check :

$$F\left(\underline{C_1}^{(2)}, -\mathcal{W}_2\right) \cdot F\left(\underline{C_1}^{(1)}, \underline{C_2}^{(1)}\right) = F(\mathcal{U}_1, \Pi) \cdot F(\Theta, \mathcal{U}_2)$$

Batch verifying n *Product* proofs would only require checking the following equation:

$$\begin{aligned} F\left(\sum_{i=1}^n \left(\gamma_i \cdot \underline{C_{1,i}}^{(2)}\right), -\mathcal{W}_2\right) \cdot \prod_{i=1}^n F\left(\gamma_i \cdot \underline{C_{1,i}}^{(1)}, \underline{C_{2,i}}^{(1)}\right) \\ = F(\mathcal{U}_1, \sum_{i=1}^n (\gamma_i \cdot \Pi_i)) \cdot F(\sum_{i=1}^n (\gamma_i \cdot \Theta_i), \mathcal{U}_2) \end{aligned}$$

Thus batch verifying n *Product* proofs would require only $n + 3$ products of four lots of standard pairings compared to $4n$ products of four lots of standard pairings if the proofs are verified individually.

BitProof

As we see in A.2, to verify a single *BitProof*, the verifier needs to check :

$$F\left(\underline{C_1}^{(1)}, -\mathcal{W}_2\right) \cdot F\left(\underline{C_1}^{(1)}, \underline{C_2}^{(1)}\right) = F(\mathcal{U}_1, \Pi) \cdot F(\Theta, \mathcal{U}_2)$$

Batch verifying n *BitProof* proofs would only require checking the following equation:

$$\begin{aligned} F\left(\sum_{i=1}^n \left(\gamma_i \cdot \underline{C_{1,i}}^{(1)}\right), -\mathcal{W}_2\right) \cdot \prod_{i=1}^n F\left(\gamma_i \cdot \underline{C_{1,i}}^{(1)}, \underline{C_{2,i}}^{(1)}\right) \\ = F(\mathcal{U}_1, \sum_{i=1}^n (\gamma_i \cdot \Pi_i)) \cdot F\left(\sum_{i=1}^n (\gamma_i \cdot \Theta_i), \mathcal{U}_2\right) \end{aligned}$$

Again, batch verifying n *BisProofs* would only require $n+3$ products of four lots of standard pairings compared to $4n$ products of four lots of standard pairings when the proofs are verified individually.

Equality

As we see in A.3, to verify a single *Equality* proof, the verifier needs to check :

$$F\left(\underline{C_1}^{(1)}, -\mathcal{W}_2\right) \cdot F\left(\mathcal{W}_1, \underline{C_2}^{(1)}\right) = F(\mathcal{U}_1, \Pi) \cdot F(\Theta, \mathcal{U}_2)$$

Batch verifying n *Equality* proofs would only require checking the following equation:

$$\begin{aligned} F\left(\sum_{i=1}^n \left(\gamma_i \cdot \underline{C_{1,i}}^{(1)}\right), -\mathcal{W}_2\right) \cdot F\left(\mathcal{W}_1, \sum_{i=1}^n \left(\gamma_i \cdot \underline{C_{2,i}}^{(1)}\right)\right) \\ = F(\mathcal{U}_1, \sum_{i=1}^n (\gamma_i \cdot \Pi_i)) \cdot F\left(\sum_{i=1}^n (\gamma_i \cdot \Theta_i), \mathcal{U}_2\right) \end{aligned}$$

Batch verifying n *Equality* proofs would therefore require only 4 products of four lots of standard pairings compared to $4n$ products of four lots of standard pairings if the proofs are verified individually.

As we will see, using batch verification in the CRS model greatly reduces the number of pairings to be executed which in turn considerably minimizes the time needed for verifying the proofs

6 Experimental Results

In this section, we present the results and the timings we have achieved for our different implementations. We first describe the two types of groups that we used, we then present the two circuits, and finally we present our timings.

