Michel Abdalla
Tanja Lange (Eds.)

Pairing-Based Cryptography – Pairing 2012

LNCS 7708

5th International Conference
Cologne, Germany, May 2012
Revised Selected Papers

Springer

Lecture Notes in Computer Science 7708

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

Michel Abdalla Tanja Lange (Eds.)

Pairing-Based Cryptography – Pairing 2012

5th International Conference
Cologne, Germany, May 16-18, 2012
Revised Selected Papers

Springer

Volume Editors

Michel Abdalla
École Normale Supérieure and CNRS
Départment d'Informatique
45 rue d'Ulm, 75005 Paris, France
E-mail: michel.abdalla@ens.fr

Tanja Lange
Technische Universiteit Eindhoven
Department of Mathematics and Computer Science
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
E-mail: tanja@hyperelliptic.org

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-36333-7 e-ISBN 978-3-642-36334-4
DOI 10.1007/978-3-642-36334-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012956168

CR Subject Classification (1998): E.3, K.6.5, D.4.6, E.4, F.2.0, I.1

LNCS Sublibrary: SL 4 – Security and Cryptology

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

Printed on acid-free paper

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

Preface

Pairing 2012 was the 5th International Conference on Pairing-Based Cryptography and took place during May 16–18, 2012, in Cologne, Germany. The conference was organized by the Coding Theory and Cryptology group at the Eindhoven Institute for the Protection of Systems and Information, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, with the aim of bringing together leading researchers and practitioners from academia and industry, all concerned with problems related to pairing-based cryptography. The General Chairs of the conference were Tanja Lange and Michael Naehrig, and the secretarial support was provided by Anita Klooster from the Technische Universiteit Eindhoven. We thank both Michael and Anita for their constant efforts and for making this conference possible.

The conference received 49 submissions and each submission was assigned to at least three committee members. Submissions co-authored by members of the Program Committee were assigned to at least four committee members. We were happy to receive a good number of high-quality submissions, and we are grateful to the committee members and external reviewers for their outstanding work in thoroughly reviewing all papers in a timely manner. After a discussion phase of 19 days, leading to 290 comments on the submissions, the Program Committee, selected 17 submissions for presentation in the academic track. Additionally, three other submissions were selected for presentation in the industrial track. The final versions of these submissions were not checked by the Program Committee and the authors bear full responsibility for their contents.

The program included four invited talks in addition to the academic and industrial tracks. These talks were given by Jean-Luc Beuchat, Jung-Hee Cheon, Dennis Hofheinz, and Hovav Shacham, and covered a wide range of topics in pairing-based cryptography. In addition, the program included shorter invited talks by Benoît Libert and Katsuyuki Takashima in a session about hot topics in pairings, following a trend established in previous editions of this conference. The abstracts of these invited talks were also included in this volume.

The reviewing process was run using the iChair software, written by Thomas Baignères from CryptoExperts, France and Matthieu Finiasz from EPFL, LASEC, Switzerland. We are grateful to them for letting us use their software.

Finally, we would like to thank our sponsors Netherlands Organization for Scientific Research (NWO), Microsoft Research, and Voltage Security for their financial support as well as all the Pairing Steering Committee for selecting us as Program Chairs. We would also like to thank Springer for accepting to publish the proceedings in the *Lecture Notes in Computer Science* series.

December 2012

Michel Abdalla
Tanja Lange

Organization

General Chairs

Tanja Lange	Technische Universiteit Eindhoven, The Netherlands
Michael Naehrig	Technische Universiteit Eindhoven, The Netherlands

Program Chairs

Michel Abdalla	École Normale Supérieure and CNRS, France
Tanja Lange	Technische Universiteit Eindhoven, The Netherlands

Program Committee

Michel Abdalla (Co-chair)	École Normale Supérieure and CNRS, France
Paulo S.L.M. Barreto	University of São Paulo, Brazil
Naomi Benger	University of Adelaide, Australia
Melissa Chase	Microsoft Research, USA
Jérémie Detrey	INRIA, France
Junfeng Fan	K.U. Leuven, Belgium
Dario Fiore	New York University, USA
David Mandell Freeman	Stanford University, USA
Steven Galbraith	University of Auckland, New Zealand
Juan González Nieto	Queensland University of Technology, Australia
Shai Halevi	IBM Research, USA
Antoine Joux	Université de Versailles and DGA, France
Kwangjo Kim	KAIST, Korea
Tanja Lange (Co-chair)	Technische Universiteit Eindhoven, The Netherlands
Kristin Lauter	Microsoft Research, USA
Allison B. Lewko	University of Texas at Austin, USA
Benoît Libert	Technicolor, France
Atsuko Miyaji	JAIST, Japan
Michael Naehrig	Technische Universiteit Eindhoven, The Netherlands
Takeshi Okamoto	University of Tsukuba, Japan
Adam O'Neill	Boston University, USA
Giuseppe Persiano	Università di Salerno, Italy

Christophe Ritzenthaler	Institut de Mathématiques de Luminy, France
Francisco Rodríguez-Henríquez	CINESTAV-IPN, Mexico
Peter Schwabe	Academia Sinica, Taiwan
Michael Scott	Certivox Ltd.
Tsuyoshi Takagi	Kyushu University, Japan
Katsuyuki Takashima	Mitsubishi Electric, Japan
Edlyn Teske-Wilson	University of Waterloo, Canada
Damien Vergnaud	École Normale Supérieure, France
Jianying Zhou	Institute for Infocomm Research (I2R), Singapore

Referees

Dario Catalano	Joseph Liu
Kai-Yuen Cheong	Lourdes López-García
Cheng-Kang Chu	Pedro Maat Costa Massolino
Craig Costello	Toru Nakanishi
Angelo De Caro	Kazuto Ogawa
Robert Drylo	Thomas Peters
Georg Fuchsbauer	Somindu C Ramanna
Matthew Green	Jothi Rangasamy
Nicolas Guillermin	Dominique Schröder
Mitsuhiro Hattori	Jae Hong Seo
Fumitaka Hoshino	Masaaki Shirase
Vincenzo Iovino	Shashank Singh
Hamza Jeljeli	Benjamin Smith
Mitsuru Kawazoe	Jia Xu
Lakshmi Kuppusamy	Takanori Yasuda
Fagen Li	Mingwu Zhang

Sponsoring Institutions

Netherlands Organization for Scientific Research (NWO)
Microsoft Research
Voltage Security

Hardware Architectures for the Cryptographic Tate Pairing (Invited Talk)

Jean-Luc Beuchat

Faculty of Engineering, Information and Systems,
University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
jeanluc.beuchat@gmail.com

Abstract. In the mid-nineties of the last century, Menezes, Okamoto & Vanstone and Frey & Rück introduced the Weil and Tate pairings in cryptography as a tool to attack the discrete logarithm problem on some classes of elliptic curves defined over finite fields. The discovery of constructive properties by Joux, Mitsunari, Sakai & Kasahara, and Sakai, Oghishi & Kasahara initiated the proposal of an ever-increasing number of protocols based on bilinear pairings: identity-based encryption, short signature, and efficient broadcast encryption, to mention but a few. However, such protocols rely critically on efficient implementations of pairing primitives at high security levels on a wide range of targets.

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

We sketch here several hardware architectures for the Tate pairing on supersingular and ordinary curves. First, we emphasize on reducing the silicon footprint of the circuit to ensure scalability, while trying to minimize the impact on the overall performances. Then, we focus on the other end of the hardware design spectrum and explain how to achieve much lower computation times, at the expense of extra hardware resources. The main lesson learned from this study is that an appropriate mix of theoretical foundations and practical considerations is essential to design cryptographic hardware: fine-tuning of the algorithms, arithmetic operand encoding, scheduling, etc.

Discrete Logarithm in Pairing Groups
(Invited Talk)

Jung Hee Cheon

ISaC & Department of Mathematical Sciences, Seoul National University
jhcheon@snu.ac.kr

Abstract. In recent years, bilinear pairings have found various applications in cryptography to construct new cryptographic primitives. Pairing-based cryptography raises lots of new computational problems, but they have not been studied very well in the literature. In this talk, we survey recent progress in this field and then would like to address some open questions on the discrete logarithm problems in pairing groups. For this purpose, this talk is roughly comprised of three parts. The first part mainly focuses on the Pollard rho algorithm on pairing groups. We introduce the *Tag Tracing* technique to speed up Pollard rho algorithm and investigate how to apply this technique to elliptic curves with bilinear maps. The second topic is on the pairing inversion problem. We discuss about polynomial representations of this problem and show how to reduce the degree of the corresponding polynomial. The last topic is related to the strong DH assumption, which is one of the most popular cryptographic assumptions in the field of pairing-based cryptography. We take a look around the security of the strong DH assumption and its following variants with auxiliary inputs. It was proved that they have less security than the square-root of p when either $p-1$ or $p+1$ has an appropriate divisor of the base group order p. We introduce an attempt to generalize this attack by using an embedding to an extension field or elliptic curves, or by exploiting a polynomial with small image size.

Keywords: Discrete Logarithm, Pollard rho, Tag Tracing, Bilinear Maps, Pairing Inversion, Auxiliary Inputs.

Structure-Preserving Cryptography
(Invited Talk)

Dennis Hofheinz

Karlsruhe Institute of Technology
Institut für Kryptographie und Sicherheit
Building 50.34, room 279, Am Fasanengarten 5, 76131 Karlsruhe, Germany
dennis.hofheinz@kit.edu

Abstract. A cryptographic scheme is called structure-preserving, if the performed operations are solely abstract group operations. (In particular, this disallows the explicit use of, say, the bit representation of group elements.) Structure-preserving schemes are interesting because they are compatible with non-interactive proof systems for equations over groups. For instance, efficient Groth-Sahai proofs can be used to prove knowledge of a signature (of a structure-preserving signature scheme). This allows to transport generic paradigms (such as the Naor-Yung paradigm to achieve chosen-ciphertext encryption security) to an efficient group-based setting. This talk first gives an overview over structure-preserving schemes, and then presents a new result that uses a structure-preserving signature scheme as an essential building block. Concretely, we show how to construct a chosen-ciphertext secure public-key encryption scheme with a tight security reduction in the multi-user, multi-challenge setting.

Alternative Structure for Bilinear Groups
(Invited Talk)

Hovav Shacham

Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404
hovav@cs.ucsd.edu

Abstract. Pairing-based cryptography is a striking illustration of the value of algebraic structure for constructing crypto schemes: A richer structure allows for a wider variety of crypto schemes. It is perhaps surprising, then, that the way in which pairings are used have become quite standard. Most often, we imagine a bilinear group G to be a cyclic group of prime order that induces a map $e\colon G \times G \to G_T$ (where G_T is treated in a similarly abstract manner).

In this talk, I survey two lines of work that seek to generalize this understanding of pairings. One line considers bilinear groups G of composite order; the other line reconsiders the mathematical structure of the group G, for example to support asymmetric pairings $e\colon G_1 \times G_2 \to G_T$. Both these lines of work have been exploited to construct new cryptographic schemes.

In addition, I consider one of the instantiations of pairing-friendly elliptic curves proposed in a recent paper of Boneh, Rubin, and Silverberg. I show that this instantiation exhibits surprising and unprecedented new structure: projecting a point from the group G onto a subgroup G_1 or G_2 requires knowledge of a trapdoor. I propose new hardness assumptions for this setting and protocols that rely on them.

This is joint work with Sarah Meiklejohn.

Revocable Group Signatures from the NNL Subset Cover Framework

(Invited Session: Hot Topics in Pairings)

Benoît Libert

Technicolor
975 Avenue des Champs Blancs
35510 Cesson-Sévigné, France
benoit.libert@technicolor.com

Abstract. Group signatures are a central cryptographic primitive where users can anonymously sign messages in the name of a group they belong to. Despite years of research, membership revocation remains a non-trivial problem. Existing solutions either suffer from important overheads or require unrevoked users to update their keys after each revocation. We describe a new scalable revocation method, based on the Naor-Naor-Lotspiech (NNL) broadcast encryption framework, that interacts nicely with techniques for building group signatures in the standard model. We eventually obtain a scheme which is truly competitive with group signatures without revocation. Moreover, unrevoked members do not need to update their keys at each revocation.

Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption
(Invited Session: Hot Topics in Pairings)

Katsuyuki Takashima

Mitsubishi Electric, Japan
Takashima.Katsuyuki@aj.MitsubishiElectric.co.jp

Abstract. We present the first inner product encryption (IPE) scheme that is adaptively secure and fully attribute-hiding (attribute-hiding in the sense of the definition by Katz, Sahai and Waters), while the existing IPE schemes are either fully attribute-hiding but selectively secure or adaptively secure but weakly attribute-hiding. The IPE scheme was proposed in Eurocrypt 2012 [1], and is proven to be adaptively secure and fully attribute-hiding under the decisional linear assumption in the standard model. The IPE scheme is comparably as efficient as the existing attribute-hiding IPE schemes. We also present a variant of the proposed IPE scheme with the same security that achieves shorter public and secret keys. A hierarchical IPE scheme can be constructed that is also adaptively secure and fully attribute-hiding under the same assumption. In this work, we extend the dual system encryption technique by Waters into a more general manner, in which new forms of ciphertext and secret keys are employed and new types of information theoretical tricks are introduced along with several forms of computational reduction. This is joint work with Tatsuaki Okamoto.

Reference

1. Okamoto, T., Takashima, K.: Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012), full version is available at http://eprint.iacr.org/2011/543

Table of Contents

Industry Track

Properties of Pairings

Signature Schemes and Applications

On Efficient Pairings on Elliptic Curves over Extension Fields

Xusheng Zhang[1,2], Kunpeng Wang[3], and Dongdai Lin[3]

[1] Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China
[2] Graduate University of Chinese Academy of Sciences, Beijing, 100049, China
xszhang.is@gmail.com
[3] SKLOIS, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, 100195, China
kunpengwang@263.net, ddlin@iie.ac.cn

Abstract. In implementation of elliptic curve cryptography, three kinds of finite fields have been widely studied, i.e. prime field, binary field and optimal extension field. In pairing-based cryptography, however, pairing-friendly curves are usually chosen among ordinary curves over prime fields and supersingular curves over extension fields with small characteristics. In this paper, we study pairings on elliptic curves over extension fields from the point of view of accelerating the Miller's algorithm to present further advantage of pairing-friendly curves over extension fields, not relying on the much faster field arithmetic. We propose new pairings on elliptic curves over extension fields can make better use of the multi-pairing technique for the efficient implementation. By using some implementation skills, our new pairings could be implemented much more efficiently than the optimal ate pairing and the optimal twisted ate pairing on elliptic curves over extension fields. At last, we use the similar method to give more efficient pairings on Estibals's supersingular curves over composite extension fields in parallel implementation.

Keywords: pairing, elliptic curve over extension field, multi-pairing technique.

1 Introduction

Elliptic curve cryptography (ECC) has the shorter key length requirement in comparison with other public-key cryptosystems such as RSA. This means faster implementation as well as more efficient use of power, bandwidth and storage. In particular, much research has been conducted on fast algorithms and implementation techniques of elliptic curve arithmetic over various finite fields. Up to now, three kinds of finite fields are widely used for ECC, i.e. prime field, binary field and optimal extension field. Binary fields $\mathbb{F}(2^m)$ are especially attractive for hardware circuit design, but does not offer the same computational advantages in a software implementation. Similarly, prime fields $\mathbb{F}(p)$ also have computational difficulties on standard computers. Optimal extension fields $\mathbb{F}(p^m)$ introduced in [1,2], offer considerable computational advantages in software by selecting p

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 1–18, 2013.

and m specifically to match the underlying hardware used to perform the arithmetic. Besides, efficient methods have been devised in [27,3] for speeding up field arithmetic for elliptic curves over general extension fields.

In recent years, there has been much interest in cryptographic schemes based on bilinear pairings on elliptic curves. So efficient implementation of pairings is of great importance. Miller [29] proposed the first effective algorithm named Miller's algorithm to compute Weil pairing and Tate pairing. As the important breakthroughs, there are many optimizations and adaptations of these pairings which offer implementation improvements, such as speeding up each Miller's iteration and the final exponentiation of the Tate pairing, and developing many truncated loop variant pairings: Eta pairing [5], ate pairing and twisted ate pairing [22], R-ate pairing [26], and optimal pairing [33]. Recently, pairing lattices [21] were proposed as the generalization contained all former pairings.

On the other side, there is much research on the generation of suitable elliptic curves for pairings, namely pairing-friendly curves, which contain the large prime subgroup and the small embedding degree. Please refer to the in-depth overview [12] for details. Whereas strong elliptic curves used in ECC can be generated randomly, the pairing-friendly curves are rare and require specific constructions. All the time, pairing-friendly curves are chosen among ordinary curves over prime fields and supersingular curves over extension fields with the characteristic 2 and 3. In the latter case, pairings are suitable for hardware implementation in lightweight cryptosystems. For higher security, pairings on ordinary pairing-friendly curves are preferred in practice.

In implementation, there are always some strong requests to use curves defined over certain extension fields, such as the extension fields with small characteristics, and the optimal extension fields which possess the fast field multiplication and inversion. So there are theoretical advantages to using pairing-friendly elliptic curves over carefully chosen finite fields. Recently, Hitt [23] and Benger *et al.* [6] outlined possible security concerns for using pairing-friendly elliptic curves defined over extension fields, and Benger *et al.* [6] gave a method for selecting curves with the highest possible security against ECDLP and DLP solving attacks, given currently known methods. To the best of our knowledge, there is still no known example of an ordinary pairing-friendly curve defined over the extension field $\mathbb{F}_{p^m}$ or $\mathbb{F}_{2^m}$. Hence, we present results which may motivate further research into the generation of pairing-friendly elliptic curves defined over extension fields.

In this paper, our main aim is to present further evidence of an advantage of using pairing-friendly elliptic curves defined over extension fields by introducing a pairing which can be computed using an accelerated version of Miller's algorithm, using the multi-pairing technique. We develop new pairings on an elliptic curve over an extension field which could be computed more efficiently not relying on the fast field arithmetic of the extension field. Concretely, for an ordinary curve E over an extension field $\mathbb{F}_{p^m}$, we modify the ate pairing and the twisted ate pairing to define new pairings as the products of several rational functions with the same Miller loop on the curves $\{E^{(p^i)}\}_{0\leq i<m}$ defined by raising the coefficients of the

equations for E to the p^i-power. These new pairings can be implemented with the multi-pairing technique which was proposed in [31,19] and first applied to a single pairing computation by Sakemi *et al.* [30]. Then we give the optimal versions of our new pairings according to the theory of pairing lattice [21], which can make better use of the multi-pairing technique for efficient implementation. Specially, our method can explain Sakemi's acceleration [30] of the twisted ate pairing on the BN curves and extend it further. Given a theoretical comparison with some implementation skills, our new optimal pairings could have more efficient performance than the optimal ate pairing and the optimal twisted ate pairing. Specially in many protocols, with the fixed argument optimization, the performance of our new optimal pairing could offer a speed up of between 30% and 43% faster than the performance of the optimal ate pairing when m is greater than 6. Finally, we develop similar pairings having much faster parallel implementation on supersingular curves over composite extension fields, and then construct concrete pairings on Estibals's supersingular curves $E_1(\mathbb{F}_{3^{5\times 97}})$ and $E_2(\mathbb{F}_{3^{17\times 67}})$ respectively.

The organization is given as: Section 2 recalls basics of pairing on elliptic curve and multi-pairing technique, and lists known conditions on suitably chosen extension fields for pairing-based cryptography; in Section 3 we propose new faster pairings on ordinary curves over extension fields; then in Section 4 we analyze the theoretical performance of our new optimal pairings compared to the optimal ate pairing and optimal twisted ate pairing; in Section 5 we extend the similar method to supersingular curves over composite extension fields.

2 Background

2.1 Bilinear Pairing

Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$ where q is a prime power, and the neutral element of which is denoted by $\mathcal{O}$. Let $r \geq 5$ be a prime factor of $|E(\mathbb{F}_q)|$ and let $k > 1$ be the smallest integer such that $r|q^k - 1$ which is named the embedding degree with respect to r. Here we define $G_1 = E[r] \cap \mathrm{Ker}(\pi_q - 1)$ and $G_2 = E[r] \cap \mathrm{Ker}(\pi_q - q)$ as the two eigenspaces of the q-power Frobenius endomorphism π_q on E. Let $\mu_r \subset \mathbb{F}_{q^k}^*$ denote the group of r-th roots of unity. For $s \in \mathbb{Z}$ and $R \in E[r]$, let $f_{s,R}$ be a $\mathbb{F}_{q^k}$-rational function with divisor $\mathrm{div}(f_{s,R}) = s(R) - ([s]R) - (s-1)(\mathcal{O})$.

Tate Pairing and Its Variants. The reduced Tate pairing [4] is given by

$$t_r : G_1 \times G_2 \to \mu_r, \quad (P, Q) \mapsto f_{r,P}(Q)^{(q^k-1)/r}.$$

Let s be an integer such that $s \equiv q \pmod r$. When $r \nmid c \equiv \sum_{j=0}^{k-1} s^{k-1-j} q^j \pmod r$, the modified ate pairing [22] is given by

$$a_s : G_2 \times G_1 \to \mu_r, \quad (Q, P) \mapsto f_{s,Q}(P)^{(q^k-1)/r}.$$

Assume that $E/\mathbb{F}_q$ admits a degree-d twist. Let $e = k/\gcd(k,d)$ and $s' \in \mathbb{Z}$ satisfy that $s' \equiv q^e \pmod r$. The modified twisted ate pairing [22] is given by

$$a_{s'}^{twist} : G_1 \times G_2 \to \mu_r, \quad (P,Q) \mapsto f_{s',P}(Q)^{(q^k-1)/r}.$$

Then a_s and $a_{s'}^{twist}$ are non-degenerate if and only if $r \nmid L = (s^k-1)/r$.

For the convenience of the construction of new pairings, we use the variants $a(Q,P) = f_{q,Q}(P)^{(q^k-1)/r}$ and $a^{twist}(P,Q) = f_{q^e,P}(Q)^{(q^k-1)/r}$ instead of the above ate pairing and twisted ate pairing in the rest of this paper.

Miller's Algorithm. Let $f_{i,P}$ be the rational function with divisor $\mathrm{div}(f_{i,P}) = i(P) - ([i]P) - (i-1)(\mathcal{O})$, and $l_{R,S}$ is the line passing through points R, S and v_{R+S} is the vertical line passing through point $R+S$ with divisor $\mathrm{div}(l_{R,S}) = (R) + (S) + (-(R+S)) - 3(\mathcal{O})$ and $\mathrm{div}(v_{R+S}) = (R+S) + (-(R+S)) - 2(\mathcal{O})$. Using the fact that $f_{i_1+i_2,P} = f_{i_1,P} f_{i_2,P} l_{[i_1]P,[i_2]P} / v_{[i_1+i_2]P}$, Miller's algorithm [29] calculates the evaluation of $f_{i,P}(Q)$ recursively. In §2.2 Algorithm 1 is just the classical Miller's algorithm when assuming $N = 1$.

Optimal Pairing. In [33], Vercauteren proposed an important conception of a pairing having the "optimal" loop length. Let $e : G_1 \times G_2 \to \mu_r$ be a non-degenerate pairing with $|G_1| = |G_2| = r$, then e is called an optimal pairing if it can be computed in $\frac{1}{\varphi(k)} \log_2 r + \epsilon(k)$ basic Miller iterations, with $\epsilon(k) \leq \log_2 k$. Furthermore, Vercauteren conjectured that any non-degenerate pairing on an elliptic curve without efficiently computable endomorphisms different from powers of Frobenius, requires at least $O(\log_2(r)/\varphi(k))$ basic Miller iterations, where the O-constant only depends on k.

Pairing Lattices. Hess [21] generalized the conception of the optimal pairing to provide pairing lattices as a convenient mathematical framework to create pairings with optimal degrees of the divisors of pairing functions. Let $r \in \mathbb{Z}$ be an integer, and let s be a primitive n-th root of unity modulo r^i for $n \geq 2$ and $i \geq 1$. Define the $\mathbb{Z}$-module $I^{(i)} = \{h(t) + (t^n - 1)\mathbb{Z}[t] | h(s) \equiv 0 \pmod{r^i}\}$, and $||h||_1 = \sum_{i=0}^m |h_i|$. For $h(t) = \sum_{i=0}^m h_i t^i \in I^{(1)}$ and $R \in E(\mathbb{F}_{q^k})[r]$, let $f_{s,h,R}$ be the $\mathbb{F}_{q^k}$-rational function with divisor $\mathrm{div}(f_{s,h,R}) = \sum_{i=0}^m h_i\big(([s^i]R) - (\mathcal{O})\big)$. It is easy to deduce that $\mathrm{div}(f_{s,ht,R}) = \mathrm{div}(f_{s,h,[s]R})$ and $\mathrm{div}(f_{s,h+g,R}) = \mathrm{div}(f_{s,h,R} f_{s,g,R})$ for $g(t) \in I^{(1)}$.

The evaluation of $f_{s,h,R}(P)$ can be calculated analogously to the method for the optimal ate pairing in [33] (also cf. [34]). Following this analysis, we may assume that the length of the Miller loop for calculating $f_{s,h,R}$ is approximated by $\log_2 ||h||_1 + \epsilon$, where $\epsilon \leq \log_2 n$.

Theorem 1. *([21], Theorem 6) Assume that r is a prime, and s is a primitive n-th root of unity modulo r^2. Let W denote the multiplicative group of functions $G_1 \times G_2 \to \mu_r$, and W^{bilin} denote the subgroup of bilinear functions. Let $a_s : I^{(1)} \to W, h \mapsto a_{s,h}$ be a map with the following properties:*

1. $a_{s,g+h} = a_{s,g}a_{s,h}$ *for all* $g, h \in I^{(1)}$,
2. $a_{s,hx} = a_{s,h}^{s}$ *for all* $h \in I^{(1)}$ *with* $a_{s,h} \in W^{bilin}$,
3. $a_{s,r} \in W^{bilin} \setminus \{1\}$ *and* $a_{s,x-s} = 1$.

Then $\mathrm{Im}(a_s) = W^{bilin}$, $\ker(a_s) = I^{(2)}$. *More precisely,* $a_{s,h} = a_{s,r}^{h(s)/r}$ *for all* $h \in I^{(1)}$. *There exists an efficiently computable* $h \in I^{(1)}$ *with* $\|h\|_1 = O(r^{1/\varphi(n)})$. *Any* $h \in I^{(1)}$ *with* $a_{s,h} \neq 1$ *satisfies* $\|h\|_1 \geq r^{1/\varphi(n)}$.

Especially, the optimal ate pairing and the optimal twisted ate pairing are well-defined and probably constructed in the ate pairing lattice and the twisted ate pairing lattice in [21] with the optimal loop length $\log_2(r)/\varphi(k) + \epsilon_1$ and $\log_2(r)/\varphi(d) + \epsilon_2$.

2.2 Multi-pairing Technique

In many protocols the evaluation of the products of the form $\prod_{i=1}^{N} t_r(P_i, Q_i)$ is required. A naive way to calculate it is to evaluate each $t_r(P_i, Q_i)$ independently, and then multiply the results. Since all $t_r(P_i, Q_i)$ share some same Miller operations, Scott [31] and Granger and Smart [19] showed the products can be calculated in a single Miller algorithm rather than the naive way. The multi-Miller algorithm only needs a single squaring in the extension field per doubling, instead of N squarings in the naive method, and also combines the final powerings required in each pairing evaluation. As far as we know, this method is usually named multi-pairing algorithm given in Algorithm 1.

Algorithm 1. Miller's Algorithm for Multi-pairing

Input: $s = \sum_{j=0}^{L} s_j 2^j \in \mathbb{N}$ (2-adic), $N \in \mathbb{N}$, $\{P_1, P_2, \cdots, P_N\}$, $\{Q_1, Q_2, \cdots, Q_N\}$
Output: $\prod_{i=1}^{N} f_{s,P_i}(Q_i)$, $\{[s]P_1, [s]P_2, \cdots, [s]P_N\}$
1: $f \leftarrow 1$
2: **for** i from N downto 1 **do**
3: $\quad T_i \leftarrow P_i$
4: **for** j from $L-1$ downto 0 **do**
5: $\quad f \leftarrow f^2$
6: $\quad$ **for** i from N downto 1 **do**
7: $\quad\quad f \leftarrow f \cdot l_{T_i,T_i}(Q)/v_{[2]T_i}(Q_i)$; $\quad T_i \leftarrow [2]T_i$
8: $\quad$ **if** $s_j = 1$ **then**
9: $\quad\quad$ **for** i from N downto 1 **do**
10: $\quad\quad\quad f \leftarrow f \cdot l_{T_i,P_i}(Q_i)/v_{T_i+P_i}(Q_i)$; $\quad T_i \leftarrow T_i + P_i$
11: **return** f.

However, not only can the multi-pairing technique be used to calculate the products of pairings, but it also can be applied to calculate a single pairing defined as the products of several rational functions with the same Miller loop. In [30], Sakemi *et al.* utilized the multi-pairing technique to calculate the improved twisted ate pairing on the BN curves with the sophisticated reduction. We extend this idea to the implementation of pairings considered in this paper.

2.3 Suitable Extension Field for Pairing-Based Cryptography

In the rest of this paper we always assume that there is a pairing-friendly curve E defined over an extension field $\mathbb{F}_q$ with $q = p^m$. Let r divide $|E(\mathbb{F}_q)|$ but do not divide any other $|E(\mathbb{F}_{p^i})|$ for $1 \leq i < m$. We list some well-known results of the security extension fields for ECC and Pairing-Based Cryptography, and show our suitable choice of the extension fields for the comparison in Section 4.

Attack on ECDLP over Extension Field. Weil descent proposed by Frey [13] aims at transferring the DLP from $E(\mathbb{F}_{q^m})$ to the Jacobian of a curve C over $\mathbb{F}_q$ and then computes the logarithm on this Jacobian by using index calculus. Many researches [15,17,14,20,28] have studied on the scope of this technique on the vulnerable curves over binary fields. Diem [9] extended this attack in odd characteristic.

Later, Gaudry [16] developed decomposition-based index calculus, which applies to all (hyper-)elliptic curves defined over small degree extension field with the running time $O(q^{2-2/m})$ for $m \geq 3$. Diem [10] proved that Gaudry's algorithm has subexponential running time when the field order p^m increases in such a way that m^2 is of order $\log_2 p$. Later, Joux and Vitse [24] improved this index calculus, when $m > 5$ and $\log_2 p \leq O(m^3)$.

But, both Weil descent and decomposition-based index calculus are often just a little more efficient than generic attacks, and ineffective for solving the ECDLP in practice.

The Static Diffie-Hellman Problem. The Static Diffie-Hellman problem (Static DHP) on an elliptic curve consists of: for a secret integer d, given two points $P, [d]P \in E(\mathbb{F}_q)$ and an oracle $Q \mapsto [d]Q$, compute $[d]R$ where R is randomly chosen point. Recently Granger [18] discovered the best known algorithm that solves the Static DHP problem on elliptic curves defined over a finite field of composite extension degree $\mathbb{F}_{q^n}$ by making $O(q^{1-\frac{1}{n+1}})$ Static DHP oracle queries and in *heuristic* time $O(q^{1-\frac{1}{n+1}})$. Estibals [11] showed that a simple but efficient protection against this attack is revoking a key after a certain amount of use.

Minimal Embedding Field. The embedding degree k should be small enough that the pairing is efficiently computable, but large enough that the DLP in $\mathbb{F}_{q^k}^*$ is hard. However, Hitt [23] showed that the minimal finite field ensures the ECDLP of $E(\mathbb{F}_q)[r]$ secure is not necessarily $\mathbb{F}_{q^k}$, but rather is $\mathbb{F}_{p^{\mathrm{ord}_r(p)}} = \mathbb{F}_{q^{\mathrm{ord}_r(p)/m}}$. Then $\mathbb{F}_{q^{\mathrm{ord}_r(p)/m}}$ is named the minimal embedding field and coincides with the traditional assumptions when $m = 1$. Later, Benger *et al.* [6] gave explicit conditions on q, k, and r, which (when satisfied) imply that the minimal embedding field of E with respect to r is $\mathbb{F}_{q^k}$.

Theorem 2. *([6], Corollary 2.10) Let A be an abelian variety over F_q, where $q = p^m$ with p prime. Let $r \neq p$ be a prime dividing $|A(F_q)|$, and suppose A has embedding degree k with respect to r. Assume that $r \nmid km$. Write $m = \alpha\beta$, where every prime dividing α also divides k and $\gcd(k, \beta) = 1$. (This factorization is*

unique.) Denote by e the smallest prime factor of β. If q, k, and r satisfy any of the following conditions:

1. $m = \alpha$ *(and* $\beta = 1$*);*
2. β *is prime and* $r > \Phi_{k\alpha}(p)$*;*
3. $r > p^{km/e}$*;*
4. $4|m$ *or* $2|k$ *and* $r > p^{km/2e} + 1$.

Then the minimal embedding field of A with respect to r is $\mathbb{F}_{p^{km}}$.

Hence, in this paper we prefer to choose a large prime p and an integer $m \geq 5$ to prevent the known attacks in practice. If there exist algorithms to generate pairing-friendly curves over $\mathbb{F}_{p^m}$ defined in [12], we may restrict m, p, k and r to satisfy one of the conditions in Theorem 2. For the comparison in Section 4, we use even embedding degrees of the form $k = 2^i 3^j$ and examine examples using: $m = 7, 11$ $(m > \phi(k))$, such that condition (2) of Theorem 2 is satisfied; and, $m = 8, 9$, such that condition (1) of Theorem 2 is satisfied.

3 New Pairings on Elliptic Curve over Extension Field

In this section we propose new pairings on an elliptic curve E over an extension field $\mathbb{F}_q$ which make better use of the multi-pairing technique to speed up their implementation. We first transform the ate pairing $a(Q,P) = f_{q,Q}(P)^{(q^k-1)/r}$ and the twisted ate pairing $a^{twist}(P,Q) = f_{q^e,P}(Q)^{(q^k-1)/r}$ as follows.

Theorem 3. *Let E be an ordinary elliptic curve defined over $\mathbb{F}_q$ with $q = p^m$. Let r be a prime such that r divides $|E(\mathbb{F}_q)|$ and $\gcd(r,p) = 1$. Let k be the minimal embedding degree with respect to r. Let $E^{(p^i)}$ be denoted the curve defined by raising the coefficients of the equation for E to the p^i-power for $0 \leq i < m$. Let π_{p^i} and $\widehat{\pi}_{p^i}$ be the p^i-power Frobenius isogeny and its dual isogeny from every $E^{(p^j)}$ to $E^{(p^{j+i})}$. For $P \in G_1$ and $Q \in G_2$, then*

$$\bar{a}(Q,P) = \left(\prod_{i=0}^{m-1} f_{p,\widehat{\pi}_{p^i}(Q)}\left(\pi_{p^{m-i}}(P)\right) \right)^{(p^{mk}-1)/r}$$

defines a pairing.

Assume that $E/\mathbb{F}_q$ admits a degree-d twist $E'/\mathbb{F}_{q^e}$ with $e = k/\gcd(k,d)$ and $d \geq 2$. Let ψ be the associated twist isomorphism $\psi : E \to E'$. Then

$$\hat{a}(Q,P) = \left(\prod_{i=0}^{m-1} f_{p,\widehat{\pi}_{p^i}\circ\psi(Q)}\left(\pi_{p^{mk-i}} \circ \psi(P)\right) \right)^{(p^{mk}-1)/r}$$

and

$$\bar{a}^{twist}(P,Q) = \left(\prod_{i=0}^{m-1} \prod_{j=0}^{e-1} f_{p,\widehat{\pi}_{p^i}([p^{mj}]P)}\left(\pi_{p^{mk-i}}(Q_{e-j-1})\right) \right)^{(p^{mk}-1)/r}$$

define pairings, where $Q_j = \pi_{p^{mj}}(Q)$ for $0 \leq j \leq e-1$.

Proof. Since $[p^i] = \pi_{p^i} \circ \widehat{\pi}_{p^i}$ with $\pi_{p^i} : E^{(p^{m-i})} \to E$ for some i, it follows that for $R \in E(\mathbb{F}_{p^k})[r]$, $\pi_{p^i}^* \mathrm{div}(f_{p,[p^i]R}) = \pi_{p^i}^* \big(p([p^i]R) - ([p^{i+1}]R) - (p-1)(\mathcal{O})\big) = p^i \big(p(\widehat{\pi}_{p^i}(R)) - (\widehat{\pi}_{p^i}([p]R)) - (p-1)(\mathcal{O})\big) = \mathrm{div}(f^{p^i}_{p,\widehat{\pi}_{p^i}(R)})$, where $\pi_{p^i}^*$ is the pullback of π_{p^i}. Thus $f_{p,[p^i]R} \circ \pi_{p^i} = f^{p^i}_{p,\widehat{\pi}_{p^i}(R)} \in \mathbb{F}_{q^k}(E^{(p^{m-i})})$. If $R = Q$, then $f_{p,[p^i]Q}(P) = f_{p,\widehat{\pi}_{p^i}(Q)}(\pi_{p^{m-i}}(P))^{p^i}$; if $R = P$, then $f_{p,[p^i]P}(Q) = f_{p,\widehat{\pi}_{p^i}(P)}(\pi_{p^{mk-i}}(Q))^{p^i}$. When E admits a twist of degree d, if $R = Q' = \psi(Q) \in E'(\mathbb{F}_{q^e})[r]$ and $P' = \psi(P) \in E'(\mathbb{F}_{q^k})[r]$, then $f_{p,[p^i]Q'}(P') = f_{p,\widehat{\pi}_{p^i}(Q')}(\pi_{p^{mk-i}}(P'))^{p^i}$.

Since $\gcd(p, r) = 1$, there exits an integer M such that $Mp^{m-1} \equiv 1 \pmod{r}$. Note that a power of a nondegenerate pairing is also a nondegenerate pairing when the power and the pairing order are coprime. Thus we can do the following reduction for a fixed power M of the ate pairing $a(Q, P)$.

$$a(Q,P)^M = f_{q,Q}(P)^{M(q^k-1)/r} = \prod_{i=0}^{m-1} f_{p,[p^i]Q}(P)^{p^{m-i-1}M(q^k-1)/r}$$
$$= \prod_{i=0}^{m-1} f_{p,\widehat{\pi}_{p^i}(Q)}(\pi_{p^{m-i}}(P))^{Mp^{m-1}(q^k-1)/r} = \prod_{i=0}^{m-1} f_{p,\widehat{\pi}_{p^i}(Q)}(\pi_{p^{m-i}}(P))^{(q^k-1)/r}.$$

When E admits a twist of degree d, then $a(Q', P') = f_{q,Q'}(P')^{(q^k-1)/r}$ also defines a pairing from Theorem 1 in [7], where $P' = \psi(P) \in E'(\mathbb{F}_{q^k})[r]$ and $Q' = \psi(Q) \in E'(\mathbb{F}_{q^e})[r]$. So a similar reduction can be done for $a(Q', P')^M$ as

$$a(Q',P')^M = f_{q,Q'}(P')^{M(q^k-1)/r} = \prod_{i=0}^{m-1} f_{p,[p^i]Q'}(P')^{p^{m-i-1}M(q^k-1)/r}$$
$$= \prod_{i=0}^{m-1} f_{p,\widehat{\pi}_{p^i}(Q')}(\pi_{p^{mk-i}}(P'))^{Mp^{m-1}(q^k-1)/r} = \prod_{i=0}^{m-1} f_{p,\widehat{\pi}_{p^i}(Q')}(\pi_{p^{mk-i}}(P'))^{(q^k-1)/r}.$$

For the twisted ate pairing, since $f_{p,\widehat{\pi}_{p^i}(P)} \in \mathbb{F}_q(E^{(p^{m-i})})$, let $Q_j = \pi_{q^j}(Q)$ for $0 \le j \le e-1$, it follows that $f_{p,\widehat{\pi}_{p^i}(P)}(\pi_{p^{mk-i}}(Q))^{q^j} = f_{p,\widehat{\pi}_{p^i}(P)}(\pi_{p^{mk-i}}(Q_j))$. Thus we have that

$$a^{twist}(P,Q)^M = f_{q^e,P}(Q)^{M(q^k-1)/r} = \prod_{j=0}^{e-1} f_{q,[q^i]P}(Q)^{q^{e-i-1}M(q^k-1)/r}$$
$$= \prod_{j=0}^{e-1}\prod_{i=0}^{m-1} f_{p,[p^{mj+i}]P}(Q)^{q^{e-i-1}p^{m-i-1}M(q^k-1)/r}$$
$$= \prod_{j=0}^{e-1}\prod_{i=0}^{m-1} f_{p,[p^{mj+i}]P}(Q_{e-j-1})^{p^{m-i-1}M(q^k-1)/r}$$
$$= \prod_{j=0}^{e-1}\prod_{i=0}^{m-1} f_{p,\widehat{\pi}_{p^i}([p^{mj}]P)}(\pi_{p^{mk-i}}(Q_{e-j-1}))^{(q^k-1)/r}.$$

Write $\bar{a}(Q,P) = a(Q,P)^M$, $\hat{a}(Q,P) = a(Q',P')^M = a(\psi(Q),\psi(P))^M$ and $\bar{a}^{twist}(P,Q) = a^{twist}(P,Q)^M$. Thus they define new pairings. □

Theorem 3 shows that the ate pairing and the twisted ate pairing on the curve E over $\mathbb{F}_{p^m}$ can be modified as the products of several rational functions with the same Miller loop on the curves $\{E^{(p^i)}\}_{0\le i<m}$. Next we give the optimal versions of the new pairings in Theorem 3 according to the theory of pairing lattices.

Theorem 4. *Use the notations in Theorem 3. Let s be a primitive (mk)-th root of unity modulo r^2 such that $s \equiv q \pmod r$. Let $h \in \mathbb{Z}[t]$ satisfy $h(s) \equiv 0 \pmod r$. For $P \in G_1$ and $Q \in G_2$, following the respective assumptions for $\bar{a}, \hat{a}, \bar{a}^{twist}$ of Theorem 3, then*

$$\bar{a}_{s,h}(Q,P) = \Big(\prod_{i=0}^{m-1} f_{s,h,\widehat{\pi}_{p^i}(Q)}\big(\pi_{p^{m-i}}(P)\big)\Big)^{(p^{mk}-1)/r},$$

$$\hat{a}_{s,h}(Q,P) = \Big(\prod_{i=0}^{m-1} f_{s,h,\widehat{\pi}_{p^i}\circ\psi(Q)}\big(\pi_{p^{mk-i}}\circ\psi(P)\big)\Big)^{(p^{mk}-1)/r},$$

$$\bar{a}^{twist}_{s,h}(P,Q) = \Big(\prod_{i=0}^{m-1}\prod_{j=0}^{e-1} f_{s,h,\widehat{\pi}_{p^i}([p^{mj}]P)}\big(\pi_{p^{mk-i}}(Q_{e-j-1})\big)\Big)^{(p^{mk}-1)/r}$$

define pairings, which are nondegenerate if and only if $h(s) \not\equiv 0 \pmod{r^2}$.

There exists an efficiently computable $h \in I^{(1)}$ with $\|h\|_1 = O(r^{1/\varphi(mk)})$. Any $h \in I^{(1)}$ with $a_{s,h} \neq 1$ satisfies $\|h\|_1 \ge r^{1/\varphi(mk)}$.

Proof. Since $f_{s,g+h,R} = f_{s,g,R}f_{s,h,R}$ and $f_{s,hx,R} = f_{s,h,[s]R}$ for $h, g \in I^{(1)}$, it follows that $\bar{a}_{s,g+h} = \bar{a}_{s,g}\bar{a}_{s,h}$, $\hat{a}_{s,g+h} = \hat{a}_{s,g}\hat{a}_{s,h}$, $\bar{a}^{twist}_{s,g+h} = \bar{a}^{twist}_{s,g}\bar{a}^{twist}_{s,h}$, and $\bar{a}_{s,hx} = (\bar{a}_{s,h})^s$, $\hat{a}_{s,hx} = (\hat{a}_{s,h})^s$, $\bar{a}^{twist}_{s,hx} = (\bar{a}^{twist}_{s,h})^s$ for the pairings $\bar{a}_{s,h}$, $\hat{a}_{s,h}$ and $\bar{a}^{twist}_{s,h}$. Let $t^{(i)}_r$ denote the Tate pairing on $E^{(p^i)}[r]$. Since $f_{r,R} = f_{s,r,R}$, we have

$$\bar{a}_{s,r}(Q,P) = \Big(\prod_{i=0}^{m-1} f_{r,\widehat{\pi}_{p^i}(Q)}(\pi_{p^{m-i}}(P))\Big)^{(p^{mk}-1)/r} = \prod_{i=0}^{m-1} t^{(i)}_r(\widehat{\pi}_{p^i}(Q), \pi_{p^{m-i}}(P)).$$

Write $t_i(Q,P) = t^{(i)}_r(\widehat{\pi}_{p^i}(Q), \pi_{p^{m-i}}(P))$, then each $t_i(Q,P)$ is a pairing on $E[r]$. As with the proof of Theorem 3, we have $f_{r,[p^i]Q}(P) = f_{r,\widehat{\pi}_{p^i}(Q)}(\pi_{p^{m-i}}(P))^{p^i}$, and furthermore $t([p^i]Q,P) = t_i(Q,P)^{p^i}$. Thus $\bar{a}_{s,r}(Q,P) = t_r(Q,P)^m$ is a pairing on $E[r]$.

Let $c \in \mathbb{Z}$ satisfy $s = p + cr$ and let $c_0 \in \mathbb{Z}$ satisfy $p^{mk} \equiv 1 + c_0 r \pmod{r^2}$, then $s^{mk} = (p+cr)^{mk} \equiv 1 + c_0 r + mkp^{mk-1}cr \equiv 1 \pmod{r^2}$. Thus $c_0 \equiv -mkp^{mk-1}c \pmod r$. We know that $a(Q,P)^{kp^{m(k-1)}} = t_r(Q,P)^{c_0}$ in [22]. From the proof Theorem 3, we have $\bar{a}(Q,P)^{p^{m-1}} = a(Q,P) = t_r(Q,P)^{-mp^{m-1}c}$. We conclude that $\bar{a}_{s,x-s}(Q,P)^{-1} = \bar{a}_{s,s-x}(Q,P) = \bar{a}(Q,P)\bar{a}_{s,r}(Q,P)^c = 1$.

Similarly, it can be demonstrated that $\hat{a}_{s,x-s}(P,Q) = 1$, $\bar{a}^{twist}_{s,x-s}(P,Q) = 1$, and $\hat{a}_{s,r}(Q,P) = t(Q',P')^m$, $\bar{a}^{twist}_{s,r}(P,Q) = t(P,Q)^{me}$ are pairings.

From Theorem 1, we conclude that for every h satisfying the conditions, $\bar{a}_{s,h}$, $\hat{a}_{s,r}$, and $\bar{a}_{s,h}^{twist}$ are nondegenerate if and only if $h(s) \not\equiv 0 \pmod{r^2}$. □

From Theorem 4, we may construct an optimal h satisfying the conditions of Theorem 4 and $\|h\|_1 = O(r^{1/\varphi(mk)})$ so that each pairing $\bar{a}_{s,h}$, $\hat{a}_{s,h}$ and $\bar{a}_{s,h}^{twist}$ has the optimal multi-Miller loop length $\log_2(r)/\varphi(mk) + \epsilon$, which is smaller than the traditional optimal loop length. We name these pairings the optimal $\bar{a}_{s,h}$, $\hat{a}_{s,h}$ and $\bar{a}_{s,h}^{twist}$. However, the implementations of these pairings involve the calculations of $\widehat{\pi}_{p^i}(R)$ and $\pi_{p^j}(R')$ for some R and R'. In practice, the implementation of the Frobenius power costs little, but the implementation of the dual Frobenius isogeny (also called Verschiebung) might be costly. We introduce skills to perform this costly calculation in Section 4.

Explanation and Extension of Sakemi's Method. In [30], Sakemi *et al.* proposed a variant of the twisted ate pairing on the BN curves with $e = 2$ (and $m = 1$ in the setting of this paper), whose pairing function is given as

$$\begin{aligned}\hat{f}_{\chi,P}(Q) = &\big(f_{2\chi,P}(\pi_p(Q))f_{2\chi,[p]P}(Q)\big)^{p^{10}+1}\big(l_{[2\chi]P,-P}(\pi_p(Q))l_{[2\chi p]P,[-p]P}(Q)\big)^{p^{10}}\\ &\cdot l_{[(2\chi-1)p^{10}]P,[2\chi]P}(\pi_p(Q))l_{[(2\chi-1)p^{11}]P,[2\chi p]P}(Q).\end{aligned}$$

Using the method of this paper and the property of the twisted ate pairing [22], we conclude that $f_{T,[p^{je}]P}(Q) = f_{T,P}(Q)^{p^{je}}$ for any $T \in \mathbb{Z}$ and $j \geq 1$, and then choose $\hat{h}(t) = (2\chi-1)t^{10} - t + 2\chi$ to transform the pairing function of $\bar{a}_{s,\hat{h}}^{twist}$ in Theorem 4 under the final exponentiation (using subfield elimination) as follows.

$$\begin{aligned}&f_{s,\hat{h},P}(\pi_p(Q))f_{s,\hat{h},[p]P}(Q)\\ \equiv &\prod_{i=0,1} f_{2\chi-1,[p^{10+i}]P}(\pi_{p^{1-i}}(Q))f_{2\chi,[p^i]P}(\pi_{p^{1-i}}(Q))l_{[(2\chi-1)p^{10+i}]P,[2\chi p^i]P}(\pi_{p^{1-i}}(Q))\\ \equiv &\prod_{i=0,1}\big(f_{2\chi,[p^i]P}(\pi_{p^{1-i}}(Q))f_{[2\chi]P,-P}(\pi_{p^{1-i}}(Q))\big)^{p^{10}}\cdot f_{2\chi,[p^i]P}(\pi_{p^{1-i}}(Q))\\ &\cdot l_{[(2\chi-1)p^{10+i}]P,[2\chi p^i]P}(\pi_{p^{1-i}}(Q))\\ = &\hat{f}_{\chi,P}(Q).\end{aligned}$$

As a further extension, we utilize $h(t) = t^3 - t^2 + t + 6\chi + 2$, originally used for the optimal ate pairing on the BN curves in [33], to obtain another variant as

$$\begin{aligned}&f_{s,h,P}(\pi_p(Q))f_{s,h,[p]P}(Q)\\ \equiv &\prod_{i=0,1} f_{6\chi+2,[p^i]P}(\pi_{p^{1-i}}(Q))\big(l_{[p^{3+i}]P,[-p^{2+i}]P}l_{[p^{3+i}-p^{2+i}]P,[p^{1+i}]P}\big)(\pi_{p^{1-i}}(Q)).\end{aligned}$$

The linear part of the above pairing function of $\bar{a}_{s,h}^{twist}(P,Q)$ is calculated efficiently by using the skew Frobenius map $\tilde{\pi}_{p^2}$ as in [30] and the new congruence $(1-2\chi)p^2-p+4\chi-1 \equiv 0$, and the hard part can be carried out by $[p^2]P = \tilde{\pi}_{p^2}(P)$, $[p^4]P = \tilde{\pi}_{p^2}^2(P)$, $[p]P = [4\chi-1]P - \tilde{\pi}_{p^2}([(2\chi-1)]P)$, $[p^3]P = \tilde{\pi}_{p^2}([p]P)$.

4 Comparison

In this section we make a theoretical comparison between the optimal pairings in the pairing lattices in Theorem 4 and the optimal ate pairing and optimal twisted ate pairing, which depends on the assumptions of the existence of the optimal pairings for all pairing lattices and the existence of the pairing-friendly curves over extension fields.

Following the analysis in [19], we assume that $\mathbb{F}_{p^{mk}}$ is a pairing-friendly field with $p^m \equiv 1 \pmod{12}$ and $k = 2^i 3^j$, and quantify the cost of a multiplication in $\mathbb{F}_{p^{mk}}$ as $3^i 5^j$ multiplications in $\mathbb{F}_{p^m}$ (cf. [25]). In implementation, the loop parameter usually has a negligible Hamming weight so that few addition steps are encountered throughout the loop. Thus we only compare the operation counts for the doubling steps in Miller's algorithm. We list the up-to-date known results [7] of operation counts for the doubling step in Table 1.

Let $\mathbf{m}_1$, $\mathbf{m}_e$, $\mathbf{m}_k$ denote multiplication in $\mathbb{F}_q$, $\mathbb{F}_{q^e}$, $\mathbb{F}_{q^k}$; let $\mathbf{s}_1$, $\mathbf{s}_e$, $\mathbf{s}_k$ denote squaring in $\mathbb{F}_q$, $\mathbb{F}_{q^e}$, $\mathbb{F}_{q^k}$. The cost part 1 is taken to update the point used for constructing the new rational function; the cost part 2 is taken to evaluate the new rational function at the right argument; then the cost part 3 is taken to update the final rational function.