In the random oracle model, we based our commitment scheme on the elliptic curve *secp256r1* from the SECG standard [27]. This is a 256-bit elliptic curve defined over the prime finite field with

$$p = 2^{224} \cdot (2^{32} - 1) + 2^{192} + 2^{96} - 1$$

elements.

In the CRS model, we used a 256-bit Barreto–Naehrig curve [1] defined over the finite field of

$$p = 36 \cdot z^4 + 36 \cdot z^3 + 24 \cdot z^2 + 6 \cdot z + 1$$

elements, where $z = 0X68000000000000112$. The curve is given by $E : Y^2 = X^3 + 18$. Such a curve allows one to use efficient implementation tricks such as sextic twists and the Ate-pairing etc. [22]. In addition having a sparse value of z (in binary form) allows for more efficient implementation of the Ate-pairing.

Our batch verification techniques were implemented using the small exponent method with small-exponents of 128-bits in length.

For the two circuits we first used a small circuit in which the prover commits to two 32-bit integers, and then proves that one is smaller than the other. In the second circuit, the prover commits to a 128-bit cryptographic key, and then proves that a public 128-bit message and 128-bit ciphertext block arise from applying the AES cipher to the message under the hidden key (similar timings result from proving a hidden key *and* hidden message produce a given ciphertext). In Table 1 we present the details of the two circuits. The last two lines in the table refer to the parameters related to our optimization in **QEeq** described earlier. Namely, the value t related to the number of parallel multiplication proofs which need to be performed, whilst $|\mathcal{M}on|$ denotes the number of products which need to be proved to be correct.

Table 1. Details of the two circuits used in the experiments

Parameter	Circuit-1	Circuit-2
No. Gates	184	33880
No. Input Wires	64	128
No. Output Wires	1	128
Total No. Wires	248	34136
t	93	15596
$ Mon $	154	32244

We can now present our timings. In Figures 1 and 2 we summarize the execution time for producing and verifying proofs for the two circuits that we consider. In each case we specify the model in which the proof is designed (ROM or CRS) and we specify which of the two methods is used (**LEq** or **QE_q**). We present timings for when each sub-proof is verified individually and when the sub-proofs are evaluated in a batch manner. All our timings are in seconds and were tested on a Linux machine with Intel Core Duo 3.00GHz processor.

Model	LEq method	QEeq method
ROM	4.7	2.25 1.95
CRS	44	15.23

Model	LEq method		QEeq method	
	Individual	Batch	Individual	Batch
ROM	5.3	1.97	2.5	1.28 2.01
CRS	450	64	163	29.5

Fig. 1. Timing (in seconds) for the running time of the prover (left) and the verifier (right) for Circuit 1

Model	LEq method	QEeq method
ROM	729	380 296
CRS	7174	2406

Model	LEq method		QEeq method	
	Individual	Batch	Individual	Batch
ROM	839	321	372	253 360
CRS	70300	9431	24861	4200

Fig. 2. Timing (in seconds) for the running time of the prover (left) and the verifier (right) for Circuit 2

In the **QEeq** and ROM model case (i.e. the last column of the first row in each table) we provided two times which represent the time when our optimization by dividing the set $\mathcal{M}on$ into t disjoint subsets is not used and the timings when it is used respectively. We needed this in order to compare both approaches when using Batch verification.

We see, as is to be expected, that the timings for the Groth et al proofs in the CRS are slower than the equivalent proofs using Σ -protocols turned into NIZK's using the Blum transform. But we notice that constructing the Groth–Sahai proofs in the CRS model is only 7-9 times less efficient than the equivalent proofs derived from the Blum transform being applied to a Σ -protocol in the random oracle model. However, the relative performance of the verifier is much worse, being around 10-20 times less efficient for the CRS based method compared to the ROM based method for the **QEeq** method.