Table 1. Operation counts for single doubling step for the ate pairing and the twisted ate pairing

	Curve & twist degree	Cost part 1	Cost part 2	Cost part 3
ate $a(Q', P')$	$y^2 = x^3 + ax,\ d = 2, 4$ $y^2 = x^3 + b,\ d = 2, 6$	$2\mathbf{m}_e + 8\mathbf{s}_e + 1\mathbf{d}_a$ $2\mathbf{m}_e + 7\mathbf{s}_e + 1\mathbf{d}_b$	$2(\frac{k}{d})\mathbf{m}_1$	$1\mathbf{m}_k + 1\mathbf{s}_k$
twisted ate $a^{twist}(P, Q)$	$y^2 = x^3 + ax,\ d = 2, 4$ $y^2 = x^3 + b,\ d = 2, 6$	$2\mathbf{m}_1 + 8\mathbf{s}_1 + 1\mathbf{d}_a$ $2\mathbf{m}_1 + 7\mathbf{s}_1 + 1\mathbf{d}_b$	$2(\frac{k}{d})\mathbf{m}_1$	$1\mathbf{m}_k + 1\mathbf{s}_k$

Since the multi-pairing technique can save $m - 1$ squarings (using 2-basis) in each iteration when computing the products of m pairings (or functions with the same Miller loop), it follows that it is less efficient for the ate-like pairing computation compared with the twisted ate-like case. However, when the high-degree twist technique in [7] is available, the ate-like pairing computation can be still more efficient with the multi-pairing technique. Thus we assume that E admits a high-degree twist, and both the optimal ate pairing and twisted ate pairing have the loop length $\lceil \log_2(r)/\varphi(k) \rceil$, and both the optimal $\hat{a}_{s,h}$ and $\bar{a}_{s,h}^{twist}$ have the loop length $\lceil \log_2(r)/\varphi(mk) \rceil$. We show that the optimal $\hat{a}_{s,h}$ and $\bar{a}_{s,h}^{twist}$ could be implemented more efficient than the optimal ate pairing and the optimal twisted ate pairing when choosing suitable values of m and k in §2.3.

Precomputation vs. Storage. The calculation of pairings in Theorem 4 involves the calculation of $\widehat{\pi}_{p^i}(R)$ for $R \in E(\mathbb{F}_{p^{mk}})$ and $1 \leq i < m$. As far as we know, there is no efficient method to calculate the dual Frobenius isogeny on the general curves. Here we rewrite $\widehat{\pi}_{p^i}(R) = \pi_{p^{mk-i}}([p^i]R)$ by using $\widehat{\pi}_{p^i} \circ \pi_{p^i} = [p^i]$ and

$\pi_{p^{mk}}(R) = R$. Thus the costly part of this calculation is the multiplication by p^i. We introduce two skills to deal with it. One named the precomputation skill (P) utilizes the fixed argument optimization first pointed out by Scott [31] and recently analyzed in more detail (cf. [8,32]); the other named the storage skill (S) is proposed in this paper for computing our new pairings.

The first skill can be applied to many protocols in which the fixed argument optimization is feasible. With the fixed argument optimization, we can precompute all calculations depending solely on the lift argument R including the calculations of all $\{\widehat{\pi}_{p^i}(R)\}_{1\leq i<m}$. Hence, in each Miller iteration, the operations for the doubling step only involve the cost part 2 and the cost part 3 in Table 1. Besides, in this situation, there is no advantage of using a pairing-friendly curve with the maximal twist, and calculating a pairing in the twisted ate pairing family.

When the fixed argument optimization is infeasible, the precomputation is useless. But we could still store these calculations depending solely on the lift argument in each pairing computation, which are useful for the calculations of $\widehat{\pi}_{p^i}(R)$, and then we do the other calculations depending on the right argument. Taking the pairing $\bar{a}(Q,P)$ in Theorem 3 for example, we assume that $\widehat{\pi}_{p^i}(Q)$ is given for some $i \in [1, m-2]$. Then the calculation of the coefficients of $f_{p,\widehat{\pi}_{p^i}(Q)}$ involves $[p]\widehat{\pi}_{p^i}(Q) = \pi_{p^{mk-i}}([p^{i+1}]Q) = \pi_p(\widehat{\pi}_{p^{i+1}}(Q))$. Thus we can compute $\widehat{\pi}_{p^{i+1}}(Q)$ easily by using $\pi_{p^{mk-1}}([p]\widehat{\pi}_{p^i}(Q)) = \widehat{\pi}_{p^{i+1}}(Q)$, which is essential to the construction of $f_{p,\widehat{\pi}_{p^{i+1}}(Q)}$. This process only increases a few costs for implementing the Frobenius power, and needs the same additional memory compared with the precomputation skill which may be feasible in modern devices. Hence, we may omit the calculations of $\widehat{\pi}_{p^i}(R)$ when using our storage skill, and then give the comparisons below.

Table 2. The proportion of the runtime cost of the Miller loop of the optimal ate pairing to the optimal $\hat{a}_{s,h}$

	Skill	$k=8$ $d=4$	$k=12$ $d=6$	$k=16$ $d=4$	$k=18$ $d=6$	$k=24$ $d=6$	$k=32$ $d=4$	$k=36$ $d=6$
$m=7$	S	1 : 0.860	1 : 0.795	—	1 : 0.794	—	—	—
	P	1 : 0.701	1 : 0.688	—	1 : 0.686	—	—	—
$m=8$	S	1 : 0.732	1 : 0.675	1 : 0.727	1 : 0.673	1 : 0.671	1 : 0.725	1 : 0.670
	P	1 : 0.593	1 : 0.581	1 : 0.583	1 : 0.579	1 : 0.575	1 : 0.576	1 : 0.574
$m=9$	S	—	1 : 0.669	—	1 : 0.668	1 : 0.666	—	1 : 0.665
	P	—	1 : 0.574	—	1 : 0.573	1 : 0.568	—	1 : 0.567
$m=11$	S	1 : 0.793	1 : 0.728	1 : 0.789	1 : 0.727	1 : 0.724	—	—
	P	1 : 0.669	1 : 0.621	1 : 0.623	1 : 0.619	1 : 0.614	—	—

Optimal Ate Pairing vs. Optimal $\hat{a}_{s,h}$. In Table 2 we make a theoretical implementation comparison between the optimal ate pairing and the optimal $\hat{a}_{s,h}$ for some suitable embedding degrees and extension degrees, when ignoring the final exponentiation and using the precomputation skill or the storage skill.

Table 2 shows that the implementation of the optimal $\hat{a}_{s,h}$ improves the runtime cost of the Miller iterations by between 30% and 43% when using the precomputation skill, and between 14% and 34% when using the storage skill.

Optimal Twisted Ate Pairing vs. Optimal $\bar{a}_{s,h}^{twist}$. Since the fixed argument technique is mainly used for pairings of the ate family in practice, we only compare the theoretical implementation of the optimal twisted ate pairing with the optimal $\hat{a}_{s,h}^{twist}$ for some suitable embedding degrees and extension degrees, by using the storage skill and ignoring the final exponentiation. Table 3 shows that the implementation of the optimal $\bar{a}_{s,h}^{twist}$ improves the runtime cost of the Miller iterations by between 26% and 47%.

Table 3. The proportion of the runtime cost of the Miller loop of the optimal twisted ate pairing to the optimal $\bar{a}_{s,h}^{twist}$

	Skill	$k=8$ $d=4$	$k=12$ $d=6$	$k=16$ $d=4$	$k=18$ $d=6$	$k=24$ $d=6$	$k=32$ $d=4$	$k=36$ $d=6$
$m=7$	S	1 : 0.736	1 : 0.693	—	1 : 0.662	—	—	—
$m=8$	S	1 : 0.628	1 : 0.590	1 : 0.564	1 : 0.564	1 : 0.544	1 : 0.533	1 : 0.532
$m=9$	S	—	1 : 0.587	—	1 : 0.562	1 : 0.543	—	1 : 0.531
$m=11$	S	1 : 0.683	1 : 0.641	1 : 0.616	1 : 0.615	1 : 0.594	—	—

5 Our Method for Supersingular Curve over Extension Field

As the earliest pairing-friendly curves utilized in pairing-based cryptography, supersingular curves have embedding degree $k = 2, 3, 4$ and 6. However, for the recommended supersingular pairing-friendly curves with $k = 4$ and 6, there are two obstacles to applying our method: (1) their defining fields $\mathbb{F}_{2^n}$ and $\mathbb{F}_{3^n}$ usually have large prime extension degrees; (2) the main advantage of applying multi-pairing technique, namely saving squarings (using 2-basis) or cubings (using 3-basis) in each iteration, might be worthless for these supersingular curves, since squaring or cubing can be implemented very fast.

But recently, Estibals [11] first considered the Tate pairing computation for supersingular curves over moderately-composite extension fields taking advantage of a much easier tower field arithmetic. Our method can be applied to Estibals's curves over composite extension fields to define new pairings $\bar{\eta}_{s,h}$, which can be implemented in an efficient and parallel way.

Theorem 5. *Let E be a supersingular curve over a composite extension field $\mathbb{F}_{q^m}$ with the embedding degree k. Let r be a large integer dividing $|E(\mathbb{F}_{q^m})|$ and let ψ be the distortion map. Let s be a primitive (mk)-th root of unity modulo r^2 such that $s \equiv q \pmod{r}$. Let $h(t) \in \mathbb{Z}[t]$ such that $h(s) \equiv 0 \pmod{r}$. For $P, Q \in E(\mathbb{F}_{q^m})[r]$, then*

$$\bar{\eta}_{s,h}(P,Q) = \left(\prod_{i=0}^{m-1} f_{s,h,P}\left(\psi([q^{-i}]Q)\right)^{q^i} \right)^{(q^{mk}-1)/r}.$$

defines a pairing, which is non-degenerate if and only if $h(s) \not\equiv 0 \pmod{r^2}$.

Proof. Given in Appendix A. □

Write $f_i(P,Q) = f_{s,h,P}\left(\psi([q^{-i}]Q)\right)^{p^i}$, then $\bar{\eta}_{s,h}(P,Q) = \prod_{i=0}^{m-1} f_i(P,Q)^{(q^{mk}-1)/r}$. When precomputing all $[q^{-i}]Q$ for $1 \le i \le m-1$, we could compute these $f_i(P,Q)$ in a natural parallel and efficient way, since they share the common pairing function $f_{s,h,P}$ whose coefficients could be computed and stored first.

5.1 Estibals's Supersingular Curve over Composite Extension Field

There are several supersingular curves of characteristic 2 and 3 on fields with composite extension degree large enough for the 128-bit or 192-bit security level given in [11]. Here, we take two most important curves $E_1(\mathbb{F}_{3^{5\times 97}})$ (128-bit security level) and $E_2(\mathbb{F}_{3^{17\times 67}})$ (192-bit security level) for example to construct the corresponding $\bar{\eta}_{s,h}$.

- $E_1(\mathbb{F}_{3^{5\times 97}}) : y^2 = x^3 - x - 1, \quad (q_1 = 3^{97}, m_1 = 5, k = 6)$
  $r_1 = 434A97AFECDEB84F16624099C436CA9DE0CE4526690A8F0B24$
  $09B61DACB97A4411F3ED1CD3F39A6647D45$ (338 *bits*)
- $E_2(\mathbb{F}_{3^{17\times 67}}) : y^2 = x^3 - x + 1, \quad (q_2 = 3^{67}, m_2 = 17, k = 6)$
  $r_2 = 4A40FE5A48A1956BEEEC98D0147445A190711D0FCA4FCD5A65$
  $598194911D4D9F5D32156CAB3B4C9D53D02B3793E8AA2B1BAD8383$
  $2815DABA55EE9A2CD28A38027D2EB2FD0B6E4BEFD03DA273CD$
  $DDC19A1507E36281BC212F28F78EA379AEE4A3353C8348E13F5890D$
  $AA8367040520FC04B2E073193BE13922CEA13F106C9D8A8FE546D2F$
  $27FE2FBEE373F79B198FC7F1A3FB5594FE97B2D6EE6ADA84E6D$
  $726A709370D86FEEFAFD20300BFBD72B4F162A26C70F9F1927AB6$
  $6111B1FD5E7C1197AAEDD81776BFE079449A11A1AC849$ (1650 *bits*)

Using the method of [21] (or [33]) to construct the $\varphi(mk)$ dimensional lattice $L = I^{(1)} = \{h(t) | h(s) \equiv 0 \pmod{r}\}$, we find a approximative "short vector" of the polynomial form $h_1(t) = t^5 + c_1 t^2 + 1$ with $c_1 = 3^{49}$ for $E_1(\mathbb{F}_{3^{5\times 97}})$; and, $h_2(t) = t^{17} + c_2 t^8 + 1$ with $c_2 = 3^{34}$ for $E_2(\mathbb{F}_{3^{17\times 67}})$. Form the theory of pairing lattice, it follows that

$$f_{s_1,h_1,P_1} = f_{c_1,P_1} \frac{l_{[q_1^{m_1}]P_1,[-q_1^{m_1}-1]P_1} l_{-P_1,P_1}}{v_{-P_1}},$$

$$f_{s_2,h_2,P_2} = f_{c_2,P_2} \frac{l_{[q_2^{m_2}]P_2,[-q_2^{m_2}-1]P_2} l_{-P_2,P_2}}{v_{-P_2}},$$

where $P_i \in E_i(\mathbb{F}_{q_i^{m_i}})[r_i]$ and $s_i \equiv q_i \pmod{r_i}$ for $i = 0, 1$. We note that the calculations of $[q_i^{m_i}]P_i$ and $[-q_i^{m_i} - 1]P_i$ are very fast by using Frobenius map

$\pi_{q_i^{m_i}}$ and the trace equation. Thus, assuming that m_i multiprocessors $(i = 1, 2)$ perform in parallel, the Miller's loop length of $\bar{\eta}_{s_i,h_i}$ for $E_i[r_i]$ can reach an small value $\log_3(c_i)$, although which is still a little worse than the theoretical minimal length $\log_3(r)/\varphi(mk)$, when using 3-basis in Miller algorithm. Further, with Estibals's compact hardware implementation of these fields arithmetic, we believe that our pairing $\bar{\eta}_{s,h}$ would be implemented at much higher speed in parallel way.

6 Conclusion

We have shown that pairing-friendly curves over extension fields could be more suitable for the pairing implementation not relying on a fast field arithmetic of certain extension field. When assuming there exists a pairing-friendly curve defined over an extension field, we have proposed new pairings and pairing lattices on this curve making better use of the multi-pairing technique to obtain a fast implementation. By the theoretical analysis in an ideal model, the performance of the optimal ones of our pairings could offer a speed up of between 30% and 43% with the fixed argument optimization, or by up to 47% with our new storage skill, compared to the performance of the optimal ate pairing and the optimal twisted ate pairing, when m is greater than 6. In addition, we have extended the similar method to supersingular curves over composite extension fields to construct more efficient pairings in parallel implementation. To sum up, our work has presented further important evidence of the advantage of pairing-friendly curves over extension fields.

In future, there are needs for careful study of the generation of pairing-friendly curves over suitably chosen extension fields, and further study of the parallel implementation of $\bar{\eta}_{s,h}$ on Estibals's supersingular curves $E_1(\mathbb{F}_{3^{5\times 97}})$ and $E_2(\mathbb{F}_{3^{17\times 67}})$.

Acknowledgments. This work is supported by the National 973 Program of China under Grant 2011CB302400, the National Natural Science Foundation of China under Grant 60970152 and 60970153, the Grand Project of Institute of Software under Grant YOCX285056. We are heartily grateful to the anonymous reviewers for their helpful and insightful comments and suggestions.

References

1. Bailey, D.V., Paar, C.: Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 472–485. Springer, Heidelberg (1998)
2. Bailey, D.V., Paar, C.: Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. Journal of Cryptology 14(3), 153–176 (2001)
3. Bajard, J.C., Imbert, L., Negre, C., Plantard, T.: Efficient multiplication in $GF(p^k)$ for elliptic curve cryptography. In: Proceedings of the 16th IEEE Symposium on Computer Arithmetic 2003, pp. 181–187. IEEE (2003)

4. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–369. Springer, Heidelberg (2002)
5. Barreto, P.S.L.M., Galbraith, S.D., hÉigeartaigh, C.Ó., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007)
6. Benger, N., Charlemagne, M., Freeman, D.M.: On the Security of Pairing-Friendly Abelian Varieties over Non-prime Fields. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 52–65. Springer, Heidelberg (2009)
7. Costello, C., Lange, T., Naehrig, M.: Faster Pairing Computations on Curves with High-Degree Twists. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 224–242. Springer, Heidelberg (2010)
8. Costello, C., Stebila, D.: Fixed argument pairings. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 92–108. Springer, Heidelberg (2010)
9. Diem, C.: The GHS attack in odd characteristic. J. Ramanujan Math. Soc. 18(1), 1–32 (2003)
10. Diem, C.: On the discrete logarithm problem in elliptic curves. Compositio Mathematica 147(01), 75–104 (2011)
11. Estibals, N.: Compact Hardware for Computing the Tate Pairing over 128-Bit-Security Supersingular Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 397–416. Springer, Heidelberg (2010)
12. Freeman, D., Scott, M., Teske, E.: A Taxonomy of Pairing-Friendly Elliptic Curves. Journal of Cryptology 23(2), 224–280 (2010)
13. Frey, G., Gangl, H.: How to disguise an elliptic curve (Weil descent). In: Talk at ECC 1998, vol. 98 (1998)
14. Galbraith, S.D., Hess, F., Smart, N.P.: Extending the GHS Weil Descent Attack. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 29–44. Springer, Heidelberg (2002)
15. Galbraith, S.D., Smart, N.P.: A Cryptographic Application of Weil Descent. In: Walker, M. (ed.) IMA - Crypto & Coding 1999. LNCS, vol. 1746, pp. 191–200. Springer, Heidelberg (1999)
16. Gaudry, P.: Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. Journal of Symbolic Computation 44(12), 1690–1702 (2009)
17. Gaudry, P., Hess, F., Smart, N.P.: Constructive and destructive facets of Weil descent on elliptic curves. Journal of Cryptology 15(1), 19–46 (2002)
18. Granger, R.: On the Static Diffie-Hellman Problem on Elliptic Curves over Extension Fields. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 283–302. Springer, Heidelberg (2010)
19. Granger, R., Smart, N.P.: On computing products of pairings. Cryptology ePrint Archive Report 2006/172 (2006), Preprint available at http://eprint.iacr.org/2006/172
20. Hess, F.: Generalising the GHS attack on the elliptic curve discrete logarithm problem. LMS Journal of Computation and Mathematics 7(1), 167–192 (2004)
21. Hess, F.: Pairing Lattices. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 18–38. Springer, Heidelberg (2008)
22. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. IEEE Trans. on Information Theory 52(10), 4595–4602 (2006)

23. Hitt, L.: On the Minimal Embedding Field. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 294–301. Springer, Heidelberg (2007)
24. Joux, A., Vitse, V.: Elliptic Curve Discrete Logarithm Problem over Small Degree Extension Fields. Application to the static Diffie-Hellman problem on $E(\mathbb{F}_{q^5})$. Cryptology ePrint Archive, Report 2010/157 (2010), Preprint available at http://eprint.iacr.org/2010/157
25. Koblitz, N., Menezes, A.: Pairing-Based Cryptography at High Security Levels. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)
26. Lee, E., Lee, H.S., Park, C.M.: Efficient and generalized pairing computation on abelian varieties. IEEE Trans. on Information Theory 55(4), 1793–1803 (2009)
27. Lim, C.H., Hwang, H.S.: Fast Implementation of Elliptic Curve Arithmetic in GF(p^n). In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 405–421. Springer, Heidelberg (2000)
28. Menezes, A., Teske, E.: Cryptographic implications of Hess' generalized GHS attack. Applicable Algebra in Engineering, Communication and Computing 16(6), 439–460 (2006)
29. Miller, V.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17(4), 235–261 (2004)
30. Sakemi, Y., Takeuchi, S., Nogami, Y., Morikawa, Y.: Accelerating Twisted Ate Pairing with Frobenius Map, Small Scalar Multiplication, and Multi-pairing. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 47–64. Springer, Heidelberg (2010)
31. Scott, M.: Computing the Tate Pairing. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
32. Scott, M.: On the Efficient Implementation of Pairing-Based Protocols. In: Chen, L. (ed.) Cryptography and Coding 2011. LNCS, vol. 7089, pp. 296–308. Springer, Heidelberg (2011)
33. Vercauteren, F.: Optimal Pairings. IEEE Trans. on Information Theory 56(1), 455–461 (2010)
34. Zhang, X., Lin, D.: Efficient Pairing Computation on Ordinary Elliptic Curves of Embedding Degree 1 and 2. In: Chen, L. (ed.) IMACC 2011. LNCS, vol. 7089, pp. 309–326. Springer, Heidelberg (2011)

A Proof of Theorem 5

We do the similar reduction as Theorem 3 for the modified Eta pairing to obtain that

$$\eta(P,Q) = f_{q^m,P}\big(\psi(Q)\big)^{(q^{mk}-1)/r} = \left(\prod_{i=0}^{m-1} f_{q,[q^i]P}\big(\psi(Q)\big)^{q^{m-i-1}}\right)^{(q^{mk}-1)/r}.$$

Since the multiplication by q^i on the supersingular curve is inseparable, it follows that $[q^i]^*\mathrm{div}(f_{q,[q^i]P}) = \mathrm{div}(f_{q,P}^{q^{2i}})$ and then $f_{q,[q^i]P}(\psi(Q)) = f_{q,P}(\psi([q^{-i}]Q))^{q^{2i}}$. Thus we have

$$\eta(P,Q) = \left(\prod_{i=0}^{m-1} f_{q,P}\Big(\psi([q^{-i}]Q)\Big)^{q^i}\right)^{q^{m-1}(q^{mk}-1)/r}.$$

Since $\gcd(q, r) = 1$, we can omit the power q^{m-1} to obtain the new pairing

$$\bar{\eta}(P, Q) = \left(\prod_{i=0}^{m-1} f_{q,P}\Big(\psi([q^{-i}]Q)\Big)^{q^i} \right)^{(q^{mk}-1)/r}.$$

Then, as with the proof of Theorem 4, we can construct $\bar{\eta}_{s,h}$ as

$$\bar{\eta}_{s,h}(P, Q) = \left(\prod_{i=0}^{m-1} f_{s,h,P}\Big(\psi([q^{-i}]Q)\Big)^{q^i} \right)^{(q^{mk}-1)/r}.$$

and demonstrate it defines a pairing using Theorem 1 similarly (omitted here).

Factor-4 and 6 (De)Compression for Values of Pairings Using Trace Maps

Tomoko Yonemura, Taichi Isogai, Hirofumi Muratani, and Yoshikazu Hanatani

Toshiba Corporation,
1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582, Japan
tomoko.yonemura@toshiba.co.jp

Abstract. The security of pairing-based cryptosystems relies on the hardness of the discrete logarithm problems in elliptic curves and in finite fields related to the curves, namely, their embedding fields. Public keys and ciphertexts in the pairing-based cryptosystems are composed of points on the curves or values of pairings. Although the values of the pairings belong to the embedding fields, the representation of the field is inefficient in size because the size of the embedding fields is usually larger than the size of the elliptic curves. We show factor-4 and 6 compression and decompression for the values of the pairings with the supersingular elliptic curves of embedding degrees 4 and 6, respectively. For compression, we use the fact that the values of the pairings belong to algebraic tori that are multiplicative subgroups of the embedding fields. The algebraic tori can be expressed by the affine representation or the trace representation. Although the affine representation allows decompression maps, decompression maps for the trace representation has not been known. In this paper, we propose a trace representation with decompression maps for the characteristics 2 and 3. We first construct efficient decompression maps for trace maps by adding extra information to the trace representation. Our decompressible trace representation with additional information is as efficient as the affine representation is in terms of the costs of compression, decompression and exponentiation, and the size.

Keywords: public-key cryptosystems, the discrete logarithm problem, algebraic tori, compression, decompression.

1 Introduction

Practical public-key cryptography is fundamental technology in the field of network security. Current security standards recommend the use of 2048-bit or larger RSA keys [2] and history in these decades suggests that this figure may increase with advances in computational power. Such key sizes are problematic for devices with limited storage, computational power or network bandwidth. One approach to overcome these limitations is a safe key compression [19,6,14,16,5,18,12,13], but these compression techniques are unsuited to RSA keys. Therefore, we focus on cryptosystems based on the discrete logarithm

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 19–34, 2013.

problem in a prime-order group. To compress the public-key size is to represent the prime-order group with fewer bits than the size of the embedding field. For instance, the recommended size of the finite field is 2048 bits, and the corresponding size of the prime-order group is 224 bits [2], because the discrete logarithm problem in the finite field is easier than in the general group, namely, the elliptic curve.

The index calculus is a relatively efficient algorithm to solve the discrete logarithm problem in finite fields. The time complexity of the index calculus is subexponential $L_q[1/3, c] = \exp((c+o(1))(\log q)^{1/3}(\log\log q)^{2/3})$ for the finite field $\mathbb{F}_q$, and does not depend on the characteristic or the extension degree [7,1,10,11] except the constant c. On the other hand, there are only exponential algorithms for solving the discrete logarithm problem in the elliptic curves.

Pairings map a pair of elliptic curve points to an element of the multiplicative group of a finite field, namely, the curve's embedding field. Since pairings are bilinear, the discrete logarithm problem in elliptic curves is also solved in their embedding fields. The bilinearity is used to develop efficient cryptographic schemes [17,9,3]. In pairing-based cryptosystems, we deal with both rational points of elliptic curves and values of pairings. Although the values of the pairings belong to the embedding fields, the representation of the field is inefficient in the size. We show factor-4 and 6 compression and decompression for the values of the pairings with the supersingular elliptic curves of embedding degrees 4 and 6, respectively. For compression, we use the fact that the values of the pairings belong to also algebraic tori that are the multiplicative subgroups of the embedding fields.

Related Work. Table 1 presents existing compression methods. There are two kinds of compression methods: the affine representation and the trace representation. Algebraic tori ($\mathbb{T}_2$, $\mathbb{T}_6$, LUC, XTR) and their subgroups (Karabina, Shirase) have compact expressions.

In the affine representation, elements of algebraic tori are embedded in extension fields and identified by an element / elements from subfields. Elements

Table 1. compression methods:ECC, FFC and ATC mean the elliptic curve cryptosystems, the finite field cryptosystems and the algebraic torus cryptosystems, respectively

system	ECC	FFC	ATC							
class	-	-	the affine representation				the trace representation			
name	-	-	$\mathbb{T}_2$	$\mathbb{T}_6$	Karabina	Karabina	LUC	XTR	Karabina	Shirase
factor	-	1	2	3	4	6	2	3	4	6
public-key size (bit)	160	1024	512	341	256	170	512	341	256	170
	224	2048	1024	683	512	341	1024	683	512	341
	256	3072	1536	1024	768	512	1536	1024	768	512
reference	-	-	[16]	[16]	[13]	[13]	[19]	[14]	[12]	[18]
comp.	-	-	available				available			
decomp.	-	-	available				no			

of subgroups of algebraic tori are identified by a tuple of an element from subfields and additional information, namely, 1 bit for factor 4 or 1 trit (ternary digit) for factor 6. Each affine representation has efficient inverse map allowing multiplication and exponentiation in the embedding fields.

In the trace representation, elements of algebraic tori or these subgroups are identified by a trace value. Because conjugates are mapped to a same trace value, no inverse map exists. Therefore, multiplication could not be defined in the trace representation. On the other hand, exponentiation can be calculated without decompression or without distinction among conjugates. Although Karabina discusses "decompression" without distinction among conjugates, no efficient "decompression" maps are presented [12]. Most cryptosystems use not only exponentiation but also multiplication. The existing trace representation is not useful because of lack of multiplication.

Our Contributions. We propose factor-4 and 6 decompressible trace representation with additional information for characteristics 2 and 3, respectively. We construct decompression maps for the trace representation by adding extra information. Our decompression maps are efficient. Since our representation permits decompression, we are able to introduce multiplication in the trace representation for the first time. All cryptographic protocols based on group law and the discrete logarithm problem can be implemented on this representation. Why do we focus not on the affine representation, but on the trace representation? One of the reasons is the trace representation seems to be suited to improving the compression factor.

There are two steps for the construction of our representation: Firstly, we find easily solvable equations whose coefficients are written by the trace value to obtain the elements of the algebraic tori in the embedding fields as solutions. Secondly, we distinguish these solutions by additional information, namely, 2 bits for factor 4 or 1 bit and 1 trit for factor 6.

In order to improve the compression factor, it is required that the tuple of a trace value and additional information have to achieve a better compression factor than Bosma's conjecture. Bosma's conjecture on generalization of XTR mentioned the tuple of a trace value and other fundamental symmetric polynomials to improve the compression factor [4]. However, the additional information is much smaller than the fundamental symmetric polynomials.

Structure of This Paper. In section 2 and 3, we present the necessary preliminaries and literature review respectively. In section 4, we propose decompression maps for the trace representation with additional information. In section 5, we compare the efficiency of our representation with existing affine representation.

2 Preliminaries and Notation

Let p be a prime, and n, m and d be positive integers. Let $\mathbb{F}_{p^m}$ be a finite field of order p^m. I_d, M_d, and S_d are costs of inversion, multiplication, and square in the

field $\mathbb{F}_{(p^m)^d}$. We ignore costs of Frobenius maps and addition in $\mathbb{F}_{(p^m)^d}$ that are small compared with the above costs. Maps $Tr_{\mathbb{F}_{(p^m)^n}/\mathbb{F}_{(p^m)^d}}$ and $N_{\mathbb{F}_{(p^m)^n}/\mathbb{F}_{(p^m)^d}}$ denote a trace map and a norm map from $\mathbb{F}_{(p^m)^n}$ to $\mathbb{F}_{(p^m)^d}$, respectively, where, d divides n. Maps $Tr_{n/d}$ and $N_{n/d}$ are short for the above maps.

Definition 1. *An algebraic torus $\mathbb{T}_n$ over $\mathbb{F}_{p^m}$ is defined by*

$$\mathbb{T}_n(\mathbb{F}_{p^m}) = \bigcap_{\mathbb{F}_{p^m} \subset F \subsetneq \mathbb{F}_{(p^m)^n}} \mathrm{Ker}\left[N_{\mathbb{F}_{(p^m)^n}/F}\right] . \tag{1}$$

Definition 2. *Let μ be the Möbius function. The n-th cyclotomic polynomial $\Phi_n(x)$ is defined by $\Phi_n(x) = \prod_{d|n}(x^d - 1)^{\mu(n/d)}$.*

Theorem 1. *(a) $\#\mathbb{T}_n(\mathbb{F}_{p^m}) = \Phi_n(p^m)$.*
(b) If $h \in \mathbb{T}_n(\mathbb{F}_{p^m})$ has a prime order not dividing n, then $h \notin \mathbb{F}_{(p^m)^d}$ for any $d|n$ with $d < n$.

Proof. (a) Note that F can be $\mathbb{F}_{(p^m)^d}$ for any $d|n$ with $d < n$. See also [16].
(b) Let prime r be the order of h. Since $r \not| n$, $X^n - 1$ has no repeated roots in the algebraic closure of $\mathbb{F}_r$. See also [4]. □

In the case of $m > 1$, $\#\mathbb{T}_{mn}(\mathbb{F}_p) = \Phi_{mn}(p)$. If $h \in \mathbb{T}_{mn}(\mathbb{F}_p)$ has a prime order not dividing mn, then $h \notin \mathbb{F}_{p^d}$ for any $d|mn$ with $d < mn$. On the other hand, the order of the finite field $\mathbb{F}_{(p^m)^n}$ is factored as in eq. (2) by using cyclotomic polynomials.

$$(p^{mn} - 1) = \prod_{d|mn} \Phi_d(x) \tag{2}$$

The secure subgroup of the multiplicative group $\mathbb{F}^{\times}_{(p^m)^n}$ is not covered in proper subfield $\mathbb{F}_{p^d}$. In other words, it is a subgroup of $\mathbb{T}_{mn}(\mathbb{F}_p)$, and is not a subgroup of $\mathbb{T}_d(\mathbb{F}_p)$. Therefore, public-key cryptosystems defined on prime-order subgroup not dividing mn of the algebraic tori $\mathbb{T}_{mn}(\mathbb{F}_p)$ have the same security level as the multiplicative group $\mathbb{F}^{\times}_{(p^m)^n}$.

Let E be an elliptic curve defined over $\mathbb{F}_{p^m}$, and let r be a positive integer such that $r|\#E(\mathbb{F}_{p^m})$. A subgroup of $E(\mathbb{F}_{p^m})$ with order r has the embedding degree k, and k is the smallest integer such that $r|\{(p^m)^k - 1\}$. The Tate pairing is a function

$$\langle \cdot, \cdot \rangle_r : E(\mathbb{F}_{p^m})[r] \times E(\mathbb{F}_{(p^m)^k})/rE(\mathbb{F}_{(p^m)^k}) \to \mathbb{F}^{\times}_{(p^m)^k}/\{\mathbb{F}^{\times}_{(p^m)^k}\}^r.$$

A value of the Tate pairing is an equivalence class in $\mathbb{F}^{\times}_{(p^m)^k}/\{\mathbb{F}^{\times}_{(p^m)^k}\}^r$. For practical purposes, we obtain the reduced Tate pairing $e(P,Q) = \langle P, Q\rangle_r^{\{(p^m)^k-1\}/r} \in \mu_r \subset \mathbb{F}^{\times}_{(p^m)^k}$ as a unique representative of this class, where μ_r is a set of r-th roots of unity. There is an important fact $\mu_r \subset \mathbb{T}_k(\mathbb{F}_{p^m}) \subset \mathbb{F}^{\times}_{(p^m)^k}$. By definition of the embedding degree, $r \not| \{(p^m)^d - 1\}$ with $d < k$. In other words, μ_r is a subgroup of $\mathbb{T}_k(\mathbb{F}_{p^m})$, and is not a subgroup of $\mathbb{T}_d(\mathbb{F}_{p^m})$.

The supersingular elliptic curves over $\mathbb{F}_{p^m}$ have the following order [15]. For embedding degree $k = 4$,

– $p = 2$ and $E_i : y^2 + y = x^3 + x + a_i$, where $a_1 = 0$ and $a_2 = 1$.
– $\#E_i(\mathbb{F}_{p^m}) = p^m \pm \sqrt{2p^m} + 1$, where m is odd.

For embedding degree $k = 6$,

– $p = 3$ and $E_i : y^2 = x^3 - x + a_i$, where $a_1 = 1$ and $a_2 = -1$.
– $\#E_i(\mathbb{F}_{p^m}) = p^m \pm \sqrt{3p^m} + 1$, where m is odd.

3 Literature Review

3.1 The Affine Representation

In this section, we recall the definition of $\mathbb{T}_2$. We use the $\mathbb{T}_2$ affine representation as the special case of the projective representation for the following construction of decompression for trace maps. Because operations are more efficient in the projective representation than the affine representation, operations are done in the projective representation. Maps between the affine representation and the projective representation are called a compression map and a decompression map.

$\mathbb{T}_2$. This is the factor-2 compression and decompression method by Rubin and Silverberg. An element of $\mathbb{T}_2(\mathbb{F}_{p^m})$ is identified by an element of $\mathbb{F}_{p^m}$. Let an element $\frac{a+b\sigma}{a+b\sigma^{p^m}}$ of $\mathbb{T}_2(\mathbb{F}_{p^m})$ be corresponding to (a, b). Where $a, b \in \mathbb{F}_{p^m}$ and $(a, b) \neq (0, 0)$, $\mathbb{F}_{(p^m)^2} = \mathbb{F}_{p^m}(\sigma)$, and $\sigma \in \mathbb{F}^{\times}_{(p^m)^2}$. This representation has a natural projective equivalence relation. The element corresponding to (a, b) is equivalent to the element corresponding to (ac, bc) for any $c \in \mathbb{F}^{\times}_{p^m}$. So, this representation can be called the projective representation. We obtain $(a/b, 1)$ as the representative point of (a, b) and it is the affine representation of $\mathbb{T}_2(\mathbb{F}_{p^m})\backslash\{1\}$.

The compression map (from the projective representation to the affine representation) $\mathcal{C}$ and the decompression map (from the affine representation to the projective representation) $\mathcal{D}$ are as follows:

$$\mathcal{C} : \mathbb{T}_2(\mathbb{F}_{p^m})\backslash\{1\} \to \mathbb{F}_{p^m} \quad \mathcal{D} : \mathbb{F}_{p^m} \to \mathbb{T}_2(\mathbb{F}_{p^m})\backslash\{1\}$$
$$\frac{a + b\sigma}{a + b\sigma^{p^m}} \mapsto a/b, \qquad a' \mapsto \frac{a' + \sigma}{a' + \sigma^{p^m}}.$$

3.2 The Trace Representation – Compression by Trace Maps

In this section, we explain Karabina and Shirase. We construct decompression maps for the compression in the next section. Note that exponentiation in the trace representation itself can be calculated, but multiplication is not done.

Karabina. This is the factor-4 compression method. Let $p = 2$ and m be odd. An element of groups $G_{\pm}$, $\#G_{\pm} = p^m \pm \sqrt{2p^m} + 1$, is identified by an element of $\mathbb{F}_{p^m}$ without distinction among conjugates. The compression map is as follows:

$$Tr_{4/1} : \mathbb{F}_{(p^m)^4} \to \mathbb{F}_{p^m}$$
$$g \mapsto g + g^{p^m} + g^{(p^m)^2} + g^{(p^m)^3}.$$

Since $\#G_+\#G_- = \Phi_4(p^m)$, The groups $G_\pm$ are subgroups of $\mathbb{T}_4(\mathbb{F}_{p^m})$. Such subgroups are related to supersingular elliptic curves of embedding degree 4. Karabina also proposed some exponentiation formulas. Although there is "decompression" without distinction between conjugates, he didn't give any efficient decompression maps.

Shirase. This is the factor-6 compression method. Let $p = 3$ and m be odd. An element of groups $G_\pm$, $\#G_\pm = p^m \pm \sqrt{3p^m} + 1$, is identified by an element of $\mathbb{F}_{p^m}$ without distinction among conjugates. The compression map is as follows:

$$\begin{aligned} Tr_{6/1} : \mathbb{F}_{(p^m)^6} &\to \mathbb{F}_{p^m} \\ g &\mapsto g + g^{p^m} + g^{(p^m)^2} + g^{(p^m)^3} + g^{(p^m)^4} + g^{(p^m)^5}. \end{aligned}$$

Since $\#G_+\#G_- = \Phi_6(p^m)$, The groups $G_\pm$ are subgroups of $\mathbb{T}_6(\mathbb{F}_{p^m})$. Such subgroups are related to supersingular elliptic curves of embedding degree 6.

4 Construction of Decompression for Trace Maps

We propose the decompressible trace representation with additional information. The trace representation is decompressed to the projective representation for factor 4 and 6, and then multiplication can be calculated. Therefore, all cryptographic protocols based on group law and the discrete logarithm problem can be implemented on this representation.

4.1 Factor 4

We show decompression from $(i, Tr_{4/1}(g)) \in \{0,1\}^2 \times \mathbb{F}_{p^m}$ to $g \in G_- \subset \mathbb{T}_4(\mathbb{F}_{p^m}) \subset \mathbb{F}_{(p^m)^4}$. Firstly, we find equations to obtain four possible solutions by $Tr_{4/1}(g)$. Secondly, we distinguish the conjugates by the additional information i. The finite field $\mathbb{F}_{(p^m)^4}$ is constructed as follows:

- primitive polynomial: $\Phi_5(x) = x^4 + x^3 + x^2 + x + 1$,
- basis: $\{x, x^{p^m}, x^{(p^m)^2}, x^{(p^m)^3}\} = \{x, x^2, x^4, x^3\}$.

We use $p^m \bmod 5 = 2$, $z = x + x^{p^m}$ and $y = x + x^{(p^m)^2}$. One can also use $p^m \bmod 5 = 3$.

Theorem 2. *Suppose $p = 2$, m is odd, $t = \sqrt{2p^m}$ and G_- is a group of order $p^m - t + 1$, then there exist the compression map $\mathcal{C}$ described by eq. (3) and the decompression map $\mathcal{D}$ described by eq. (4).*

$$\begin{aligned} \mathcal{C} : G_- \backslash \{1\} &\to \{0,1\}^2 \times \mathbb{F}_{p^m} \\ \frac{h}{h^{(p^m)^2}} &\mapsto (i, Tr_{4/1}(g)) \end{aligned} \tag{3}$$

$$\begin{aligned} \mathcal{D} : \{0,1\}^2 \times \mathbb{F}_{p^m} &\to G_- \backslash \{1\} \\ (i, Tr_{4/1}(g)) &\mapsto \frac{f+z}{f+z^{(p^m)^2}} \end{aligned} \tag{4}$$

Where, the projective representation of g is $\frac{h}{h^{(p^m)^2}}$, $h \in \mathbb{F}_{(p^m)^4}$, and the $\mathbb{T}_2$ affine representation of g is $\frac{f+z}{f+z^{(p^m)^2}}$, $f = \delta_1 y + \delta_2 y^{p^m} \in \mathbb{F}_{(p^m)^2}$ for some δ_1, $\delta_2 \in \mathbb{F}_{p^m}$. Let i be a tuple of the least bits in the vector representation of δ_1 and δ_2.

Proof. The decompression map $\mathcal{D}$ is calculated by solving eq. (5) in Lemma 2 and eq. (6) in Lemma 3. The following Lemma 1 is condition for the element in the subgroup of the algebraic torus, and leads to Lemma 2 and 3. Lemma 4 shows why i distinguishes the four solutions. □

Calculations of the compression map $\mathcal{C}$ and the decompression map $\mathcal{D}$ are shown in Algorithm 1 and 2.

Algorithm 1. Factor-4 compression $\mathcal{C}$

Input: the projective representation $\frac{h}{h^{(p^m)^2}} = \frac{h_0+h_1 z}{h_0+h_1 z^{(p^m)^2}}$ for g
Output: $(i, Tr_{4/1}(g))$

1: $f = \delta_1 y + \delta_2 y^{p^m} \leftarrow h_0/h_1$
2: $i_1 \leftarrow$ the least bit of δ_1 in the vector representation
3: $i_2 \leftarrow$ the least bit of δ_2 in the vector representation
4: $i \leftarrow (i_1, i_2)$
5: $Tr_{4/1}(g) \leftarrow \frac{\delta_1^2+\delta_2^2+\delta_1+\delta_2+1}{\delta_1^4+\delta_1^2\delta_2^2+\delta_2^4+\delta_1^3+\delta_2^3+\delta_1\delta_2+\delta_2^2+\delta_2+1}$

Algorithm 2. Factor-4 decompression $\mathcal{D}$

Input: $(i, Tr_{4/1}(g))$
Output: the $\mathbb{T}_2$ affine representation $\frac{f+z}{f+z^{(p^m)^2}}$, $f = \delta_1 y + \delta_2 y^{p^m}$ for g

1: solve $D^2 + D + 1 = \{Tr_{4/1}(g)\}^{p^m - t}$ for D and obtain a solution D with the least bit $i_1 + i_2$ in the vector representation
2: solve $\delta_2^2 + \delta_2 = D^2 + \{Tr_{4/1}(g)\}^{p^m - t} D^t$ for δ_2 and obtain δ_2 with the least bit i_2 in the vector representation
3: $\delta_1 \leftarrow \delta_2 + D$

Lemma 1. *Use the notation in Theorem 2. Let $g = \frac{f+z}{f+z^{(p^m)^2}} \in G_- \backslash \{1\}$. When $t \bmod 5 = 2$, δ_1 and δ_2 satisfy*

$$(\delta_1 + \delta_2)^{t+1} = \delta_1^t + \delta_2 + 1.$$

Proof. The condition $g \in G_- \backslash \{1\}$ leads to $g^{p^m+1} = g^t$. We substitute $g = \frac{f+z}{f+z^{(p^m)^2}}$ in the above equation, and then

$$\{\delta_1^{t+1} + \delta_1\delta_2^t + \delta_1^t + (\delta_1^2 + \delta_1\delta_2 + \delta_2^2 + \delta_1 + \delta_2^t)\}y \\ + \{\delta_2^{t+1} + \delta_1^t\delta_2 + \delta_2 + 1 + (\delta_1^2 + \delta_1\delta_2 + \delta_2^2 + \delta_1 + \delta_2^t)\}y^{p^m} = 0.$$

We obtain the desired equation from the sum of coefficients for y and y^{p^m}. □

Lemma 2. *Use the notation in Theorem 2. $D = \delta_1 + \delta_2 \in \mathbb{F}_{p^m}$ satisfies*

$$D^2 + D + 1 = \{Tr_{4/1}(g)\}^{p^m - t}. \tag{5}$$

Proof. Lemma 1 leads to

$$\begin{cases} \delta_1^2 + \delta_2 = (\delta_1^t + \delta_2^t)(\delta_1^2 + \delta_2^2 + \delta_1 + \delta_2 + 1), \\ \delta_1 + \delta_2^2 = (\delta_1^t + \delta_2^t + 1)(\delta_1^2 + \delta_2^2 + \delta_1 + \delta_2 + 1) + 1. \end{cases}$$

We substitute the above equations to the trace value

$$Tr_{4/1}(g) = \frac{\delta_1^2 + \delta_2^2 + \delta_1 + \delta_2 + 1}{\delta_1^4 + \delta_1^2\delta_2^2 + \delta_2^4 + \delta_1^3 + \delta_2^3 + \delta_1\delta_2 + \delta_2^2 + \delta_2 + 1},$$

and then we obtain

$$(\delta_1^2 + \delta_2^2 + \delta_1 + \delta_2 + 1)^{t+1} = Tr_{4/1}(g)^{-1}.$$

Where $t^2 = 2p^m$ and $\delta_1, \delta_2, Tr_{4/1}(g) \in \mathbb{F}_{p^m}$, we obtain

$$\delta_1^2 + \delta_2^2 + \delta_1 + \delta_2 + 1 = \{Tr_{4/1}(g)\}^{p^m - t}.$$

□

Note that the characteristic is 2, and square is calculated by the Frobenius map involving rotation of elements in the normal basis. We obtain two solutions D_0 and D_1 of eq. (5) immediately.

Lemma 3. *Use the notation in Theorem 2. The element δ_2 satisfies*

$$\delta_2^2 + \delta_2 = D^2 + \{Tr_{4/1}(g)\}^{p^m - t} D^t. \tag{6}$$

Proof. Lemma 1 leads to

$$\delta_1^2 + \delta_2 = (\delta_1^t + \delta_2^t)(\delta_1^2 + \delta_2^2 + \delta_1 + \delta_2 + 1).$$

We show how to transform the equation of Lemma 1 to the above equation later. The left-hand side of the above equation is $(D + \delta_2)^2 + \delta_2 = D^2 + \delta_2^2 + \delta_2$, and then we obtain eq.(6).

Where $t^2 = 2p^m$ and $\delta_1, \delta_2 \in \mathbb{F}_{p^m}$, the equation of Lemma 1 to the power of $(t-1)$ is

$$(\delta_1 + \delta_2) = (\delta_1^t + \delta_2 + 1)^{t-1},$$

and then we obtain

$$(\delta_1+\delta_2)(\delta_1^t+\delta_2+1)=\delta_1^2+\delta_2^t+1.$$

We add $\delta_1^t+\delta_2^t$ to the equation of Lemma 1 multiplied by $\delta_1+\delta_2+1$,

$$(\delta_1^t+\delta_2^t)(\delta_1^2+\delta_2^2+\delta_1+\delta_2+1)=(\delta_1+\delta_2)(\delta_1^t+\delta_2+1)+\delta_2^t+\delta_2+1.$$

We substitute $(\delta_1+\delta_2)(\delta_1^t+\delta_2+1)=\delta_1^2+\delta_2^t+1$ to the above equation, and then we obtain the first equation in this proof. □

We obtain two δ_2 by solving eq. (6) with fixed D. Therefore, we obtain four solutions for (δ_1,δ_2).

Lemma 4. *Use the notation in Theorem 2. The least bits in the vector representation of δ_1 and δ_2 identify $g\in G_-\backslash\{1\}$ from solutions of eq. (5) and (6).*

Proof. The element g changes by p^m power: $(\delta_1,\delta_2)\to(\delta_2+1,\delta_1)\to(\delta_1+1,\delta_2+1)\to(\delta_2+1,\delta_1)$. □

4.2 Factor 6

We show decompression from $(i,Tr(g))\in\{0,1\}\times\{0,1,2\}\times\mathbb{F}_{p^m}$ to $g\in G_-\subset\mathbb{T}_6(\mathbb{F}_{p^m})\subset\mathbb{F}_{(p^m)^6}$. Firstly, we find equations to obtain six possible solutions by $Tr_{6/1}(g)$. Secondly, we distinguish the conjugates by the additional information i. The finite field $\mathbb{F}_{(p^m)^6}$ is constructed as follows:

- primitive polynomial: $\Phi_7(x)=x^6+x^5+x^4+x^3+x^2+x+1$,
- basis: $\{x,x^{p^m},x^{(p^m)^2},x^{(p^m)^3},x^{(p^m)^4},x^{(p^m)^5}\}=\{x,x^5,x^4,x^6,x^2,x^3\}$.

We use $p^m \bmod 7=5$, $z=x+x^{(p^m)^2}+x^{(p^m)^4}$ and $y=x+x^{(p^m)^3}$. One can also use $p^m \bmod 7=3$.

Theorem 3. *Suppose $p=3$, m is odd, $t=\sqrt{3p^m}$ and G_- is a group of order p^m-t+1, then there exist the compression map $\mathcal{C}$ described by eq. (7) and the decompression map $\mathcal{D}$ described by eq. (8).*

$$\begin{aligned}\mathcal{C}: G_-\backslash\{1\}&\to\{0,1\}\times\{0,1,2\}\times\mathbb{F}_{p^m}\\ \frac{h}{h^{(p^m)^3}}&\mapsto(i,Tr_{6/1}(g))\end{aligned}\tag{7}$$

$$\begin{aligned}\mathcal{D}: \{0,1\}\times\{0,1,2\}\times\mathbb{F}_{p^m}&\to G_-\backslash\{1\}\\ (i,Tr_{6/1}(g))&\mapsto\frac{f+z}{f+z^{p^m}}\end{aligned}\tag{8}$$

Where, the projective representation of g is $\frac{h}{h^{(p^m)^3}}$, $h\in\mathbb{F}_{(p^m)^6}$, and the $\mathbb{T}_2$ affine representation of g is $\frac{f+z}{f+z^{p^m}}$, $f=\delta_1 y+\delta_2 y^{p^m}+\delta_3 y^{(p^m)^2}\in\mathbb{F}_{(p^m)^3}$ for some δ_1, δ_2, $\delta_3\in\mathbb{F}_{p^m}$. The bit $\{1,0\}$ is transformed from $\{1,2\}$ of a trit in A^{-1} for $A=\delta_1+\delta_2+\delta_3$. The trit is in β' calculated from δ_2.

Proof. The decompression map $\mathcal{D}$ is calculated by solving eq. (9) in Lemma 6 and eq. (10) in Lemma 7. The following Lemma 5 is condition for the element in the subgroup of the algebraic torus, and leads to Lemma 6, 7 and 8. Lemma 9 shows why i distinguishes the six solutions. □

Calculations of the compression map $\mathcal{C}$ and the decompression map $\mathcal{D}$ are shown in Algorithm 3 and 4.