We also see that **QEeq** for performing the proof is always significantly more efficient. Of particular interest is that proving, in the random oracle model, a significantly large circuit, such as knowledge of an AES encryption key, is a relatively simple task.

As for batch verification, we see that batch verification provides better time saving in the CRS model than it does in the ROM model as it made verifying **LEq** about 7 times faster than individual verification and made verifying **QEeq** about 6 times faster compared to about 2.5 times faster for **LEq** and twice as fast for **QEeq** in the ROM model.

As a by-product of our experiments using the above circuits we obtain the first timings of the Groth et al proof techniques. In Table 2 we present these for the three example proofs given in the Appendix. It should come as no surprise that the timings are all remarkably similar, since the equations we are using are all very similar. Increasing the complexity of the quadratic equations will lead to an increase in the number of terms in the four pairing products which need to be evaluated by the verifier, which will impact significantly on the overall run times. However, we hope that these initial timings for Groth–Sahai proofs

Table 2. Timings for the different examples of Groth–Sahai[21]

Proof Name	Prover Time	Verifier Time
Product	0.035	0.52
BitProof	0.035	0.52
Equality	0.04	0.52

in the asymmetric pairing setting will be useful for other researchers wishing to use them in practical schemes. We note (for completeness) for such researchers, that the time needed to compute a commitment in the Groth–Sahai setting we found to be 0.005, 0.025, 0.030 seconds respectively, for commitments in \mathbb{B}_1 , \mathbb{B}_2 and $\mathbb{B}_1 \times \mathbb{B}_2$.

Acknowledgements. The authors would like to thank J. Groth for helpful comments on an earlier version of this paper, in particular they thank him for suggesting using batch verification techniques. All authors were partially supported by the FP7 projects CACE and eCrypt-2, whilst the second author was also supported by a Royal Society Wolfson Merit Award.

References

1. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
2. Bellare, M., Garay, J., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
3. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Conference on Computer and Communication Security – CCS 1993, pp. 62–73. ACM, New York (1993)
4. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: Symposium on Theoretical Computer Science – STOC 1988, pp. 103–112. ACM, New York (1988)
5. 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)
6. Boyar, J., Damgård, I., Peralta, R.: Short non-interactive cryptographic proofs. *Journal of Cryptology* 13, 449–472 (2000)
7. Brickell, E., Gordon, D., McCurley, K., Wilson, D.: Fast exponentiation with pre-computation. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 200–207. Springer, Heidelberg (1993)
8. Camenisch, J., Hohenberger, S., Pedersen, M.: Batch verification of short signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007)
9. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)

10. Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998)
11. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
12. Damgård, I.: Non-interactive circuit based proofs and non-interactive proofs of knowledge with preprocessing. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 341–355. Springer, Heidelberg (1993)
13. De Santis, A., Di Crescenzo, G., Persiano, G.: Non-interactive zero-knowledge: A low-randomness characterization of NP. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 271–280. Springer, Heidelberg (1999)
14. De Santis, A., Di Crescenzo, G., Persiano, G.: Randomness-optimal characterization of two NP proof systems. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 179–193. Springer, Heidelberg (2002)
15. Feige, U., Lapidot, D., Shamir, A.: Non-interactive zero-knowledge proofs based on a single random string. In: Symposium of Foundations of Computer Science – FOCS 1990, pp. 308–317 (1990)
16. Galbraith, S., Paterson, K., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156, 3113–3121 (2008)
17. Granger, R., Smart, N.P.: On computing products of pairings. Cryptology ePrint Archive, Report 2006/172 (2006), <http://eprint.iacr.org/2006/172/>
18. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive Zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
19. Groth, J., Ostrovsky, R., Sahai, A.: Perfect Non-interactive zero-knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
20. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for non-interactive zero-knowledge. Full version of [18] and [19], <http://www.brics.dk/~jg/NIZKJournal3.ps>
21. 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)
22. Hess, F., Smart, N.P., Vercauteren, F.: The Eta pairing revisited. IEEE Transactions on Information Theory 52, 4595–4602 (2006)
23. Kilian, J., Petrank, E.: An efficient non-interactive proof system for NP with general assumptions. Journal of Cryptology 11, 1–27 (1998)
24. Nguyen, K.Q., Bao, F., Mu, Y., Varadharajan, V.: Zero-knowledge proofs of possession of digital signatures and its applications. In: Varadharajan, V., Mu, Y. (eds.) ICICS 1999. LNCS, vol. 1726, pp. 103–118. Springer, Heidelberg (1999)
25. Okamoto, T.: Provable secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
26. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4, 161–174 (1991)