Algorithm 3. Factor-6 compression $\mathcal{C}$

Input: the projective representation $\frac{h}{h^{(p^m)^2}} = \frac{h_0+h_1 z}{h_0+h_1 z^{p^m}}$ for g
Output: $(i, Tr_{6/1}(g))$

$f = \delta_1 y + \delta_2 y^{p^m} + \delta_3 y^{(p^m)^2} \leftarrow h_0/h_1$
$\alpha \leftarrow \delta_1 - 1$
$\beta \leftarrow \delta_2 - 1$
$\gamma \leftarrow \delta_3 - 1$
$A \leftarrow \alpha + \beta + \gamma (= \delta_1 + \delta_2 + \delta_3)$
$B \leftarrow \alpha\beta^2 + \beta\gamma^2 + \gamma\alpha^2$
$C \leftarrow \alpha\beta\gamma$
$i_1 \leftarrow a_i \bmod 2$ for a_i that is the smallest nonzero trit of A^{-1} in the vector rep.
$\beta' \leftarrow \frac{A^{t+3}+A^2+1}{A^{t+1}(A^2-1-A\beta)}$
$i_2 \leftarrow$ the least trit of β' in the vector representation
$i \leftarrow (i_1, i_2)$
$Tr_{6/1}(g) \leftarrow \frac{A}{B+C-A^3-A}$

Algorithm 4. Factor-6 decompression $\mathcal{D}$

Input: $(i, Tr_{6/1}(g))$
Output: the $\mathbb{T}_2$ affine representation $\frac{f+z}{f+z^{p^m}}$, $f = \delta_1 y + \delta_2 y^{p^m} + \delta_3 y^{(p^m)^2}$ for g

solve $A^{-2} = -[\{Tr_{6/1}(g)\}^{t-2} + 1]$ for A^{-1} and obtain solutions A_0^{-1}, A_1^{-1}
if $i_1 = 1$ **then**
    select A_1^{-1} with $a_i = 1$ for the least nonzero trit a_i in the vector representation
else if $i_1 = 0$ **then**
    select A_0^{-1} with $a_i = 2$ for the least nonzero trit a_i in the vector representation
end if
solve $\beta'^3 - \beta' - (\frac{A^2+1}{A^{t+3}} + 1) = 0$ and obtain a solution β' with the least trit i_2
$\beta \leftarrow -A\{(\frac{A^2+1}{A^{t+3}} + 1)\frac{1}{\beta'} + 1 - A^{-2}\}$
$t_b \leftarrow -(1 + A^{-2})A^{-t} - A^{-1}$
$t_c \leftarrow A^2 - At_b + A^{-1}t_b - t_b^2 - A^{-4}$
$\gamma \leftarrow \frac{(1-A^{-2})\beta^2 + t_b\beta - t_c}{-A^{-1}\beta^2 + (1-A^{-2})\beta}$

Lemma 5. *Use the notation in Theorem 3. Let* $g = \frac{f+z}{f+z^{p^m}}$. *Note* $z^{(p^m)^3} = z^{p^m}$. *When* $m \bmod 12 = 5$, δ_1, δ_2 *and* δ_3 *satisfy*

$$\delta_1^t(\delta_3-\delta_2)+\delta_3^t(\delta_1+\delta_2+\delta_3)+2\delta_3\delta_1+\delta_1\delta_2+\delta_1^2+\delta_2^2-\delta_3=2,$$
$$\delta_2^t(\delta_1-\delta_3)+\delta_1^t(\delta_1+\delta_2+\delta_3)+2\delta_1\delta_2+\delta_2\delta_3+\delta_2^2+\delta_3^2-\delta_1=2,$$
$$\delta_3^t(\delta_2-\delta_1)+\delta_2^t(\delta_1+\delta_2+\delta_3)+2\delta_2\delta_3+\delta_3\delta_1+\delta_3^2+\delta_1^2-\delta_2=2.$$

Proof. $g \in G_-\backslash\{1\}$ leads to $g^{p^m+1} = g^t$. We substitute $g = \frac{f+z}{f+z^{(p^m)^2}}$ in $g^{p^m+1} = g^t$, and then we obtain the desired equations from coefficients for y, y^{p^m} and $y^{(p^m)^2}$. □

Lemma 6. *Use the notation in Theorem 3. Let* $\alpha = \delta_1 - 1$, $\beta = \delta_2 - 1$, $\gamma = \delta_3 - 1$, $A = \alpha+\beta+\gamma$, $B = \alpha\beta^2+\beta\gamma^2+\gamma\alpha^2$ *and* $C = \alpha\beta\gamma$, *where,* $\alpha, \beta, \gamma, A, B, C \in \mathbb{F}_{p^m}$. *A satisfies*

$$A^{-2} = -[\{Tr_{6/1}(g)\}^{t-2} + 1]. \tag{9}$$

Where $B = ((-1-A^2)/A^t) - A$ *and* $C = A^3 - B + (AB - B^2 - 1)/A^3$.

Proof. We substitute the equations of Lemma 5 to the trace value

$$Tr_{6/1}(g) = \frac{A}{B + C - A^3 - A},$$

and then we obtain eq. (9). □

Two solutions A_0 and A_1 of eq. (9) are calculated by square root.

Lemma 7. *Use the notation in Theorem 3. The element* β *satisfies*

$$\beta^3 - A\beta^2 - (A^2-1)\beta - C = 0. \tag{10}$$

Proof. $g \in \mathbb{T}_2(\mathbb{F}_{(p^m)^3})$ leads $\alpha^2+\beta^2+\gamma^2 = 1$. Eq. (10) is obtained from the above equation, $B = ((-1-A^2)/A^t) - A$ and $C = A^3 - B + (AB - B^2 - 1)/A^3$. □

We obtain three β by solving eq. (10).

Lemma 8. *Use the notation in Theorem 3. The element* γ *satisfies*

$$(-A\beta^2 + (A^2-1)\beta)\gamma + (-A^2+1)\beta^2 - B\beta + AC = 0. \tag{11}$$

Proof. Eq. (11) is obtained from $\alpha^2 + \beta^2 + \gamma^2 = 1$.

We obtain γ by solving eq. (11). Therefore we obtain six solutions for $(\delta_1, \delta_2, \delta_3)$.

Lemma 9. *Use the notation in Theorem 3. The additional information* $i = (i_1, i_2)$ *identify* $g \in G_-\backslash\{1\}$ *from solutions of eq. (9) and eq. (10). Let* $i_1 = a_i \bmod 2$ *for the least nonzero trit* a_i *of* A^{-1}, *and* i_2 *be the least trit of* $\beta' = \frac{A^{t+3}+A^2+1}{A^{t+1}(A^2-1-A\beta)}$ *in the vector representation.*

Proof. A trit of A_0^{-1} and a trit of A_1^{-1} in the same place are different unless the trit is zero because of $A_1^{-1} = -A_0^{-1}$. Solutions of the degree-3 equation are $\{\beta'_0, \beta'_0+1, \beta'_0-1\}$, and then trits of these solutions are different in all places. □

5 Performance

In this section, we compare costs of compression, decompression and exponentiation in our representation with existing schemes using the affine representation. We summarize our findings first and then present the detailed calculations.

Table 2 shows that compression and decompression costs are comparable. Note that the cost of solving the equation $X^p \pm X + C = 0$ is negligible, where $X, C \in \mathbb{F}_{p^m}$, C is constant. Exponentiation costs are the same, because we can use the projective representation and store precomputed information in the $\mathbb{T}_2$ affine representation. In the future, there is hope to improve upon naive exponentiation in the trace representation by using precomputation. The size of the additional information is comparable. We consider the computations of our representation with the compression factor of 4 and 6 in detail.

Factor 4. The compression map $\mathcal{C}$ described by eq. (3) costs $I_2 + I_1 + M_2 + 4M_1$ to calculate f for i and $Tr_{4/1}$. Alternatively, we can also calculate $Tr_{4/1}$ using h rather than f. In this case, $\mathcal{C}$ costs $I_2 + M_4 + S_4 + 4M_2 + 6M_1 \sim I_1 + 42M_1$. The decompression map $\mathcal{D}$ described by eq. (4) costs $I_1 + 2M_1 + S_1$ to calculate the coefficient of eq. (5) and eq. (6).

Because the image of the decompression map is in the projective representation in both cases of the affine representation and our representation, operations are calculated similarly. There is an exponentiation formula for the trace representation [12]. Its cost is estimated to be $(4M_1 + 1S_1)\log_2 r$, which is efficient compared with cost of simple square and multiplying $(M_4 + S_4)\log_2 r$. However, this is inefficient compared with cost of width-w NAF in the projective representation.

Factor 6. The compression map $\mathcal{C}$ described by eq. (7) costs $I_3 + I_1 + M_3 + 18M_1 + 2S_1$ to calculate f and A^{-1} for i and also to calculate $Tr_{6/1}$. Where, A^{-1}, $\{A^{t+1}(A^2 - 1 - A\beta)\}^{-1}$ and $(B + C - A^3 - A)^{-1}$ can be calculated by one inversion of the product $A \cdot \{A^{t+1}(A^2 - 1 - A\beta)\} \cdot (B + C - A^3 - A)$ in Algorithm 3. The decompression map $\mathcal{D}$ described by eq. (8) costs $4I_1 + SqRt + 10M_1 + 5S_1$ to perform the following calculations: to calculate the coefficient of eq. (9) and

Table 2. costs: let $M_6 = 18M_1$, $M_4 = 9M_1$, $M_3 = 6M_1$, $M_2 = 3M_1$ (by Karatsuba's method), $S_6 = M_6$, $S_1 = M_1$ (for simplicity), $I_3 = I_1 + 3M_3$ and $I_2 = I_1 + 2M_2$ (by Itoh-Tsujii's method [8]). $SqRt = \{\log_2 \frac{m-1}{2} + HW(\frac{m-1}{2})\}M_1 + S_1$ [12].

class	the affine representation		the trace representation	
name	Karabina	Karabina	this work	this work
factor	4	6	4	6
comp.	$I_1 + 9M_1$	$I_1 + 24M_1$	$2I_1 + 16M_1$	$2I_1 + 44M_1$
decomp.	$3M_1$	$I_1 + 9M_1$	$I_1 + 3M_1$	$4I_1 + SqRt + 15M_1$
exp.	$\frac{6}{w+1}\log_2 rM_1$	$\frac{24}{2w+1}\log_3 rM_1$	$\frac{6}{w+1}\log_2 rM_1$	$\frac{24}{2w+1}\log_3 rM_1$
added info.	1 bit	1 trit	2 bits	1 bit and 1 trit

the square root, to solve degree-3 equation, to transform the solution and to calculate γ. We explain solving degree-3 equation in detail. Let $\beta = -A\{(\frac{A^2+1}{A^{t+3}} + 1)\frac{1}{\beta'} + \frac{A^2-1}{A^2}\}$, then eq. (10) is written $\beta'^3 - \beta' - (\frac{A^2+1}{A^{t+3}} + 1) = 0$ by using β'. Note that the characteristic is 3, cubing is calculated by the Frobenius map involving rotation of elements in the normal basis. We obtain three solutions β' of the above equation immediately. One calculates the coefficient of the above equation and the transformation from β' for β.

The cost of an exponentiation formula for the trace representation [12] is estimated to be $(23M_1 + S_1)\log_3 r$, which is efficient compared with cost of simple cubing and multiplying $(2M_6 + C_6)\log_3 r$. However, this is inefficient compared with cost of width-w radix-3 NAF in the projective representation.

6 Conclusion

In this paper, we proposed the factor-4 and 6 decompressible trace representation with additional information for the characteristics 2 and 3, respectively. This representation has an efficient decompression map for the trace representation distinguishing conjugates by using the additional information. Since this representation permits decompression, we succeed in introducing multiplication in the trace representation for the first time. Practically, this representation is not worse than the affine representation. Although compression and decompression incur some extra field inversions in comparison with the affine representation, this fact is not a serious disadvantage of the proposed representation because the costs of compression and decompression is much smaller than the costs of encryption and decryption. It is clear that the cost of inversion in the base field is much smaller than the cost of exponentiation in the embedding field. In future work, we intend to improve the compression factor and reduce costs for the exponentiation, the compression and the decompression.

References

1. Adleman, L.M.: The Function Field Sieve. In: Huang, M.-D.A., Adleman, L.M. (eds.) ANTS 1994. LNCS, vol. 877, pp. 108–121. Springer, Heidelberg (1994)
2. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for Key Management - Part 1: Genaral (Revised). Special Publication 800/57, NIST (2007)
3. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–220. Springer, Heidelberg (2001)
4. Bosma, W., Hutton, J., Verheul, E.R.: Looking beyond XTR. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 46–63. Springer, Heidelberg (2002)
5. Giuliani, K.J., Gong, G.: Efficient Key Agreement and Signature Schemes Using Compact Representations in GF(p^{10}). In: ISIT 2004, p. 13. IEEE (2004)
6. Gong, G., Harn, L.: Public-key Cryptosystems besed on Cubic Finite Field Extensions. IEEE Trans. Inform. Theory 45, 2601–2605 (1999)
7. Gordon, D.: Discrete Logarithms in GF(p) Using the Number Field Sieve. SIAM J. on Discrete Math. 6, 124–138 (1993)

8. Itoh, T., Tsujii, S.: A Fast Algorithm for Computing Multiplicative Inverses in GF(2^m) Using Normal Bases. Information and Computation 78(3), 171–177 (1988)
9. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–393. Springer, Heidelberg (2000)
10. 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)
11. Joux, A., Lercier, R., Smart, N.P., 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)
12. Karabina, K.: Factor-4 and 6 Compression of Cyclotomic Subgroups. J. of Mathematical Cryptology 4(1), 1–42 (2010)
13. Karabina, K.: Torus-based Compression by Factor 4 and 6. Cryptology ePrint Archive, Report 2010/525 (2010)
14. Lenstra, A.K., Verheul, E.R.: The XTR Public Key System. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 1–19. Springer, Heidelberg (2000)
15. Miyaji, A., Nakabayashi, M., Takano, S.: New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. IEICE Trans. E84-A(5), 1234–1243 (2001)
16. Rubin, K., Silverberg, A.: Torus-Based Cryptography. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 349–365. Springer, Heidelberg (2003)
17. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptsystems based on Pairing. In: SCIS 2000 (2000)
18. Shirase, M., Han, D., Hibino, Y., Kim, H., Takagi, T.: A More Compact Representation of XTR Cryptosystem. IEICE Trans. E91-A(10), 2843–2850 (2008)
19. Smith, P., Skinner, C.: A Public-key Cryptosystem and a Digital Signature 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)

A The Affine Representation

Compression and decompression costs of the affine representation [13] are calculated as follows.

Factor 4. The point is that the condition for the element in the subgroup of the algebraic torus $\mathbb{T}_n(\mathbb{F}_{p^m})$ is solved and $n/2$ solutions are distinguished by using additional information in the $\mathbb{T}_2$ affine representation. The order of the subgroups is $\#G_{\pm} = p^m \pm t + 1, t = \sqrt{2p^m}$, where, $p = 2$, m is odd.

The compression map $\mathcal{C}$ is described by eq. (12) and the decompression map $\mathcal{D}$ is described by eq. (13). Where, $G \in \{G_-, G_+\}$.

$$\begin{aligned} \mathcal{C} : G\backslash\{1\} &\to \{0,1\} \times \mathbb{F}_{p^m} \\ \frac{\alpha + \beta\sigma}{\alpha + \beta(1+\sigma)} &\mapsto (i, b) \end{aligned} \tag{12}$$

$$\begin{aligned} \mathcal{D} : \{0,1\} \times \mathbb{F}_{p^m} &\to G\backslash\{1\} \\ (i, b) &\mapsto \frac{(a + bw) + \sigma}{(a + bw) + 1 + \sigma} \end{aligned} \tag{13}$$

Let $\mathbb{F}_{(p^m)^4} = \mathbb{F}_{(p^m)^2}(\sigma)$, $\mathbb{F}_{(p^m)^2} = \mathbb{F}_{p^m}(w)$, $\sigma \in \mathbb{F}_{(p^m)^4}$, $w \in \mathbb{F}_{(p^m)^2}$. We obtain $a, b \in \mathbb{F}_{p^m}$ by $\alpha/\beta = a + bw$ from $\alpha, \beta \in \mathbb{F}_{(p^m)^2}$. We calculate roots of polynomial $P_1(x, b) = x^t + x + u(b) \in \mathbb{F}_{p^m}[x]$ led by $g^{p^m \pm t+1} = 1$ in order to obtain a from b. If the above polynomial $P_1(x)$ has two distinct roots a_0 and a_1, then $a_1 = a_0 + 1$. Note that if the characteristic is 2, solving $P_1(x) = 0$ is easy. The additional information i is a bit of a in the vector representation.

The compression map $\mathcal{C}$ described by eq. (12) costs $I_2 + M_2$ to calculate $\alpha/\beta = a + bw, a, b \in \mathbb{F}_{p^m}$ from $\alpha, \beta \in \mathbb{F}_{(p^m)^2}$. The decompression map $\mathcal{D}$ described by eq. (13) costs $3M_1$ to calculate $u(b)$. Because $u(b)$ is

$$\begin{cases} u(b) = b^{t+1} + (u_0 + u_4)b^t + (u_0 + u_3 + 1)b + (u_0u_3 + u_2 + u_6) \text{ for } G_- \\ u(b) = b^{t+1} + (u_0 + u_4)b^t + (u_0 + u_3 + 1)b + (u_0u_3 + u_2 + u_6 + 1) \text{ for } G_+ \end{cases}$$

where $u_0, u_2, u_3, u_4, u_6 \in \mathbb{F}_{p^m}$ are precomputable parameters.

We recall an estimation of exponentiation cost [13]. We determine the width-w NAF representation of the power r, and then it contains on average $\log_2 r/(w+1)$ nonzero digits. After precomputing $g_i = g^i, i \in \{\pm 1, \pm 3, \pm 5, \cdots, \pm 2^{w-1} - 1\}$, it costs $\log_2 rS_4 + \log_2 r/(w+1)M_4$ to calculate g^r on average. If we calculate in the projective representation and store results of precomputation in the $\mathbb{T}_2$ affine representation, then we can replace M_4 with $2M_2 = 6M_1$.

Factor 6. Let $\#G_\pm = p^m \pm t + 1, t = \sqrt{3p^m}, p = 3$ and m be odd.

The compression map $\mathcal{C}$ is described by eq. (14) and the decompression map $\mathcal{D}$ is described by eq. (15).

$$\begin{aligned} \mathcal{C} : G_-\backslash\{1\} &\to \{0, 1, 2\} \times \mathbb{F}_{p^m} \\ \frac{\alpha + \beta\sigma}{\alpha - \beta\sigma} &\mapsto (i, c) \end{aligned} \tag{14}$$

$$\begin{aligned} \mathcal{D} : \{0, 1, 2\} \times \mathbb{F}_{p^m} &\to G_-\backslash\{1\} \\ (i, c) &\mapsto \frac{(a + bw + cw^2) + \sigma}{(a + bw + cw^2) - \sigma} \end{aligned} \tag{15}$$

Let $\mathbb{F}_{(p^m)^6} = \mathbb{F}_{(p^m)^3}(\sigma)$, $\mathbb{F}_{(p^m)^3} = \mathbb{F}_{p^m}(w)$, $\sigma \in \mathbb{F}_{(p^m)^6}$,$w \in \mathbb{F}_{(p^m)^3}$. We obtain $a, b, c \in \mathbb{F}_{p^m}$ by $\alpha/\beta = a + bw + cw^2$ from $\alpha, \beta \in \mathbb{F}_{(p^m)^3}$. We calculate roots of $P_6(x, c) = x^3 + 2c^{2t}x + C(c) \in \mathbb{F}_{p^m}[x]$ led by $g^{p^m - t+1} = 1$ in order to obtain a, b from c. Where, $C(c) = \frac{2(c^{3t+3} + c^{2t} + 1)}{c^3}$. The above polynomial $P_6(x)$ has roots $\{c^tR, c^t(R+1), c^t(R-1)\}$ as b^t. R is a solution of $x^3 - x + D(c) = 0$. Note that if the characteristic is 3, solving the above equation is easy. a^t is a root of degree-1 polynomial $P_2(x) \in \mathbb{F}_{p^m}[x]$. Therefore, three solutions are $\{(a, b, c), (a - b + c, b + c, c), (a + b + c, b - c, c)\}$. The additional information i is a trit of b in the vector representation. The place is the same for the least nonzero trit of c.

The compression map $\mathcal{C}$ described by eq. (14) costs $I_3 + M_3$ to calculate $\alpha/\beta = a + bw + cw^2, a, b, c \in \mathbb{F}_{p^m}$ from $\alpha, \beta \in \mathbb{F}_{(p^m)^3}$. The decompression map $\mathcal{D}$ described by eq. (15) costs $I_1 + 7M_1 + 2S_1$ to calculate

$$D(c) = \frac{2(c^{3t+3} + c^{2t} + 1)}{c^{3t+3}}$$

and to solve the following degree-1 polynomial

$$P_2(x) = x + 2b^{2t+3} + 2b^{2t}c^3 + b^t c^{t+3} + c^{2t+3} + 2c^t \in \mathbb{F}_{p^m}[x].$$

We recall an estimation of exponentiation cost [13]. We determine the width-w radix-3 NAF representation of the power r, and then it contains on average $2\log_3 r/(2w+1)$ nonzero digits. After precomputation, it costs $\log_3 rC_6 + 2\log_3 r/(2w+1)M_6$ to calculate g^r on average. If we use the projective representation and the $\mathbb{T}_2$ affine representation, then we replace M_6 with $2M_3 = 12M_1$.

An Improved Twisted Ate Pairing over KSS Curves with $k = 18$

Shan Chen[1,2], Kunpeng Wang[1], and Dongdai Lin[1]

[1] SKLOIS, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, 100093, P.R. China
[2] Graduate University of Chinese Academy of Sciences, Beijing, 100049, P.R. China
chenshanabcdef9@gmail.com, kunpengwang@263.net, ddlin@iie.ac.cn

Abstract. When implementing an efficient pairing calculation over KSS curves with embedding degree 18 and order r, the lower bound of the number of loop iterations of Miller's algorithm is $\frac{1}{6}\lfloor \log_2 r \rfloor$. But the twisted Ate pairing requires $\frac{1}{2}\lfloor \log_2 r \rfloor$ loop iterations, and thus is slower than the optimal Ate pairing which achieves the lower bound. This paper proposes an improved twisted Ate pairing and uses multi-pairing techniques to compute it. Therefore, the number of loop iterations in Miller's algorithm for the new pairing achieves the lower bound and it becomes faster than the original twisted Ate pairing by 30%.

Keywords: pairing-based cryptography, Miller's algorithm, twisted Ate pairing, multi-pairing, KSS curves.

1 Introduction

In the past years, pairing-based cryptographic applications have developed at an extraordinary pace for bilinear pairings can be used in many "constructive" ways, such as key agreement schemes [8], ID-based cryptography [5] and group signature schemes [12]. Since the implementation of pairing-based cryptosystems involves pairing evaluation, the development of efficient pairing calculations becomes a significant topic of research. The most common pairings used in applications are the Tate and Weil pairings on elliptic curves over finite fields. In general, pairing calculation can be divided into two parts, the first one being Miller's algorithm [11] and the second one the final exponentiation. Miller's algorithm is an iterative algorithm that can evaluate rational functions from scalar multiplications of divisors, and compute bilinear pairings at a linear complexity cost with respect to the size of the input. In practice, many methods have been designed to optimize Miller's algorithm, such as denominator elimination [3], the selection of pairing-friendly groups [4] and the methods to shorten the Miller loop [2,13,20]. During all these methods, shortening the Miller loop is regarded as one of the most important methods. Following this idea, several efficient pairings have been proposed, such as the Ate pairings [13], the twisted Ate pairings [13], the R-ate pairings [10] and optimal pairings [20]. It is proved that all pairings are in a

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 35–45, 2013.

group from an abstract point of view [22]. Hess's research [14] proves the lower bound of loop iterations of Miller's algorithm is $\log_2 r/\varphi(k)$. It means that any non-degenerate pairing on an elliptic curve without extra efficiently computable endomorphisms different from the Frobenius requires at least $\log_2 r/\varphi(k)$ basic Miller operations. Optimal pairings attain this lower bound by their definition.

It is well known that the Ate pairing and the twisted Ate pairing are defined on $\mathbb{G}_2 \times \mathbb{G}_1$ and $\mathbb{G}_1 \times \mathbb{G}_2$, respectively. Generally, the group $\mathbb{G}_1$ is defined over a prime field $\mathbb{F}_p$, and the group $\mathbb{G}_2$ is defined over an extension of $\mathbb{F}_p$. Their precise definitions can be found in Section 2. Recently, many efficiency improvements for the Ate pairing are proposed, such as Ate_i pairings [21], the R-ate pairings and optimal pairings. Comparing with them, the twisted Ate pairing is usually slower for its large number of loop iterations. But the twisted Ate pairing is defined by the points in $\mathbb{G}_1$, so its calculation requires less operations in the extended field than that of Ate-type pairings. From this view, it is meaningful to accelerate the twisted Ate pairing by shortening its number of loop iterations.

Recently, Sakemi *et al.* [15] propose an improved twisted Ate pairing using Frobenius maps and a small scalar multiplication over BN elliptic curves. The proposal splits Miller's algorithm into several independent parts, for which multi-pairing techniques apply efficiently. So the proposed twisted Ate pairing with multi-pairing techniques becomes faster than the original twisted Ate pairing. However, as the authors of [15] mention, the target pairing-friendly curves on which the proposed twisted Ate pairing becomes more efficient than the original twisted Ate pairing are restricted. And it is not always possible to combine an efficient split together with an efficient multi-pairing techniques. Fortunately, we find this method can also be applied to KSS curves with embedding degree $k = 18$ [9] which have already been identified as a suitable candidate for secure pairings at the 192-bit security level by Scott [18]. Similarly, we propose an improved twisted Ate pairing over this family of elliptic curves.

In this paper, we consider the family of KSS curves with embedding degree $k = 18$ [6], which is a class of ordinary pairing-friendly elliptic curves. For KSS curves the number of loop iterations of the Ate pairing, the twisted Ate pairing, and the optimal Ate pairing are about $\frac{2}{3}\lfloor \log_2 r \rfloor$, $\frac{1}{2}\lfloor \log_2 r \rfloor$, $\frac{1}{6}\lfloor \log_2 r \rfloor$, respectively. So the twisted Ate pairing is often slower than the optimal Ate pairing. Here, we use a special feature of this family to give an improved twisted Ate pairing, whose loop iterations can achieve $\frac{1}{6}\lfloor \log_2 r \rfloor$ when using multi-pairing techniques. Therefore, the new pairing is faster than the original twisted Ate pairing by 30%. Embedding degree 18 is useful and practical because many techniques promoting the efficiency of pairings are available. And the proposed pairing is of twisted type and optimal. So our proposal may provide efficient algorithms in many protocols.

This paper is organized as follows. Section 2 and 3 recall the fundamentals from [15]. Section 4 gives new results of this paper. In Section 5, we compare the proposed pairing and the original twisted Ate pairing.

2 Twisted Ate Pairing for KSS Curves with $k = 18$

Let $E/\mathbb{F}_p$ be an elliptic curve over a finite field $\mathbb{F}_p$ and p is a prime number. Denote be $E(\mathbb{F}_p)$ the set of rational points on the curve, which is an additive Abelian group with the infinity point $\mathbb{O}$ as the identity element. Let r be a prime number that divides $\#E(\mathbb{F}_p)$. The smallest positive integer k such that r divides $p^k - 1$ is called the embedding degree. Then we have the relationship $\#E(\mathbb{F}_p) = p + 1 - t$, where t is the Frobenius trace of $E(\mathbb{F}_p)$.

KSS curves with embedding degree $k = 18$, constructed in paper [9], are a class of ordinary pairing-friendly elliptic curves. They are defined over a prime field $\mathbb{F}_p$ and have the short Weierstrass equation $E : y^2 = x^3 + b$, $b \in \mathbb{F}_p$. The related parameters of the KSS curves are given with an integer χ as follows:

$$\begin{cases} p(\chi) = \frac{1}{21}(\chi^8 + 5\chi^7 + 7\chi^6 + 37\chi^5 + 188\chi^4 + 259\chi^3 + 343\chi^2 + 1763\chi + 2401), \\ r(\chi) = \frac{1}{343}(\chi^6 + 37\chi^3 + 343), \\ t(\chi) = \frac{1}{7}(\chi^4 + 16\chi + 7). \end{cases} \tag{1}$$

In order to keep p and r being primes, we just consider the case in which $\chi \equiv 14 \mod 42$. In the following of this paper, we will substitute χ with $14 + 42\chi$. Since the explicit coefficients of the new parameters are too large, we won't list them here.

The general definition of the twisted Ate pairing on elliptic curves can be found in paper [13]. Here we give the twisted Ate pairing on KSS curves with $k = 18$ according to the general definition. Let $\phi : E(\mathbb{F}_{p^{18}}) \to E(\mathbb{F}_{p^{18}})$ be the Frobenius endomorphism, $E(\mathbb{F}_{p^{18}})[r]$ denote the subgroup of rational points of order r in $E(\mathbb{F}_{p^{18}})$ and let ζ_6 be a primitive 6-th root of unity. Define

$$[\zeta_6] : E \to E, (x, y) \mapsto ({\zeta_6}^2 x, {\zeta_6}^3 y).$$

Let

$$\begin{aligned} \mathbb{G}_1 &= \langle P \rangle = E(\mathbb{F}_{p^{18}})[r] \cap \mathrm{Ker}(\phi - [1]), \\ \mathbb{G}_2 &= \langle Q \rangle = E(\mathbb{F}_{p^{18}})[r] \cap \mathrm{Ker}([\zeta_6]\phi^3 - [1]) = E(\mathbb{F}_{p^{18}})[r] \cap \mathrm{Ker}(\phi - [p]). \end{aligned}$$

Then the twisted Ate pairing is defined as

$$\alpha(\cdot, \cdot) : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{F}^*_{p^{18}} / \mathbb{F}^{*r}_{p^{18}}, (P, Q) \mapsto f_{T^3,P}(Q)^{(p^{18}-1)/r},$$

where $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and $T^3 \equiv p^3 \mod r$. When $f_{T^3,P}(Q)^{(p^{18}-1)/r}$ is calculated using Miller's algorithm and the final exponentiation, the number of loop iterations of Miller's algorithm to compute the twisted Ate pairing is determined by $\log_2 T^3$, and

$$T^3 \equiv p^3 \equiv -74088\chi^3 - 74088\chi^2 - 24696\chi - 2762 \mod r. \tag{2}$$

3 Review of Useful Tools

In this section, we cite some useful tools which are already known in the literature, concluding divisor theory, skew Frobenius maps and multi-pairing techniques. In fact, all these objects have been explained carefully by Sakemi *et al.* in

the section 2 of [15] when they improve the twisted Ate pairing over BN elliptic curves. But for the integrality of this article, we list them again.

3.1 Divisors

For $a, b \in \mathbb{Z}$ and $Q \in E[r]$, let $f_{a,Q}, f_{b,Q} \in \mathbb{F}_{p^k}(E)$ be the uniquely determined monic functions with

$$\begin{aligned} \mathrm{div}(f_{a,Q}) &= a(Q) - ([a]Q) - (a-1)(\mathcal{O}), \\ \mathrm{div}(f_{b,Q}) &= b(Q) - ([b]Q) - (b-1)(\mathcal{O}). \end{aligned}$$

Then there exist the following relations [15]:

$$\begin{aligned} f_{a+b,Q} &= f_{a,Q} \cdot f_{b,Q} \cdot g_{[a]Q,[b]Q}, \\ f_{ab,Q} &= f_{a,Q}^{b} \cdot f_{b,[a]Q} = f_{b,Q}^{a} \cdot f_{a,[b]Q}, \end{aligned}$$

where $g_{[a]Q,[b]Q} = l_{[a]Q,[b]Q}/v_{[a]Q+[b]Q}$, $l_{[a]Q,[b]Q}$ denotes the line which passes through $[a]Q$ and $[b]Q$, and $v_{[a]Q+[b]Q}$ denotes the vertical line which passes through the point $[a]Q + [b]Q$.

3.2 Skew Frobenius Maps

Let E be an ordinary elliptic curve over a finite field $\mathbb{F}_p$ and E' be the twisted elliptic curve of E with degree d defined over the finite field $\mathbb{F}_{p^e}$, where e is a positive integer. Then the embedding degree $k = de$ and the twist isomorphism is given as follows:

$$\psi_d : E^{'}(\mathbb{F}_{p^e}) \to E(\mathbb{F}_{p^{de}}), (x, y) \mapsto (xv^{2/d}, yv^{3/d}).$$

Corresponding to the twist degree d, v is chosen as a quadratic non residue, a cubic non residue, or a quadratic and cubic non residue in $\mathbb{F}_{p^e}$.

Since $P^{'} = \psi_d^{-1}(P) \subset E^{'}(\mathbb{F}_{p^k})$ for an arbitrary rational point $P \in \mathbb{G}_1 \subset E(\mathbb{F}_p)$, the following relation [13] holds,

$$(\phi_e - [p^e])P^{'} = \mathbb{O}, \quad \phi_e(P^{'}) = (x_{P'}^{p^e}, y_{P'}^{p^e}),$$

where $P^{'} = (x_{P'}, y_{P'})$. Then the skew Frobenius map $\tilde{\phi}_e$ is defined as [16] :

$$\tilde{\phi}_e : \mathbb{G}_1 \to \mathbb{G}_1, (x, y) \mapsto (x^p/v^{2(p^e-1)/d}, y^p/v^{3(p^e-1)/d}).$$

And the following relation holds, $\tilde{\phi}_e(P) = [p^e]P$, which will be used in Algorithm 2.

In the case of KSS curves considered in this paper, since $k = 18,\ d = 6,\ e = k/d = 3$, the skew Frobenius map $\tilde{\phi}_3$ is

$$\tilde{\phi}_3 : \mathbb{G}_1 \to \mathbb{G}_1, (x, y) \mapsto (x/v^{(p^3-1)/3}, y/v^{(p^3-1)/2}),$$

and $\tilde{\phi}_3(P) = [p^3]P$.

3.3 Multi-pairing

In order to calculate the product of pairings efficiently, Granger et al. have provided the multi-pairing algorithm in [7]. Let

$$S_P = \{P_1, P_2, \cdots, P_N \in \mathbb{G}_1\},$$
$$S_Q = \{Q_1, Q_2, \cdots, Q_N \in \mathbb{G}_2\},$$

then the product of N pairings $M_N = \prod_{i=1}^{N} f_{s,P_i}(Q_i)^{(p^k-1)/r}$ can be calculated by Algorithm 1, called $MMA(s, N, S_P, S_Q)$, which is shown in Section 2.8 of [15].

Algorithm 1. Miller's Algorithm for multi-pairing $MMA(s, N, S_P, S_Q)$

Input: $s, N \in \mathbb{N}$, $S_P = \{P_1, P_2, \cdots, P_N \in \mathbb{G}_1\}$, $S_Q = \{Q_1, Q_2, \cdots, Q_N \in \mathbb{G}_2\}$
Output: $\prod_{i=1}^{N} f_{s,P_i}(Q_i)$, $S_R = \{[s]P_1, [s]P_2, \cdots, [s]P_N \in \mathbb{G}_1\}$
1: Write $s = \sum_{j=0}^{L} s_j 2^j$, with $s_j \in \{0, 1\}$ and $s_L = 1$
2: $f \leftarrow 1$
3: **for** i from N downto 1 **do**
4: $R_i \leftarrow P_i$
5: **end for**
6: **for** j from $L - 1$ downto 0 **do**
7: $f \leftarrow f^2$
8: **for** i from N downto 1 **do**
9: $f \leftarrow f \cdot g_{R_i,R_i}(Q_i)$; $R_i \leftarrow [2]R_i$
10: **end for**
11: **if** $s_j = 1$ **then**
12: **for** i from N downto 1 **do**
13: $f \leftarrow f \cdot g_{R_i,P_i}(Q_i)$; $R_i \leftarrow R_i + P_i$
14: **end for**
15: **end if**
16: **end for**
17: **return** $f, [R_1, R_2, \cdots, R_N]$

4 New Results

In this section, we propose an improved twisted Ate pairing over KSS curves with $k = 18$. The new pairing can be calculated efficiently when we use multi-pairing techniques. In fact, this pairing is based on the relation given by an equation

$$p \equiv 2z_0 + z_0 p^3 \mod r, \tag{3}$$

which is from the example in [20] of an optimal pairing over KSS curves with $k = 18$. First, we use the special equation to construct the new pairing. Then we give the complete proof of the bi-linearity and non-degeneracy of the improved twisted Ate pairing in the appendix. At last, we use the multi-pairing techniques to calculate it and give the explicit algorithm.

4.1 Construction

In this subsection, we will give the new pairing which is denoted by $\alpha_{pt}(\cdot,\cdot)$. The shortest vector $V=[2z,1,0,z,0,0]$ in [20] means that, there exists an integer m such that

$$mr = 2z + p + zp^3.$$

Set $z_0 = -z = -\chi/7$, then an equation $p \equiv 2z_0 + z_0p^3 \mod r$ can be obtained easily. It is well known that, when the embedding degree k is an even number such as in the case of KSS curves with $k=18$, all terms of $v_{[a]P+[b]P}(Q)$ lie in a subfield of $\mathbb{F}^*_{p^{18}}$ and thus may be eliminated in the final exponentiation. So in the following calculation, we use $l_{[a]P,[b]P}(Q)$ instead of $g_{[a]P,[b]P}(Q)$.

According to Section 3.1, the calculation of the pairing $f_{p^3,P}(Q)^{(p^{18}-1)/r}$ over KSS curves is given by

$$f_{p^3,P}(Q)^{(p^{18}-1)/r} = (f^{p^2}_{p,P}(Q)\cdot f^{p}_{p,[p]P}(Q)\cdot f_{p,[p^2]P}(Q))^{(p^{18}-1)/r}.$$

Let $P_p=[p]P,\ P_{p^2}=[p^2]P,\ Q_p=[p]Q,\ Q_{p^2}=[p^2]Q$. Since $P\in\mathbb{G}_1$, $Q\in\mathbb{G}_2$, particularly, $\phi(P)=P,\ \phi(Q)=[p]Q=Q_p$, then we can see

$$f^{p}_{p,P}(Q) = f_{p,\phi(P)}(\phi(Q)) = f_{p,P}(Q_p).$$

It follows that

$$\begin{aligned} f_{p^3,P}(Q)^{(p^{18}-1)/r} &= (f^{p^2}_{p,P}(Q)\cdot f^{p}_{p,[p]P}(Q)\cdot f_{p,[p^2]P}(Q))^{(p^{18}-1)/r} \\ &= (f_{p,P}(Q_{p^2})\cdot f_{p,[p]P}(Q_p)\cdot f_{p,[p^2]P}(Q))^{(p^{18}-1)/r} \\ &= (f_{p,P}(Q_{p^2})\cdot f_{p,P_p}(Q_p)\cdot f_{p,P_{p^2}}(Q))^{(p^{18}-1)/r}. \end{aligned}$$

Let $p = 2z_0 + z_0p^3 + cr, c\in\mathbb{Z}$, then we obtain

$$\begin{aligned} f_{p,P}(Q_{p^2}) &= f_{2z_0+z_0p^3+cr,P}(Q_{p^2}) \\ &= (f^{p^3+2}_{z_0,P}\cdot l_{[z_0]P,[z_0]P}\cdot l_{[2z_0]P,[z_0p^3]P}\cdot f_{p^3,[z_0]P}\cdot f^{c}_{r,P})(Q_{p^2}), \\ f_{p,P_p}(Q_p) &= f_{2z_0+z_0p^3+cr,P_p}(Q_p) \\ &= (f^{p^3+2}_{z_0,P_p}\cdot l_{[z_0]P_p,[z_0]P_p}\cdot l_{[2z_0]P_p,[z_0p^3]P_p}\cdot f_{p^3,[z_0]P_p}\cdot f^{c}_{r,P_p})(Q_p), \\ f_{p,P_{p^2}}(Q) &= f_{2z_0+z_0p^3+cr,P_{p^2}}(Q) \\ &= (f^{p^3+2}_{z_0,P_{p^2}}\cdot l_{[z_0]P_{p^2},[z_0]P_{p^2}}\cdot l_{[2z_0]P_{p^2},[z_0p^3]P_{p^2}}\cdot f_{p^3,[z_0]P_{p^2}}\cdot f^{c}_{r,P_{p^2}})(Q). \end{aligned}$$

Define

$$\begin{aligned} \tilde{f}_{z_0,P}(Q) = {} & [f_{z_0,P}(Q_{p^2})\cdot f_{z_0,P_p}(Q_p)\cdot f_{z_0,P_{p^2}}(Q)]^{p^3+2} \\ & \cdot(l_{[z_0]P,[z_0]P}\cdot l_{[2z_0]P,[z_0p^3]P})(Q_{p^2})\cdot(l_{[z_0]P_p,[z_0]P_p}\cdot l_{[2z_0]P_p,[z_0p^3]P_p})(Q_p) \\ & \cdot(l_{[z_0]P_{p^2},[z_0]P_{p^2}}\cdot l_{[2z_0]P_{p^2},[z_0p^3]P_{p^2}})(Q), \end{aligned} \tag{4}$$

then we can define the improved twisted Ate pairing $\alpha_{pt}(\cdot,\cdot)$ as

$$\alpha_{pt}(\cdot,\cdot): \mathbb{G}_1\times\mathbb{G}_2 \to \mathbb{F}^*_{p^{18}}/\mathbb{F}^{*r}_{p^{18}}, (P,Q)\mapsto \tilde{f}_{z_0,P}(Q)^{(p^{18}-1)/r}, \tag{5}$$

where $P\in\mathbb{G}_1, Q\in\mathbb{G}_2$. The bi-linearity and non-degeneracy of $\alpha_{pt}(\cdot,\cdot)$ is shown in the appendix.

4.2 Calculation

In this subsection, we discuss how to use multi-pairing techniques to calculate the new pairing $\alpha_{pt}(\cdot,\cdot)$ and give the explicit algorithm which is shown in Algorithm 2.

Algorithm 2. Miller's Algorithm for $\tilde{f}_{z_0,P}(Q)$

Input: $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, z_0, p$
Output: $\tilde{f}_{z_0,P}(Q)$
1: $P_1 \leftarrow [z_0]P$
2: $P_2 \leftarrow [2]P_1$
3: $P_3 \leftarrow \tilde{\phi}_3(P_1)$
4: $Q_3 \leftarrow Q, Q_2 \leftarrow \phi(Q), Q_1 \leftarrow \phi^2(Q)$
5: $B \leftarrow l_{P_1,P_1}(Q_1)$
6: $C \leftarrow l_{P_2,P_3}(Q_1)$
7: $f \leftarrow B \cdot C$
8:$P_p \leftarrow P_2 + P_3$
9:$P_1 \leftarrow [z_0]P_p$
10:$P_2 \leftarrow [2]P_1$
11:$P_3 \leftarrow \tilde{\phi}_3(P_1)$
12:$B \leftarrow l_{P_1,P_1}(Q_2)$
13:$C \leftarrow l_{P_2,P_3}(Q_2)$
14:$f \leftarrow f \cdot B \cdot C$
15:$P_{p^2} \leftarrow P_2 + P_3$
16:$P_1 \leftarrow P, P_2 \leftarrow P_p, P_3 \leftarrow P_{p^2}$
17:$A \leftarrow MMA(z_0, 3, S_P, S_Q)$
18:$P_1 \leftarrow R_3$
19:$P_2 \leftarrow [2]P_1$
20:$P_3 \leftarrow \tilde{\phi}_3(P_1)$
21:$B \leftarrow l_{P_1,P_1}(Q_3)$
22:$C \leftarrow l_{P_2,P_3}(Q_3)$
23:$f \leftarrow f \cdot B \cdot C$
24:$f \leftarrow A^{p^3+2} \cdot f$
25:return f

Set

$$\begin{aligned} A &= f_{z_0,P}(Q_{p^2}) \cdot f_{z_0,P_p}(Q_p) \cdot f_{z_0,P_{p^2}}(Q), \\ L &= (l_{[z_0]P,[z_0]P} \cdot l_{[2z_0]P,[z_0p^3]P})(Q_{p^2}) \\ &\quad \cdot (l_{[z_0]P_p,[z_0]P_p} \cdot l_{[2z_0]P_p,[z_0p^3]P_p})(Q_p) \\ &\quad \cdot (l_{[z_0]P_{p^2},[z_0]P_{p^2}} \cdot l_{[2z_0]P_{p^2},[z_0p^3]P_{p^2}})(Q), \end{aligned}$$

then $\tilde{f}_{z_0,P}(Q) = A^{p^3+2} \cdot L$. In order to use multi-pairing techniques to calculate A, we set $S_P = \{P, P_p, P_{p^2}\}, S_Q = \{Q_{p^2}, Q_p, Q\}, N = 3, s = z_0$. Then $A = MMA(z_0, 3, S_P, S_Q)$. When it comes to L, we can compute P_p and P_{p^2} by using the equation $p \equiv 2z_0 + z_0p^3 \mod r$ as follows:

$$P_p = [p]P = [2z_0]P + [z_0p^3]P = [2z_0]P + [z_0]\tilde{\phi}_3(P),$$
$$P_{p^2} = [p]P_p = [2z_0]P_p + [z_0p^3]P_p = [2z_0]P_p + [z_0]\tilde{\phi}_3(P_p),$$

where $\tilde{\phi}_3$ is the skew Frobenius map introduced in Section 3.2, thus $[p^3]P = \tilde{\phi}_3(P)$. On the other hand, $Q_p = [p]Q = \phi(Q)$ is easily computed by the Frobenius map. Since $[z_0]P_{p^2}$ can be obtained for free as the output of $MMA(z_0, 3, S_P, S_Q)$, the main part of computing L is $[z_0]P$ and $[z_0]P_p$. But they are just scalar multiplications over the base field and z_0 is usually very short, so these calculations are much cheaper than the calculation of MMA. In conclusion, L can be calculated efficiently and cheaply. The complete algorithm is shown in Algorithm 2.

When it comes to the implementation of the improved twisted Ate pairing, besides the multi-pairing techniques, there are many other general techniques that can be used. Roughly speaking, the extension tower:

$$\mathbb{F}_p \subset \mathbb{F}_{p^3} \subset \mathbb{F}_{p^{18}}$$

can be used to construct the finite field $\mathbb{F}_{p^{18}}$ by simple polynomials. Sextic twists of KSS curves with embedding degree $k = 18$ can be used for pairing calculation and rational point compression. The final exponentiation can be dealt with by the Frobenius map in the finite field and the details about this techniques can be found in [19]. But all these general techniques are not the important topics in this paper. We just focus on the multi-pairing techniques, which is the point why the proposed twisted Ate pairing is faster than the conventional twisted pairing.

5 Efficiency Comparison

This section compares the calculation costs of the new pairing

$$\alpha_{pt}(\cdot,\cdot) = \tilde{f}_{z_0,P}(Q)^{(p^{18}-1)/r} = (A^{p^3+2} \cdot L)^{(p^{18}-1)/r}$$

with the conventional twisted Ate pairing

$$\alpha(\cdot,\cdot) = f_{T^3,P}(Q)^{(p^{18}-1)/r}.$$

Since both of them involve the same final powering step , L can be computed almost for free and the small cost of A^{p^3+2} can be ignored for Frobenius map in the finite field is efficient, we need only focus on comparing the costs of A and $f_{T^3,P}(Q)$.

Following [13], let M_s, S_s denote the cost of Multiplication and Squaring in the finite field $\mathbb{F}_{q^s}$. If we use the pairing-friendly fields with $s = 2^i3^j$, then we have $M_s = 3^i5^jM_1, S_s = 3^i5^jS_1$. We refer to the $f_{N,P}(Q)$ as a Miller-Lite operation and denote the cost of Miller-Lite algorithm by C_{Lite}. Then for the elliptic curves of the form

$$Y^2 = X^3 + B,$$

the cost of $f_{N,P}(Q)$ is

$$C_{Lite} = (5S_1 + (2e + 6)M_1 + S_k + M_k)\lfloor \log_2 N \rfloor.$$

In the case of KSS curves, $s = 18$, $i = 1$, $j = 2$, $e = 3$, $k = 18$, and in order to compare the costs of two pairings explicitly, we assume $S_1 = M_1$. Then $C_{Lite} = 167S_1\lfloor \log_2 N \rfloor$. Denote the costs of $f_{T^3,P}(Q)$ and A by c_1 and c_2, respectively. Considering equation (3) and z_0, we have

$$c_1 = 167S_1\lfloor \log_2 T^3 \rfloor \approx 501S_1\lfloor \log_2 \chi \rfloor,$$

$$c_2 = 351S_1\lfloor \log_2 z_0 \rfloor \approx 351S_1\lfloor \log_2 \chi \rfloor,$$

where χ is the curve parameter. So the new pairing with the multi-pairing techniques becomes faster than the original twisted Ate pairing by 30%. For example, set $\chi = -2^{62} - 1$, then $\lfloor \log_2 z_0 \rfloor = 64$, $\lfloor \log_2 T^3 \rfloor = 202$, it follows that $c_1 = 167 \times 202S_1 = 33734S_1$, $c_2 = 351 \times 64S_1 = 22464S_1$.

6 Conclusion and Future Work

This paper uses the special equation $p \equiv 2z_0 + z_0p^3 \mod r$ to construct a new pairing $\alpha_{pt}(\cdot, \cdot)$ defined over $\mathbb{G}_1 \times \mathbb{G}_2$ on KSS curves with embedding degree $k = 18$. Following the multi-pairing techniques, the maximum of the length of loop reaches the lower bound $\frac{1}{6}\lfloor \log_2 r \rfloor$. Thus the improved twisted Ate pairing is faster than the original one by 30%.

Neither the new pairing nor the algorithms considered herein are exhaustive; we thus hope that these are the first steps toward further improvement of the twisted Ate pairing. We leave it as future work to find useful equations for other pairing-friendly curve families to give more improved twisted Ate pairings and to compute them using the multi-core parallelization approach proposed by Aranha *et al.* in [1].

Acknowledgments. We would like to thank the anonymous reviewers for their helpful comments. In particular, we are heartily grateful to Michael Naehrig for his insightful suggestions. This work is supported by the National 973 Program of China under Grant 2011CB302400, the National Natural Science Foundation of China under Grant 60970152 and 60970153, the Grand Project of Institute of Software under Grant YOCX285056.

References

1. Aranha, D.F., Knapp, E., Menezes, A., Rodríguez-Henríquez, F.: Parallelizing the Weil and Tate Pairings. In: Chen, L. (ed.) Cryptography and Coding 2011. LNCS, vol. 7089, pp. 275–295. Springer, Heidelberg (2011)
2. Barreto, P.S.L.M., Galbraith, S., hÉigeartaigh, C.Ó., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007)

3. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
4. Barreto, P.S.L.M., Lynn, B., Scott, M.: On the Selection of Pairing-Friendly Groups. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 17–25. Springer, Heidelberg (2004)
5. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
6. Brezing, A.W.: Elliptic curves suitable for pairing-based cryptography. Designs, Codes and Cryptography 37(1), 133–141 (2005)
7. Granger, R., Smart, N.P.: On computing products of pairings. Cryptology ePrint Archive: Report 2006/172
8. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. Journal of Cryptology 17, 263–276 (2004)
9. Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 126–135. Springer, Heidelberg (2008)
10. Lee, E., Lee, H., Park, C.: Efficient and generalized pairing computation on Abelien varieties. IEEE Transactions on Information Theory 55(4), 1793–1803 (2009)
11. Miller, V.S.: The Weil Pairing and its efficient calculation. Journal of Cryptology 17(4), 235–261 (2004)
12. Nakanishi, T., Funabiki, N.: Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 533–548. Springer, Heidelberg (2005)
13. Hess, F., Smart, N.P., Vercauteren, F.: The eta pairing revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
14. Hess, F.: Pairing Lattices. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 18–38. Springer, Heidelberg (2008)
15. Sakemi, Y., Takeuchi, S., Nogami, Y., Morikawa, Y.: Accelerating Twisted Ate Pairing with Frobenius Map, Small Scalar Multiplication, and Multi-pairing. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 47–64. Springer, Heidelberg (2010)
16. Sakemi, Y., Nogami, Y., Okeya, K., Kato, H., Morikawa, Y.: Skew Frobenius Map and Efficient Scalar Multiplication for Pairing–Based Cryptography. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 226–239. Springer, Heidelberg (2008)
17. Silverman, J.H.: The arithmetic of elliptic curves, 2nd edn. GTM 106 (2009)
18. Scott, M.: Faster Pairings Using an Elliptic Curve with an Efficient Endomorphism. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 258–269. Springer, Heidelberg (2005)
19. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 78–88. Springer, Heidelberg (2009)
20. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory 56(1), 455–461 (2010)
21. Zhao, C.-A., Zhang, F., Huang, J.: A note on the Ate pairing. Int. J. Inf. Security Arch. 7(6), 379–382 (2008)

22. Zhao, C.-A., Zhang, F., Huang, J.: All pairings are in a group. IEICE Trans. Fundam. E91-A(10), 3084–3087 (2008)

A The Bi-linearity and Non-degeneracy

This part can be divided into two parts, one is about the bi-linearity of $f_{p^3,P}(Q)$, the other is about the bi-linearity and non-degeneracy of $\alpha_{pt}(\cdot,\cdot)$.