27. SECG. Standards for Efficient Cryptography, SEC 2: Recommended elliptic curve domain parameters, <http://www.secg.org>
28. Szydlo, M.: Risk assurance for hedge funds using zero knowledge proofs. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 156–171. Springer, Heidelberg (2005)

A Example Pairing Based Proofs

We now present three examples. These have been chosen as they are ones which we will need quite a lot of in our comparisons. Bellare at all in their paper suggested a way to com

A.1 Product

Consider the equation $\underline{x}_1^{(1)} \cdot \underline{x}_2^{(1)} - \underline{x}_1^{(2)} = 0$. i.e. we want to show that one committed value is product of two other committed values. We have

$$n_1 = 2, \quad n_2 = 1, \quad t = 0, \quad \underline{a}_1 = (0, -1), \quad \underline{a}_2 = (0), \quad \Gamma = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The commitments are given by $\underline{C}_1^{(1)}$, $\underline{C}_1^{(2)}$ and $\underline{C}_2^{(1)}$ and the proof is given by

$$\begin{aligned} \Theta &= \underline{r}_2 \cdot \iota'_1(\underline{a}_2)^t + \underline{r}_2 \cdot \Gamma^t \cdot \iota'_1(\underline{x}_1)^t + s \cdot \mathcal{U}_1 \\ &= \underline{r}_2^{(1)} \cdot (1, 0) \cdot (\underline{x}_1^{(1)} \cdot \mathcal{W}_1, \underline{x}_1^{(2)} \cdot \mathcal{W}_1)^t + s \cdot \mathcal{U}_1 \\ &= (\underline{r}_2^{(1)} \cdot \underline{x}_1^{(1)}) \cdot \mathcal{W}_1 + s \cdot \mathcal{U}_1, \end{aligned}$$

$$\begin{aligned} \Pi &= \underline{r}_1 \cdot \iota'_2(\underline{a}_1)^t + \underline{r}_1 \cdot \Gamma \cdot \iota'_2(\underline{x}_2)^t + (\underline{r}_1 \cdot \Gamma \cdot \underline{r}_2^t - s) \cdot \mathcal{U}_2 \\ &= (\underline{r}_1^{(1)}, \underline{r}_1^{(2)}) \cdot \iota'_2((0, -1))^t + (\underline{r}_1^{(1)}, \underline{r}_1^{(2)}) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot \iota'_2(\underline{x}_2^{(1)}) \\ &\quad + ((\underline{r}_1^{(1)}, \underline{r}_1^{(2)}) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot \underline{r}_2^{(1)} - s) \cdot \mathcal{U}_2 \\ &= (\underline{r}_1^{(1)}, \underline{r}_1^{(2)}) \cdot (0, -\mathcal{W}_2)^t + \underline{r}_1^{(1)} \cdot \underline{x}_2^{(1)} \cdot \mathcal{W}_2 + (\underline{r}_1^{(1)} \cdot \underline{r}_2^{(1)} - s) \cdot \mathcal{U}_2 \\ &= (\underline{r}_1^{(1)} \cdot \underline{x}_2^{(1)} - \underline{r}_1^{(2)}) \cdot \mathcal{W}_2 + (\underline{r}_1^{(1)} \cdot \underline{r}_2^{(1)} - s) \cdot \mathcal{U}_2. \end{aligned}$$