Let $\gamma = [\zeta_6] \circ \phi^3$, then $\gamma(P) = [p^3]P, \gamma(Q) = Q$. Since γ is purely inseparable of degree p^3, we obtain from Lemma 4 in [13]

$$f_{p^3,\gamma(P)} \circ \gamma = f_{p^3,P}^{p^3}.$$

Then we have

$$f_{p^3,[p^3]P}(Q) = f_{p^3,\gamma(P)} \circ \gamma(Q) = f_{p^3,P}^{p^3}(Q).$$

Since $p^{18} \equiv 1 \mod r$, we can set $mr = p^{18} - 1 = (p^3)^6 - 1$, thus

$$\begin{aligned} f_{r,P}^m(Q) &= f_{mr,P}(Q) \\ &= f_{p^{18}-1,P}(Q) \\ &= f_{p^{18},P}(Q) \\ &= f_{(p^3)^6,P}(Q) \\ &= f_{p^3,P}^{(p^3)^5}(Q) \cdot f_{p^3,[p^3]P}^{(p^3)^4}(Q) \ldots f_{p^3,[(p^3)^5]P}(Q) \\ &= f_{p^3,P}^{6\times p^{15}}(Q). \end{aligned}$$

Then $f_{p^3,P}(Q) = f_{r,P}^{m/(6\times p^{15})}(Q) = f_{r,P}^{mp^3/6}(Q)$ is a bi-linear pairing for the Tate pairing $f_{r,P}(Q)$ is bi-linear.

Section 3.1 shows that

$$\begin{aligned} f_{p^3,P}(Q)^{(p^{18}-1)/r} &= \tilde{f}_{z_0,P}(Q)^{(p^{18}-1)/r} \cdot [f_{p^3,[z_0]P}(Q_{p^2}) \cdot f_{r,P}^c(Q_{p^2}) \cdot f_{p^3,[z_0]P_p}(Q_p) \\ &\quad \cdot f_{r,P_p}^c(Q_p) \cdot f_{p^3,[z_0]P_{p^2}}(Q) \cdot f_{r,P_{p^2}}^c(Q)]^{(p^{18}-1)/r} \\ &= \tilde{f}_{z_0,P}(Q)^{(p^{18}-1)/r} \cdot [f_{p^3,P}^{3z_0p^2}(Q) \cdot f_{r,P}^{3cp^2}(Q)]^{(p^{18}-1)/r}. \end{aligned}$$

Therefore

$$\tilde{f}_{z_0,P}(Q)^{(p^{18}-1)/r} = [f_{p^3,P}^{1-3z_0p^2}(Q) \cdot f_{r,P}^{3cp^2}(Q)]^{(p^{18}-1)/r}.$$

Since $f_{p^3,P}(Q) = f_{r,P}^{mp^3/6}(Q)$, then it follows that

$$\tilde{f}_{z_0,P}(Q)^{(p^{18}-1)/r} = [f_{r,P}^{mp^3(1-3z_0p^2)/6-3cp^2}(Q)]^{(p^{18}-1)/r}.$$

Set

$$N = mp^3(1 - 3z_0p^2)/6 - 3cp^2 \mod r,$$

then N and r are polynomials of integer χ. Through calculation, it can be proved that $r \nmid N$. So

$$\alpha_{pt}(\cdot,\cdot) = \tilde{f}_{z_0,P}(Q)^{(p^{18}-1)/r}$$

is a pairing with bi-linearity and non-degeneracy.

Controlled Joining on Encrypted Relational Database

Jun Furukawa and Toshiyuki Isshiki

NEC Corporation Japan

Abstract. If a user encrypts data, stores them in a relational database (RDB), and keeps the key for both encryption and decryption by himself, then the risk of leaking data from the RDB directly can be mitigated. Such a strategy can be considered as a natural solution for preventing data leakage when the manager of the database cannot be entirely trusted or the burden of managing the database needs to be lightened. However, if the database cannot access to this key, it can execute only a few relational algebraic operations by itself, which spoils the serviceability of the database.

This paper first introduces the notion of an encryption for controlled joining (ECJ), which enables RDB to execute "natural join" of tables when and only when its user required it. This technique can directly be applied for union, difference, and intersection of tables also. Then, the paper proposes an instance under a novel but natural assumption on asymmetric bilinear group. Combining an ECJ with a searchable encryption and an order-preserving encryption, one can construct an encrypted database which can executes the major part of relational algebraic operations. The proposed instance is efficient in a reasonable extent and sacrifices its security only in a minimum extent. We consider such a technique can bring an enhanced security into the database-as-service environment.

Keywords: encrypted RDB, natural join, union, difference, intersection, asymmetric bilinear group, non-transitivity.

1 Introduction

1.1 Encryption of Relational Database

A database (DB) is a system in which a large amount of data is stored and portions of it can be retrieved smoothly. It has been an indispensable platform for providing variety of services through the network. Since many of DBs store sensitive information such as customer information, private information, or trade secrets, they are potentially vulnerable to abuse, leakage, and theft. Hence, it is crucially important to unfailingly protect confidentiality of data in many DBs.

The primary method of protecting data in today's DB systems is access control. Although this has been a fairly effective approach, it can no longer be highly reliable if the DB can potentially be compromised. And DBs, indeed, may be

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 46–64, 2013.

compromised in diverse ways, e.g., physical theft of hard disks or other memories, data leakage by malicious or careless system managers, leakage by viruses infected, via unpatched vulnerability of the system, design error, configuration fault, etc. Hence, in addition to access control, it is desirable to enforce the confidentiality of DBs by encryption. Such a succinct strategy is considered to be especially effective for the database-as-service environment.

Encryption is already an accepted approach to data protection for DBs. For example, PCI DSS (PCI Data Security Standard) [32] which is for to enforce payment account data security requires stored card-holder data to be encrypted. And several existing DB applications such as [18,29,31] actually support encryption of stored data. Since the outsourcing of data and services are getting more common, as we can also recognize in the recent widespread of cloud computing services, the situation are getting worse and more complicated. Hence, it is envisaged that encryption of DBs will become more common and vital.

In encryption mechanisms of DBs that are already in use [18,29,31], the keys used to encrypt data are kept within their systems. Hence, these keys may be leaked with the data themselves when the DBs are compromised. In this sense, such mechanisms are still not a satisfactory approach for high leakage resistant DB systems. Improving these solutions, several systems such as [15,24,25,33] encrypt sensitive data, store them in an outsourced DB, and keep the key for the encryption under control of the user. The main challenge in this approach is to avoid imposing users to retrieve all data in the DB, decrypt them, and find necessary data among them when they use the DB. This is because the most significant serviceability of DBs is spoiled if it imposes a large amount of computation and communication on users.

A searchable encryption [3,5,21,23] and an order preserving encryption [2,9,10] provide a way to salvage, in a decent extent, the serviceability of DBs when data are encrypted by keys kept by users themselves. The searchable encryption enables a DB to search necessary data among those encrypted by users without decrypting them. The order preserving encryption enables a DB to compare numerical size of data that are encrypted by users without decrypting them. Because of these ability, a DB is able to return only ciphertexts of data that are required by users. Hence, communication and computational complexity of users are substantially reduced, and thus the serviceability of DB is salvaged.

However, searchable encryption alone is often insufficient for searching data in a DB since the most used DB is relational database (RDB) [20]. An RDB system decomposes each large table (relation) into smaller and well-formed (normalized) tables so as to prevent data manipulation anomaly and data integrity loss. Then, the DB often needs to partially reconstruct the original table from normalized tables before searching data. This is done by "natural join" procedure in Structured Query Language (SQL) [20]. Therefore, without natural join, searchable encryption is incapable of salvaging serviceability of encrypted RDBs. Moreover, the natural join, union, difference, and intersection of tables are also indispensable to generate a wide variety of tables and views for variety of purposes. These are issues we focus in this paper.

More precisely, we consider the following concrete example to obtain clear understanding of the problem. Suppose that a user specifies two tables I and II. Table I has two columns, one is for attribute A and the other is for attribute B. Table II has two columns, one is for attribute C and the other is for attribute D. Then the user requires a database to join Table I and II with respect to attributes A and C, and returns a row whose attribute B is X. If the columns for attribute A of Table I and attribute C of Table II are encrypted by, for example, with different key or different randomness, the DB is unable to join them by himself. In such a case, the DB is only able to choose a row whose attribute B is X from Table I. But it is unable to choose a row in Table II that is supposed to be joined to the above chosen row in Table I. The only way that the DB can do to meet the need of the user is to send whole Table II. Consequently such a DB is not useful any more.

1.2 Encryption for Controlled Joining

As discussed in the Section 1.1, an encrypted RDB needs measures to run many SQL procedures such as a natural join without sacrificing efficiency of the users. Such an issue has been considered also in previous works [1,15,19,22,24,25,26,34,35,33]. Some works consider to **bucketize** encrypted data, where rough join is executed in DB using special indices, but the decryption and the final tuning of the join is executed by users themselves. This approach has a trade-off between amount of data leaked to the DB and the computation the users are required. Some works consider to **fragment** data in several DBs so that none of them can recover confidential relations by itself. This approach requires multiple DBs, which need to be trusted as a whole. Some works consider to **progressively decrypt** data, encrypted in layer, until they become comparable.

Another strategy is to encrypt every data deterministically as in the case of searchable encryption but with the same key for all data in the DB. Then, since values in different tables can be compared, the DB can join tables without decrypting them. However, this strategy leads to security concerns by leaking relations. Suppose that Table I contains names of card holders and encrypted credit card numbers and that Table II contains encrypted credit card numbers and invalidated dates. If the numbers in the both tables are encrypted deterministically, it is easy to recognize whose card is invalidated from the leaked tables. This is because the relation is leaked from the encrypted tables. In contrast, if the numbers in both tables are encrypted by different keys or with different randomness, it is unlikely that leakage of these tables are serious privacy exposure. Although the public-key encryption with keyword search [12] cannot solve this problem, the works [13,37], following [38], present interesting idea that key words can be searched only by delegated entities.

In this paper, we choose a novel approach and propose such an encryption scheme that the DB is able to check equivalence of encrypted values in two

columns, without decrypting them completely. This check is possible only when the user requested it to do so and only in the **minimum** extent. And, this request is quite easy for the user. With this scheme, the DB manager is able to execute natural join, union, difference, and intersection of tables once it receives the request. We call such scheme "encryption for controlled joining (**ECJ**)".

Schemes with bucketization such as those in [15,26,24] and schemes with fragmentation such as those in [19,1,22] suffer the trade-off between user's cost and data confidentiality. That means users need additional computation to enhance the confidentiality of the data. On the other hand, our scheme requires users no additional computation except the decryption of received result. Schemes with fragmentation such as those in [19,1,22] also require servers to be trusted not to collude, but our scheme does not. Although searchable encryption schemes such as those in [5,21,23], like our scheme, require users no additional computation except the decryption of received result, they leak relations to the DB as is discussed above.

We note that our scheme does not succeed in preventing the DB from obtaining relations (each data is still encrypted) between two columns when the user required to join them. Such a prevention is out of our security goal since it is likely to be impossible to prevent it unless we use such a heavy[1] cryptographic primitive as "private information retrieval protocol" introduced in [16,17,28]. DB users usually do not accept such inefficiency. However, unlike deterministic encryption approach, the leak of relations in our scheme is the minimum in the sense that each relation is hidden unless the user requires to use it for relational algebraic operations and also in the sense that the revealed relation does not unnecessary reveal other relations.

We now roughly introduce the model of the novel encryption scheme "ECJ" that satisfies the above properties:

1. Each data is encrypted with respect to some label.
2. The encryption is of symmetric key and probabilistic.
3. From any pair of two labels and the symmetric key, one can generate a projection key. With this key, the equivalence of two data which are encrypted with respect to either of these two labels can be checked.
4. The projection key is short, its generation costs is small, and the cost for checking equivalence of m encrypted values mutually is at most of order m.

We illustrate how this ECJ can be applied to an RDB system. We associate each column of tables in the DB with a label, and encrypt data in the column with respect to this associated label by using Property 1. Because of Property 2, ciphertexts alone do not give knowledge (except the size of each table and length of each data in tables) to the DB. Exploiting Property 3, the user can generate a projection key from the symmetric key and the two labels for arbitrary pair of two columns. By sending this key to the DB, the DB is able to check the

[1] Operations such as "join and select" requires a large amount of computation and communication.

equivalence of encrypted values in these two columns, each of which is related to either of the two labels (This function can simulate the equivalence condition in "`WHERE a column name = another column name`" clause for join in SQL). Hence, the RDB is able to execute natural join, union, difference, and intersection of tables without decrypting their data. As ECJ satisfies Property 4, the user can generates a projection key and send it to the DB very easily. The cost for the DB to join two tables at most order of the size of the relevant tables. After the tables are joined, the user is able to request the DB to select necessary rows from the joined table by using a searchable encryption or an order-preserving encryption. Hence, the amount of data the DB sends to the user as the response to the query is as small as that in ordinary DB systems.

The essentials of ECJ are that it enables RDBs to check equivalence of encrypted values in different tables when required but it allows the equivalence check only in the minimum extent. Only when furnished with this ability, an RDB is able to efficiently take a union, difference, intersection of tables as well as take natural join of tables. These are indubitably major relational algebraic operations in RDB. The **major difficulty** of the ECJ lies in avoiding values in two tables, say, I and II being compared *unlimitedly* even when there exists a table III such that values in Table I and III are comparable (by user's request) and values in Table II and III are comparable (by user's request). Obviously a value X in Table I and a value Y in Table II are comparable if there exists either X or Y in Table III, but comparison should be possible only to that extent. In other words, equivalence check needs to be **non-transitive**.

As a concrete scheme that realizes our ECJ, we also proposes a scheme by using asymmetric bilinear group under a novel but natural assumption. The use of bilinear groups imposes users a rather heavy pairing computation, but is necessary to avoid the unlimited comparison. A proxy-reencryptions [8,27,4] also depend on bilinear groups if they avoid unrestricted transitivity for a similar reason. If unlimited comparison is allowed, a simple exponentiation serves well[2]. But we consider this privacy gain by the heavy pairing computation is beneficial in some applications since it can avoid complex and fallible policy checking that decides who (the user of the DB) joins the tables. We estimate its efficiency and conclude that the scheme achieves practical efficiency for some applications. The proof of its security is in the random oracle model.

1.3 Organization

Section 2 introduces the formal model of ECJ, that is, its algorithms and security requirements. Section 3 presents a concrete scheme of ECJ and consider its efficiency. Section 4 analyses the security of the proposed scheme. Section 5 concludes the paper and poses an open problem.

[2] Let value x in column A be encrypted as $x^{\mathsf{Hash}(A,key)}$ and compare it with values in column B by giving $pjkey = \mathsf{Hash}(B, key)/\mathsf{Hash}(A, key)$ and generating $x^{\mathsf{Hash}(B,key)} = (x^{\mathsf{Hash}(A,key)})^{pjkey}$.

2 The Formal Model of ECJ

2.1 Algorithms

The algorithms for ECJ are given as in the following. Here, $\{lab, lab'\}$ denotes the set of lab and lab'. Since both notations $\{lab, lab'\}$ and $\{lab', lab\}$ indicate the same set, they are encoded into the same code when they are given to algorithms as an input.

KeyGen**:** An algorithm for DB users that, given a **security parameter** $\kappa \in \mathbb{N}$, generates a **master key** $mkey$ and a **system parameter** $param$ as $(mkey, param) \leftarrow \mathsf{KeyGen}(\kappa)$. The system parameter defines the spaces of labels, plaintexts, ciphertexts, etc.

Enc**:** An algorithm for users that, given $param$, $mkey$, a **label** lab, a **plaintext** msg, outputs a **ciphertext** $ciph$ as $ciph \leftarrow \mathsf{Enc}(param, mkey, lab, msg)$.

Dec**:** An algorithm for users that, given $param$, $mkey$, and a ciphertext $ciph$, outputs a plaintext msg as $msg \leftarrow \mathsf{Dec}(param, mkey, ciph)$.

ProKeyGen**:** An algorithm for users that, given $param$, $mkey$, and a set of two labels $\{lab_1, lab_2\}$, outputs **projection key** $pjkey$ as $pjkey \leftarrow \mathsf{ProKeyGen}(param, mkey, \{lab_1, lab_2\})$.

Project: An algorithm for the DB manager that, given $param$, $pjkey$, a set of two labels $\{lab_1, lab_2\}$, a ciphertext $ciph$, and a label $lab \in \{lab_1, lab_2\}$, outputs **comparison value** cv as $cv \leftarrow \mathsf{Project}(param, pjkey, \{lab_1, lab_2\}, ciph, lab)$. (Here, $ciph$ is supposed to be encrypted with respect to lab.)

2.2 Security Requirements

Definition 1. *An ECJ is* **complete** *if the following two conditions are satisfied.*

Condition 1: *For every κ, lab, msg, the followings hold;*

$$(mkey, param) \leftarrow \mathsf{KeyGen}(\kappa), ciph \leftarrow \mathsf{Enc}(param, mkey, lab, msg)$$
$$msg \leftarrow \mathsf{Dec}(param, mkey, ciph),$$

Condition 2: *It is computationally difficult (with respect to the security parameter κ) to find lab, lab', msg, msg' such the followings hold;*

$$(mkey, param) \leftarrow \mathsf{KeyGen}(\kappa), ciph \leftarrow \mathsf{Enc}(param, mkey, lab, msg)$$
$$ciph' \leftarrow \mathsf{Enc}(param, mkey, lab', msg'), pjkey \leftarrow \mathsf{ProKeyGen}(param, mkey, \{lab, lab'\})$$
$$cv \leftarrow \mathsf{Project}(param, pjkey, \{lab, lab'\}, ciph, lab),$$
$$cv' \leftarrow \mathsf{Project}(param, pjkey, \{lab', lab\}, ciph', lab')$$
$$\big((cv \neq cv') \wedge (msg = msg')\big) \vee \big((cv = cv') \wedge (msg \neq msg')\big)$$

Since the completeness guarantees that the comparison values generated from the two ciphertexts are the same if and only if the ciphertexts are of the same

message, the DB manager is able to compare whether or not two ciphertexts are of the same message.

We next define indistinguishability of ECJ. As a preliminary, we define a distinguishing game and the validness.

Definition 2. *The* **distinguishing game** *is played between challenger C and adversary A as in the following. It begins when C is given $\kappa \in \mathbb{N}$, runs $(mkey, param) \leftarrow \mathsf{KeyGen}(\kappa)$, and gives param to A. C randomly chooses $b \in \{0,1\}$ and responds to queries from A as in the following.*

- *When C receives $(\mathsf{encrypt}, lab, msg)$, it returns $ciph = \mathsf{Enc}(param, mkey, lab, msg)$ to A.*
- *When C receives $(\mathsf{prokey}, \{lab, lab'\})$, it returns $pjkey = \mathsf{ProKeyGen}(param, mkey, \{lab, lab'\})$ to A.*
- *C receives $(\mathsf{target}, msg_0^*, msg_1^*)$ such that $|msg_0^*| = |msg_1^*|$ only once in the game.*
- *When C receives (test, lab) after C has received $(\mathsf{target}, msg_0^*, msg_1^*)$, it returns $ciph = \mathsf{Enc}(param, mkey, lab, msg_b^*)$ to A.*

At the end of the game, A sends $b' \in \{0,1\}$ to C. The result of the game $\mathsf{Exp}_{C,A}^{\kappa}$ is 1 if $b = b'$; otherwise 0.

A is allowed to query (test, lab) for multiple times for various lab's. This is natural since the same value can be stored in many tables. The game considers only of type "chosen plaintext attacks" but not "chosen ciphertext attacks". Consequently, the Definitions 4 and 6 presented below do not consider chosen ciphertext attacks, which is stronger and commonly considered. However, encrypt-then-MAC [6] generic construction can easily make the scheme resistant for them.

The distinguishing game challenges the adversary's ability to distinguish ciphertexts. However, if a certain set of queries is sent to the challenger, it is inevitable to prevent rational adversaries from distinguishing ciphertexts. Hence, the cases and only the case when such queries are sent needs to be excluded to measures the strength of ECJ scheme. For this purpose we introduce the notion of validness.

Definition 3. *Let $\mathcal{L}^*$ denote the set of all lab's such that a query (test, lab) exists. Let $\bar{\mathcal{L}}$ be the set of all lab's such that there exists a query $(\mathsf{encrypt}, lab, msg_\beta^*)$ for some $\beta \in \{0,1\}$. We say a distinguishing game is* **valid** *if, for every query $(\mathsf{prokey}, \{lab, lab^*\})$ by the challenger, it holds that $\{lab, lab^*\} \cap \bar{\mathcal{L}} = \emptyset$ or $\{lab, lab^*\} \cap \mathcal{L}^* = \emptyset$.*

If the adversary asks $(\mathsf{prokey}, \{lab, lab^*\})$ such that $lab \in \bar{\mathcal{L}}$ and $lab^* \in \mathcal{L}^*$, the adversary is able to directly compare a known message $msg_{\beta\in\{0,1\}}^*$ (β is known also) encrypted with respect to lab and the unknown test message msg_b^* encrypted with respect to lab^*. If the adversary asks $(\mathsf{prokey}, \{lab, lab^*\})$ such that $lab \in \bar{\mathcal{L}} \cap \mathcal{L}^*$ for an arbitrary lab^*, the adversary is able to directly compare a known message $msg_{\beta\in\{0,1\}}^*$ encrypted with respect to lab and the unknown test message msg_b^* encrypted with respect to lab. These are cases in which the adversary can trivially check the value of the test message msg_b^* as long as the

scheme is complete. That is, it does not make sense to require the indistinguishability in these cases. Excluding these cases from valid games yields our notion of validness.

Definition 4. *We say that an ECJ is* **indistinguishable** *if, for every polynomial time adversary* A^*, $\mathsf{Adv}^{\kappa}_{C,A^*} := |\Pr[\mathsf{Exp}^{\kappa}_{C,A^*} = 0] - \Pr[\mathsf{Exp}^{\kappa}_{C,A^*} = 1]|$ *is negligible with respect to* κ *in valid games.*

Definition 5. Selective distinguishing game *is the exactly the same as the game defined in Definition 2 except that* A *sends two sets of labels* $\mathcal{L}^*$, $\bar{\mathcal{L}}$ *to* C *before the game begins.*

Definition 6. *We say that an ECJ is* **selectively indistinguishable** *if it is indistinguishable as Defined in 4 but with the selective distinguishing game*

Definition 6 assumes that two sets of labels are given to the challenger in advance, which is not a realistic scenario for attacks. It is quite easy to construct a scheme that is secure under such an assumption but is totally not secure without it. This situation is very similar to that given in [14]. Here, it is shown that one can easily construct a scheme which is proven to be secure in the random oracle model but is totally vulnerable with any real hash function. In this sense, the security that is guaranteed by Definition 6 is only heuristic just as that guaranteed by the random oracle model.

2.3 Complement to Security Requirements

Since the encryption is probabilistic, neither the data themselves nor relations is recognized by the DB unless projection keys are given. However, once a projection key is given, relevant ciphertexts are no longer probabilistic. Such property seems to be a weakness of our model. However, we consider this model still has a significance as justified in the following.

We suppose that honest managers of DB systems erase each query of ECJ after responding to this query. Then, as long as intrusions of an adversary is instantaneous, the chance for the adversary to obtain SQL queries may be little while it may obtain a large portion of data in the DB. If the adversary obtains no query, the data that the adversary obtained from the DB are only the ciphertexts of semantically secure encryption scheme. Thus, in such a case, the confidentiality of data in the DBs is strongly protected. The adversary may obtains several queries during it is intruding into the DB. However, if what the adversary can obtain from these queries and encrypted data are no more than what the DB manager needs for required relational algebraic operation, we can consider the leakage is the minimum. Here, we particularly concern whether the combination of queries may leak more than the sum of leakage of each query.

Suppose that we want to join two tables of size n and m. Then, unless the data in these tables are encrypted in deterministic way so as the data in them are directly comparable[3], the required computational cost for their join is at least

[3] We say two values are directly comparable if no operation other than comparison of two numerical values is required for the comparison.

$\mathcal{O}(nm)$. This cost is unacceptably high for the most of practical DB systems. Hence, we allow the DB to directly check equivalence of the indices. At the cost of introducing this functionality, some knowledge about data structure is leaked to the DB. However, we require this direct comparison is limited to the minimum in the sense that the comparison is possible only within the tables with respect to which the query is generated. Therefore, we consider that the amount of this knowledge leaked to the DB in our scheme is the minimum and acceptable for the DB to maintain its serviceability.

In case such a minimum leakage is not allowed, users should not use ECJ in its query. But they should simply retrieve the entire encrypted tables, decrypt them, and join them with in their system. We also note that the DB is able to join two tables A and B to obtain a table C, and is still able to join tables C and D if it is requested to do so. This is possible by letting the column of table C, with respect to which join is executed, to inherit label of either A or B. Hence, any number of consecutive joins is possible.

3 Proposed ECJ Scheme

3.1 Asymmetric Bilinear Groups and Preliminary

Let $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T$ be cyclic groups of order prime p such that an efficiently computable bilinear map $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ and homomorphism $\sigma : \mathcal{G}_2 \to \mathcal{G}_1$ exist and that the decision Diffie-Hellman problems in $\mathcal{G}_1$ and $\mathcal{G}_T$ are infeasible to solve in polynomial time. Note that, in contrast, the decision Diffie-Hellman problems in $\mathcal{G}_2$ is easy because of the map σ and e. Elliptic curves introduced in [30] (MNT curves) are considered to satisfy these properties.

Let κ be a security parameter and $\mathcal{IV} = \{0,1\}^\kappa$ be a space of initial vectors. Initial vectors are used to label columns. Let $(\mathsf{enc}, \mathsf{dec})$ be a symmetric-key cryptosystem. For each security parameter κ, it specifies $\mathcal{K} = \{0,1\}^\kappa$ and $\mathcal{M}' = \mathcal{C} = \{0,1\}^*$ which are, respectively, its key space, message space, and ciphertext space. Functions are such that $\mathsf{enc} : \mathcal{K} \times \mathcal{M}' \to \mathcal{C}$ and $\mathsf{dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}'$. Let $\mathcal{M}$ be such that $\mathcal{M}' = \mathcal{IV} \times \mathcal{M}$. For a space $\mathcal{R}$, let $\mathsf{Hash}_\mathcal{R}$ be a cryptographic hash function $\mathsf{Hash}_\mathcal{R} : \{0,1\}^* \to \mathcal{R}$. We assume $\mathsf{Hash}_\mathcal{R}$ and $\mathsf{Hash}_{\mathcal{R}'}$ are independent if $\mathcal{R} \neq \mathcal{R}'$.

The decryption algorithm Dec is trivial and is not necessary for joining procedure. Hence, we may omit to consider it. However, we do present it so as to comfortably call our proposal an encryption scheme.

3.2 Scheme

KeyGen: Given a security parameter $\kappa \in \mathbb{N}$, KeyGen specifies a prime q of size polynomial of κ, order q cyclic groups $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T$, a generator g_1 of $\mathcal{G}_1$, a generator g_2 of $\mathcal{G}_2$, a symmetric-key encryption scheme $(\mathsf{enc}, \mathsf{dec})$, and hash functions $\mathsf{Hash}_{\mathbb{Z}_q}$, $\mathsf{Hash}_\mathcal{K}$, $\mathsf{Hash}_{\mathcal{G}_1}$. Let *param* denotes the parameter that specifies the above. KeyGen randomly chooses a master key $mkey \in \{0,1\}^\kappa$ and outputs *mkey* and *param*.