Creating the proof therefore requires one double multiplication in \mathbb{B}_1 and one double multiplication in \mathbb{B}_2 . This reduces to only a single multiplication in \mathbb{B}_1 and one double multiplication in \mathbb{B}_2 when $\underline{x}_1^{(1)} = 0$. To verify the proof we simply check that

$$F(\underline{C}_1^{(2)}, -\mathcal{W}_2) \cdot F(\underline{C}_1^{(1)}, \underline{C}_2^{(1)}) = F(\mathcal{U}_1, \Pi) \cdot F(\Theta, \mathcal{U}_2)$$

To verify the proof we simply need to compute a product of four evaluations of F . Thus to verify the proof we need to execute four products of four standard pairings.

A.2 BitProof

Now consider the equation $\underline{x_1}^{(1)} \cdot \underline{x_2}^{(1)} - \underline{x_1}^{(1)} = 0$. We have

$$n_1 = 1, \quad n_2 = 1, \quad t = 0, \quad \underline{a_1} = (-1), \quad \underline{a_2} = (0), \quad \Gamma = (1).$$

The commitments are given by $\underline{C_1}^{(1)}$ and $\underline{C_2}^{(1)}$ and the proof is given by

$$\begin{aligned} \Theta &= \underline{r_2} \cdot \iota'_1(\underline{a_2})^t + \underline{r_2} \cdot \Gamma^t \cdot \iota'_1(\underline{x_1})^t + s \cdot \mathcal{U}_1 \\ &= (\underline{r_2}^{(1)} \cdot \underline{x_1}^{(1)}) \cdot \mathcal{W}_1 + s \cdot \mathcal{U}_1, \end{aligned}$$

$$\begin{aligned} \Pi &= \underline{r_1} \cdot \iota'_2(\underline{a_1})^t + \underline{r_1} \cdot \Gamma \cdot \iota'_2(\underline{x_2})^t + (\underline{r_1} \cdot \Gamma \cdot \underline{r_2}^t - s) \cdot \mathcal{U}_2 \\ &= \underline{r_1}^{(1)} \cdot \iota'_2(-1)^t + \underline{r_1}^{(1)} \cdot (1) \cdot \iota'_2(\underline{x_2}^{(1)}) + (\underline{r_1}^{(1)} \cdot (1) \cdot \underline{r_2}^{(1)} - s) \cdot \mathcal{U}_2 \\ &= -\underline{r_1}^{(1)} \cdot \mathcal{W}_2 + \underline{r_1}^{(1)} \cdot \underline{x_2}^{(1)} \cdot \mathcal{W}_2 + (\underline{r_1}^{(1)} \cdot \underline{r_2}^{(1)} - s) \cdot \mathcal{U}_2 \\ &= (\underline{r_1}^{(1)} \cdot \underline{x_2}^{(1)} - \underline{r_1}^{(1)}) \cdot \mathcal{W}_2 + (\underline{r_1}^{(1)} \cdot \underline{r_2}^{(1)} - s) \cdot \mathcal{U}_2. \end{aligned}$$

Creating the proof therefore requires one double multiplication in \mathbb{B}_1 and one double multiplication in \mathbb{B}_2 . This reduces to only a single multiplication in \mathbb{B}_1 and one double multiplication in \mathbb{B}_2 when $\underline{x_1}^{(1)} = 0$. To verify the proof we simply check that:

$$F(\underline{C_1}^{(1)}, -\mathcal{W}_2) \cdot F(\underline{C_1}^{(1)}, \underline{C_2}^{(1)}) = F(\mathcal{U}_1, \Pi) \cdot F(\Theta, \mathcal{U}_2).$$

Again to verify the proof we require four products of four standard pairings.

A.3 Equality

As our final example, consider the equation $\underline{x_2}^{(1)} - \underline{x_1}^{(1)} = 0$. i.e. we want to show that one two committed values are equal where the commitments lie in different groups. We have

$$n_1 = 1, \quad n_2 = 1, \quad t = 0, \quad \underline{a_1} = (-1), \quad \underline{a_2} = (1), \quad \Gamma = (0).$$

The commitments are given by $\underline{C_1}^{(1)}$ and $\underline{C_2}^{(1)}$ and the proof is given by

$$\begin{aligned} \Theta &= \underline{r_2} \cdot \iota'_1(\underline{a_2})^t + \underline{r_2} \cdot \Gamma^t \cdot \iota'_1(\underline{x_1})^t + s \cdot \mathcal{U}_1 \\ &= \underline{r_2}^{(1)} \cdot \mathcal{W}_1 + s \cdot \mathcal{U}_1, \end{aligned}$$

$$\begin{aligned} \Pi &= \underline{r_1} \cdot \iota'_2(\underline{a_1})^t + \underline{r_1} \cdot \Gamma \cdot \iota'_2(\underline{x_2})^t + (\underline{r_1} \cdot \Gamma \cdot \underline{r_2}^t - s) \cdot \mathcal{U}_2 \\ &= -\underline{r_1}^{(1)} \cdot \mathcal{W}_2 - s \cdot \mathcal{U}_2. \end{aligned}$$

Creating the proof therefore requires one double multiplication in \mathbb{B}_1 and one double multiplication in \mathbb{B}_2 . To verify the proof we simply check that

$$F(\underline{C_1}^{(1)}, -\mathcal{W}_2) \cdot F(\mathcal{W}_1, \underline{C_2}^{(1)}) = F(\mathcal{U}_1, \Pi) \cdot F(\Theta, \mathcal{U}_2).$$

Which is again four products of four standard pairings.

Author Index

- Attrapadung, Nuttapong 278
Betsumiya, Koichi 65
Betty, Rowena Alma L. 65
Borghoff, Julia 133
Boucher, Delphine 38
Cathalo, Julien 326
Chatterjee, Sanjit 236
Chen, Hao 263
Chen, Liqun 301
Dallot, Léonard 222
Danielsen, Lars Eirik 418
Davenport, James H. 301
Edel, Yves 383
Fanali, Stefania 91
Fleischmann, Ewan 153
Galbraith, Steven 368
Geisler, Martin 252
Gérard, Benoît 112
Ghadafi, Essam 469
Gorski, Michael 153
Harada, Masaaki 78
Hawkes, Philip 433
Imai, Hideki 278
Joux, Antoine 351
Karabina, Koray 236, 336
Khader, Dalia 301
Khaleghi, Azadeh 1
Knudsen, Lars R. 133
Kschischang, Frank R. 1
Leander, Gregor 433
Lercier, Reynald 351
Ling, San 263
Lucks, Stefan 153
Menezes, Alfred 236
Mesnager, Sihem 402
Munemasa, Akihiro 65, 78
Murphy, Sean 202
Naccache, David 326, 351
Özen, Onur 176
Padró, Carles 263
Parker, Matthew G. 418
Paterson, Maura B. 202
Pott, Alexander 383
Quisquater, Jean-Jacques 326
Rosnes, Eirik 22
Ruprai, Raminder S. 368
Schaathun, Hans Georg 56
Schindler, Werner 446
Silva, Danilo 1
Smart, Nigel P. 252, 469
Solé, Patrick 418
Stam, Martijn 176
Stolpe, Mathias 133
Thomé, Emmanuel 351
Tillich, Jean-Pierre 112
Ulmer, Felix 38
Vergnaud, Damien 222
Walter, Colin D. 446
Wang, Huaxiong 263
Warinschi, Bogdan 469
Xing, Chaoping 263
Zenner, Erik 433