Enc: Given a system parameter $param$, a master key $mkey$, a label $lab \in \mathcal{IV}$, and a message $msg \in \mathcal{M}$, Enc randomly chooses $r_1, r_2 \in \mathcal{IV}$ and generates

$$f = \mathsf{Hash}_{\mathbb{Z}_q}(1, param, mkey, msg), x = \mathsf{Hash}_{\mathbb{Z}_q}(2, param, mkey, lab)$$
$$h = {g_1}^{xf}, c = \mathsf{Hash}_{\mathcal{G}_1}(\mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab), r_1) \cdot h$$
$$d = \mathsf{enc}(\mathsf{Hash}_{\mathcal{K}}(2, param, mkey), (r_2, msg)).$$

Then, Enc outputs ciphertext $ciph = (r_1, c, d)$.

Dec: Given a master key $mkey$ and a ciphertext $ciph = (r_1, c, d)$, Dec generates $(r_2, msg) = \mathsf{dec}(\mathsf{Hash}_{\mathcal{K}}(2, param, mkey), d)$ and outputs msg.

ProKeyGen: Given a system parameter $param$, a master key $mkey$, and a set of two labels $\{lab_1, lab_2\}$, ProKeyGen generates

$$k_1 = \mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab_1), k_2 = \mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab_2)$$
$$x_1 = \mathsf{Hash}_{\mathbb{Z}_q}(2, param, mkey, lab_1), x_2 = \mathsf{Hash}_{\mathbb{Z}_q}(2, param, mkey, lab_2)$$
$$z_1 = {g_2}^{x_1}, z_2 = {g_2}^{x_2}, p = \mathsf{Hash}_{\mathbb{Z}_q}(3, param, mkey, \{lab_1, lab_2\})$$
$$w_1 = {z_2}^{p}\ , \quad w_2 = {z_1}^{p}$$

and outputs projection key $pjkey = \{(lab_1, k_1, w_1), (lab_2, k_2, w_2)\}$.

Project: Given a system parameter $param$, a set of two labels $\{lab_1, lab_2\}$, a ciphertext $ciph = (r_1, c, d)$, a projection key $pjkey = \{(lab_1, k_1, w_2), (lab_2, k_2, w_2)\}$, and a label lab_ℓ for $\ell \in \{1, 2\}$, Project generates $h' = c \cdot \mathsf{Hash}_{\mathcal{G}_1}(k_\ell, r_1)^{-1}$, $cv = e(h', w_\ell)$ and outputs cv.

3.3 Completeness

How and why the proposed scheme works is described in the proof of the following theorem.

Theorem 1. *The proposed scheme is complete*

Proof. Condition 1 holds from the fact that $(\mathsf{enc}, \mathsf{dec})$ is a symmetric-key encryption scheme. That means encrypted data can correctly be decrypted.

Condition 2 holds as shown below. Let

$$f = \mathsf{Hash}_{\mathbb{Z}_q}(1, param, mkey, msg), f' = \mathsf{Hash}_{\mathbb{Z}_q}(1, param, mkey, msg')$$
$$x = \mathsf{Hash}_{\mathbb{Z}_q}(2, param, mkey, lab), h = {g_1}^{xf}, x' = \mathsf{Hash}_{\mathbb{Z}_q}(2, param, mkey, lab'),$$
$$h' = {g_1}^{x'f'}, z = {g_2}^{x}, z' = {g_2}^{x'}, p = \mathsf{Hash}_{\mathbb{Z}_q}(3, param, mkey, \{lab, lab'\}), w = {z'}^{p}, w' = z^{p}.$$

Then, it holds that

$$\mathsf{Project}(param, pjkey, \{lab', lab\}, ciph, lab) = e(h, w) = e({g_1}^{xf}, {z'}^{p}) = e({g_1}^{x}, z')^{fp}$$
$$= e({g_1}^{x}, {g_2}^{x'})^{fp} \overset{\text{iff } f=f'}{=} e({g_1}^{x'}, {g_2}^{x})^{f'p} = e({g_1}^{x'}, z)^{f'p} = e({g_1}^{x'f'}, z^{p}) = e(h', w')$$
$$= \mathsf{Project}(param, pjkey, \{lab, lab'\}, ciph', lab').$$

With respect to 5th equation, it is computationally difficult find a pair of msg and msg' such that $(f = f') \wedge (msg \neq msg')$. Thus, the theorem follows.

3.4 Efficiency

Pairing is considered to be a very heavy operation even among asymmetric key cryptographic operations. In our scheme, this pairing is absolutely the dominant time consuming operation. However, recent efforts have greatly enhanced its efficiency. J. -L. Beuchat et al. [7] reported an implementation of pairing (ate pairing) over a 254-bit prime field in just 2.33 million of clock cycles on a single core of an Intel Core i7 2.8GHz processor. This implies that the pairing computation takes only 0.832msec. If we join two tables of size 100,000 rows, it is estimated that it takes 10 seconds with 16 cores. This is not very fast though a large resources are consumed, but we consider it acceptable time for handling highly confidential data in many cases.

4 Security

The security of ours scheme depends on the new assumption that we introduce below.

Assumption 1. (Chained Decision Bilinear Diffie-Hellman Assumption)
Let $x_1, x_2, x_3, \alpha, \beta, \gamma, \delta \in \mathbb{Z}_q$ *and* $y_i = {g_1}^{x_i}, z_i = {g_2}^{x_i}$ *for* $i = 1, 2, 3$. *We also let*

$$W := \begin{pmatrix} w_1 & w_2 & w_3 & \\ w_4 & w_5 & & \\ w_6 & & w_7 & \\ & w_8 & w_9 & \end{pmatrix} = \begin{pmatrix} y_1 & y_2 & y_3 & \\ {y_1}^{\gamma} & {y_2}^{\delta} & & \\ {z_1}^{\alpha} & & {z_3}^{\alpha} & \\ & {z_2}^{\beta} & {z_3}^{\beta} & \end{pmatrix}.$$

For every polynomial time adversary A *that is given* W, *the probability, that* A *distinguishes whether* $x_1, x_2, x_3, \alpha, \beta, \gamma, \delta \in \mathbb{Z}_q$ *are randomly and independently chosen or they are so except with the restriction* $\delta = \gamma$, *is negligible in* κ.

Roughly, Assumption 1 is a stronger variant of decisional Diffie-Hellman assumption. It claims that a deciding whether $(w_1, w_2, w_4, w_5) \in {\mathcal{G}_1}^4$ is a Diffie-Hellman tuple or not ($\gamma = \delta$ or not) is infeasible even if w_6, w_7, w_8, and w_9 are given. By contrast, deciding whether $y_1, y_3, {y_1}^{\gamma}, {y_3}^{\epsilon}$ is a Diffie-Hellman tuple or not ($\gamma = \epsilon$ or not) is easy because of w_6 and w_7. We only needs to check $e({y_1}^{\gamma}, w_7) \stackrel{?}{=} e({y_3}^{\epsilon}, w_6)$. Deciding whether $y_2, y_3, {y_2}^{\delta}, {y_3}^{\epsilon}$ is a Diffie-Hellman tuple or not ($\delta = \epsilon$ or not) is also easy because of w_8 and w_9.

Note that deciding whether or not (w_1, w_2, w_4, w_5) is a Diffie-Hellman is always easy if $\mathcal{G}_1, \mathcal{G}_2$ are supersingular curve. Hence, we need an ordinary elliptic curve with pairing that forms asymmetric bilinear groups such as MNT curves [30].

Theorem 2. *Assumption 1 is valid in the generic asymmetric bilinear groups.*

Proof. The proof is given in Appendix A.

Intuition for the indistinguishability of the proposed scheme is as follows. Suppose that there are three labels lab_1, lab_2, and lab_3 and that lab_i corresponds to y_i and z_i for each $i = 1, 2, 3$. In our scheme, distinguishing ciphertexts with respect to lab_1 and lab_2 corresponds to deciding whether or not $(y_1, y_2, {y_1}^{\gamma}, {y_2}^{\delta})$ is a Diffie-Hellman tuple ($\delta = \gamma$ or not). Similarly, distinguishing ciphertexts with respect to lab_1 and lab_3 corresponds to deciding whether $y_1, y_3, {y_1}^{\gamma}, {y_3}^{\epsilon}$ is a Diffie-Hellman tuple or not ($\gamma = \epsilon$ or not), and distinguishing ciphertexts with respect to lab_2 and lab_3 corresponds to deciding whether $y_2, y_3, {y_2}^{\delta}, {y_3}^{\epsilon}$ is a Diffie-Hellman tuple or not ($\delta = \epsilon$ or not). The projection key with respect to $\{lab_1, lab_3\}$ and that with respect to $\{lab_2, lab_3\}$, respectively, correspond to (w_6, w_7) and (w_8, w_9). Now, given these projection keys, while distinguishing ciphertexts with respect to lab_1 and lab_3 and that with respect to lab_2 and lab_3 are easy, that with respect to lab_1 and lab_2 is difficult. And the adversary's goal is distinguishing ciphertexts with respect to lab_1 and lab_2.

Theorem 3. *The proposed scheme is selectively indistinguishable under Assumption 1 in the random oracle model.*

Proof. The proof is by contraposition. Suppose that there exists an adversary A^* such that $\mathsf{Adv}^{\kappa}_{C,A^*} := |\Pr[\mathsf{Exp}^{\kappa}_{C,A^*} = 0] - \Pr[\mathsf{Exp}^{\kappa}_{C,A^*} = 1]|$ is non negligible in κ. We shows that the existence of A^* contradicts to the Assumption 1. In particular, the contradiction follows by the hybrid argument from the lemmas 1, 2, 3, 4, 5, and 6 with respect to the sequence of games by challengers C_1, C_2, C_3, C_4, and C_5. Proofs of lemmas 1, 2, 3, 4, and 6 are given in Appendix B.

Definition 7. *Challenger C_1 is the same as the challenger C in Definition 2 except in the following:*

- *C_1 prepares tables for the random oracles $\mathsf{Hash}_{\mathcal{K}}, \mathsf{Hash}_{\mathbb{Z}_q}$ and use them to simulate them. That is, when the input to the random oracles is not on the tables, C_1 chooses random number from $\mathcal{K}$ or $\mathbb{Z}_q$, returns it as the output, and writes the pair of the input and the output on the table. Otherwise, it returns the corresponding output that exists on the table.*
- *If A^* sends mkey to the random oracles $\mathsf{Hash}_{\mathcal{K}}$ or $\mathsf{Hash}_{\mathbb{Z}_q}$, aborts the game.*

Lemma 1. *For every polynomial time A^*, $|\mathsf{Adv}^{\kappa}_{C_1,A^*} - \mathsf{Adv}^{\kappa}_{C,A^*}|$ is negligible in κ.*

Definition 8. *Challenger C_2 is the same as the challenger C_1 except in the following:*

- *Let R_2^* be the set of all r_2^* that C_2 used in encryption procedures such as $\mathsf{enc}(\mathsf{Hash}_{\mathcal{K}}(2, param, mkey), (r_2^*, msg_b^*))$ for responding queries $(\mathsf{test}, \cdot)$. Let R_2 be the set of all r_2 that C_2 used in encryption procedures such as $\mathsf{enc}(\mathsf{Hash}_{\mathcal{K}}(2, param, mkey), (r_2, msg))$ for responding queries $(\mathsf{encrypt}, lab, msg)$. If $R_2 \cap R_2^* \neq \emptyset$, C_2 aborts the game.*

Lemma 2. *For every polynomial time A^*, $|\mathsf{Adv}^{\kappa}_{C_2,A^*} - \mathsf{Adv}^{\kappa}_{C_1,A^*}|$ is negligible in κ.*

Definition 9. *Challenger C_3 is the same as the challenger C_2 except in the following:*

- *Suppose that the original answer to a query (test, lab) is $ciph = (r_1^*, c^*, d^*)$. With appropriate K and r_2^*, C_3 replaces d^* with $d^\dagger := \mathsf{enc}(K, (r_2^*, msg^\dagger))$ where $msg^\dagger$ is randomly chosen string such that $|msg^\dagger| = |msg_b^*|$ with appropriate K and r_2^*.*
- *Suppose that the original answer to a query $(\mathsf{encrypt}, lab, msg)$ is $ciph = (r_1^*, c^*, d^*)$. With appropriate K and r_2^*, C_3 replaces d^* with $d^\dagger := \mathsf{enc}(K, (r_2^*, msg^\dagger))$ where $msg^\dagger$ is randomly chosen string such that $|msg^\dagger| = |msg|$.*

Lemma 3. *For every polynomial time A^*, $|\mathsf{Adv}^\kappa_{C_3,A^*} - \mathsf{Adv}^\kappa_{C_2,A^*}|$ is negligible in κ.*

Definition 10. *Challenger C_4 is the same as the challenger C_3 except in the following:*

- *When C_4 receives $(\mathsf{encrypt}, lab^\dagger, msg_\beta^*)$ for some $\beta \in \{0,1\}$ and $lab^\dagger \in \mathcal{L}^* \cap \bar{\mathcal{L}}$, C_4 answers it as if C_4 received $(\mathsf{encrypt}, lab^\dagger, msg^\dagger)$ with randomly chosen $msg^\dagger$ such that $|msg^\dagger| = |msg_\beta^*|$.*

Lemma 4. *For every polynomial time A^*, $|\mathsf{Adv}^\kappa_{C_4,A^*} - \mathsf{Adv}^\kappa_{C_3,A^*}|$ is negligible in κ.*

Definition 11. *Challenger C_5 is the same as the challenger C_4 except in the following:*

- *Let $ciph = (r_1^*, c^*, d^\dagger)$ be the response to (test, lab). Suppose that part of this response is generated as in the following.*

$$r_1^*, r_2^* \in_R \mathbb{Z}_q, f^* = \mathsf{Hash}_{\mathbb{Z}_q}(6, param, mkey), x^* = \mathsf{Hash}_{\mathbb{Z}_q}(2, param, mkey, lab)$$
$$h^* = {g_1}^{x^* f^*}, c^* = \mathsf{Hash}_{\mathcal{G}_1}(\mathsf{Hash}_{\mathcal{K}}(1, param, mkey^*, lab^*), r_1^*) \cdot h^*$$

If lab is the first to appear, C_5 randomly chooses $h^\dagger \in \mathcal{G}_1$ and replaces h^* with it. If lab has appeared before, use the same $h^\dagger$ to replace h^* with.

Lemma 5. *For every polynomial time A^*, $|\mathsf{Adv}^\kappa_{C_5,A^*} - \mathsf{Adv}^\kappa_{C_4,A^*}|$ is negligible in κ under Assumption 1.*

Proof. We prove that if $|\mathsf{Adv}^\kappa_{C_5,A^*} - \mathsf{Adv}^\kappa_{C_4,A^*}|$ is non negligible, an adversary S that breaks Assumption 1 can be constructed from A^*. Suppose that S is given

$$W = \begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & \\ w_6 & & w_7 \\ & w_8 & w_9 \end{pmatrix}.$$

Then, S simulates the game and distinguishes from which distribution W is chosen by using the guess of A^* as follows. When A^* sends $(1, param, mkey, msg)$

to the random oracle $\mathsf{Hash}_{\mathbb{Z}_q}$ for some msg, S guess whether or not $msg \in \{msg_0^*, msg_1^*\}$. When A^* sends $(\mathsf{encrypt}, lab, msg)$ to S for some msg, S guess whether or not $msg \in \{msg_0^*, msg_1^*\}$ also. When S receives $(\mathsf{target}, msg_0^*, msg_1^*)$, whether or not these guesses was correct becomes evident. And after that, the S is able to guess them correctly. Since the number of messages, including those in $(\mathsf{target}, msg_0^*, msg_1^*)$, that appear in the game are of polynomial, S is able to guess them correctly in some reasonable way with non negligible probability.

In what follows, we assume S succeeded in these guessing. We describe how S responds to A^* for randomly chosen $b \in \{0, 1\}$.

- S first generates $param$ appropriately and gives it to A^*. S generates $mkey$ randomly.
- When S receives $(\mathsf{encrypt}, lab, msg)$, it generates $ciph = (r_1, c, d^\dagger)$ and returns it to A as follows. First, S generates

$$\alpha = \mathsf{Hash}_{\mathbb{Z}_q}(4, param, mkey, lab), \quad f = \mathsf{Hash}_{\mathbb{Z}_q}(1, param, mkey, msg),$$
$$r_1, r_2 \in_R \mathbb{Z}_q, \quad K_{lab} = \mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab),$$
$$K = \mathsf{Hash}_{\mathcal{K}}(2, param, mkey),$$
$$msg^\dagger \in_R \{msg' | msg' \in \mathcal{M} \wedge |msg'| = |msg|\}.$$

  Here, hash functions are random oracles that S simulates.
  - In case $lab \in \mathcal{L}^* \cap \bar{\mathcal{L}}$, S generates

$$x = \mathsf{Hash}_{\mathbb{Z}_q}(2, param, mkey, lab), \qquad h = {g_1}^{x \cdot f}$$
$$c = \mathsf{Hash}_{\mathcal{G}_1}(K_{lab}, r_1) \cdot h, \qquad d^\dagger = \mathsf{enc}(K, r_2, msg^\dagger).$$

  - In case $lab \notin \mathcal{L}^* \cup \bar{\mathcal{L}}$, where $msg \notin \{msg_0^*, msg_1^*\}$ by definition, S generates,

$$h = {w_3}^{\alpha \cdot f}, \quad c = \mathsf{Hash}_{\mathcal{G}_1} c(K_{lab}, r_1) \cdot h, \quad d^\dagger = \mathsf{enc}(K, r_2, msg^\dagger).$$

  - In case $lab \in \bar{\mathcal{L}} \setminus \mathcal{L}^*$;
    - If $msg \neq msg_b^*$, S generates

$$h = {w_2}^{\alpha \cdot f}, \quad c = \mathsf{Hash}_{\mathcal{G}_1}(K_{lab}, r_1) \cdot h, \quad d^\dagger = \mathsf{enc}(K, r_2, msg^\dagger).$$

    - If $msg = msg_b^*$, S generates

$$h = {w_5}^{\alpha}, \quad c = \mathsf{Hash}_{\mathcal{G}_1}(K_{lab}, r_1) \cdot h, \quad d^\dagger = \mathsf{enc}(K, r_2, msg^\dagger).$$

  - In case $lab \in \mathcal{L}^* \setminus \bar{\mathcal{L}}$, where $msg \neq msg_b^*$ by definition, S generates

$$h = {w_1}^{\alpha \cdot f}, \quad c = \mathsf{Hash}_{\mathcal{G}_1}(K_{lab}, r_1) \cdot h, \quad d^\dagger = \mathsf{enc}(K, r_2, msg^\dagger).$$

- When S receives (test, lab), it generates $ciph = (r_1, c, d^\dagger)$ and returns it to A as follows.

$$msg^\dagger \in_R \{msg' | msg' \in \mathcal{M} \wedge |msg'| = |msg|\}$$
$$K_{lab} = \mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab), \qquad K = \mathsf{Hash}_{\mathcal{K}}(2, param, mkey)$$
$$\alpha = \mathsf{Hash}_{\mathbb{Z}_q}(4, param, mkey, lab), \qquad h = {w_4}^{\alpha},$$
$$c = \mathsf{Hash}_{\mathcal{G}_1}(K_{lab}, r_1) \cdot h^*, \qquad d^\dagger = \mathsf{enc}(K, r_2, msg^\dagger)$$

– When S receives $(\mathsf{prokey}, \{lab, lab'\})$, it generates $pjkey$ and returns it to A as follows. Since the case when lab and lab' are exchanged are clear, we only gives one. If a set $Q \in \mathcal{IV}$, we letQ^c denotes $\mathcal{IV} \setminus Q$.
First, S generates

$$\alpha = \mathsf{Hash}_{\mathbb{Z}_q}(4, param, mkey, lab), \qquad \alpha' = \mathsf{Hash}_{\mathbb{Z}_q}(4, param, mkey, lab')$$
$$p = \mathsf{Hash}_{\mathbb{Z}_q}(3, param, mkey, \{lab, lab'\}), \quad K_{lab} = \mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab)$$
$$K = \mathsf{Hash}_{\mathcal{K}}(2, param, mkey).$$

Next;

- In case $(lab, lab') \in (\mathcal{L}^* \setminus \bar{\mathcal{L}}) \times (\mathcal{L}^* \cup \bar{\mathcal{L}})^c$, S generates;

$$w = {w_7}^{\alpha' p}, w' = {w_6}^{\alpha p}.$$

- In case $(lab, lab') \in (\bar{\mathcal{L}} \setminus \mathcal{L}^*) \times (\mathcal{L}^* \cup \bar{\mathcal{L}})^c$, S generates;

$$w = {w_9}^{\alpha' p}, w' = {w_8}^{\alpha p}.$$

- In case $(lab, lab') \in (\bar{\mathcal{L}} \setminus \mathcal{L}^*) \times (\bar{\mathcal{L}} \setminus \mathcal{L}^*)$, S generates;

$$w = {w_8}^{\alpha' p}, w' = {w_8}^{\alpha p}.$$

- In case $(lab, lab') \in (\mathcal{L}^* \setminus \bar{\mathcal{L}}) \times (\mathcal{L}^* \setminus \bar{\mathcal{L}})$, S generates;

$$w = {w_9}^{\alpha' p}, w' = {w_9}^{\alpha p}.$$

- In case $(lab, lab') \in (\mathcal{L}^* \cup \bar{\mathcal{L}})^c \times (\mathcal{L}^* \cup \bar{\mathcal{L}})^c$, S generates;

$$w = {w_6}^{\alpha' p}, w' = {w_6}^{\alpha p}.$$

S assigns $pjkey = \{(lab, k, w), (lab', k', w')\}$.

The distribution of the view of the simulated game by S is exactly the same as that by C_4 if W is such that $x_1, x_2, x_3, \alpha, \beta, \gamma, \delta \in \mathbb{Z}_q$ are randomly and independently chosen. On the other hand, the distribution is exactly the same as that by C_5 if W is such that $x_1, x_2, x_3, \alpha, \beta, \gamma, \delta \in \mathbb{Z}_q$ are randomly and independently chosen with the restriction $\delta = \gamma$. Therefore, if the advantages of A^* in games C_4 and C_5 differ with non negligible probability, S is able to distinguish from which distribution W is chosen. In particular, S outputs 1 if the output of A^* coincides with b but 0 otherwise. Hence, from Assumption 1, the lemma follows.

Lemma 6. *For every polynomial time A^*, $|\mathsf{Adv}^{\kappa}_{C_5, A^*}|$ is negligible in κ.*

5 Conclusion and Open Problem

We proposed a novel notion of encryption for controlled joining (ECJ) and its instance. The ECJ enables RDB to execute natural join of tables when and only when required by user with reasonable efficiency. The natural join is most frequently used relational algebraic operation in RDB. We instantiated a novel ECJ scheme by using asymmetric pairing. The cost the proposed scheme requires is at most linear to the size of joined tables. Our ECJ also enables RDB to efficiently execute relational algebraic operations such as union, difference, and intersection for tables.

It it an open problem to propose a scheme that is provably secure in non selective model.We consider dedicated hardware accelerators are still necessary for the state-of-the-art ECJ to be applied to practical RDB. More efficient schemes are strongly desired.

References

1. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed architecture for secure database services. In: CIDR, pp. 186–199 (2005)
2. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order-preserving encryption for numeric data. In: Weikum, G., König, A.C., Deßloch, S. (eds.) SIGMOD Conference, pp. 563–574. ACM (2004)
3. Amanatidis, G., Boldyreva, A., O'Neill, A.: Provably-Secure Schemes for Basic Query Support in Outsourced Databases. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 14–30. Springer, Heidelberg (2007)
4. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: NDSS. The Internet Society (2005)
5. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and Efficiently Searchable Encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
6. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
7. Beuchat, J.-L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
8. Blaze, M., Bleumer, G., Strauss, M.J.: Divertible Protocols and Atomic Proxy Cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
9. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-Preserving Symmetric Encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009)
10. Boldyreva, A., Chenette, N., O'Neill, A.: Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 578–595. Springer, Heidelberg (2011)

11. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
12. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
13. Canard, S., Fuchsbauer, G., Gouget, A., Laguillaumie, F.: Plaintext-Checkable Encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 332–348. Springer, Heidelberg (2012)
14. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
15. Ceselli, A., Damiani, E., di Vimercati, S.D.C., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM Trans. Inf. Syst. Secur. 8(1), 119–152 (2005)
16. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS, pp. 41–50 (1995)
17. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
18. CipherGate. eciphergate, http://www.ciphergate.co.jp/product/ecipher/index.html
19. Ciriani, V., di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Trans. Inf. Syst. Secur. 13(3) (2010)
20. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (1970)
21. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 79–88. ACM (2006)
22. di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Controlled information sharing in collaborative distributed query processing. In: ICDCS, pp. 303–310. IEEE Computer Society (2008)
23. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003), http://eprint.iacr.org/
24. Hacigümüs, H., Iyer, B.R., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: Franklin, M.J., Moon, B., Ailamaki, A. (eds.) SIGMOD Conference, pp. 216–227. ACM (2002)
25. Hacigümüs, H., Mehrotra, S., Iyer, B.R.: Providing database as a service. In: ICDE, p. 29. IEEE Computer Society (2002)
26. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B. (eds.) VLDB, pp. 720–731. Morgan Kaufmann (2004)
27. Jakobsson, M.: On Quorum Controlled Asymmetric Proxy Re-encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 112–121. Springer, Heidelberg (1999)
28. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS, pp. 364–373 (1997)
29. Microsoft. Deploying sql server 2008 based on payment card industry data security standards (2008), http://www.parentebeard.com/Uploads/Files/Deploying_SQL_Server_2008_Based_on_PCI_DSS.PDF

30. Miyaji, A., Nakabayashi, M., Takano, S.: Characterization of Elliptic Curve Traces under FR-Reduction. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 90–108. Springer, Heidelberg (2001)
31. Oracle. Oracle database 11g, oracle advanced security, http://www.oracle.com/technology/global/jp/products/security/db_security/htdocs/aso.html
32. PCI-DSS. Payment card industry (pci) data security standard version 2.0, https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf
33. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: Wobber, T., Druschel, P. (eds.) SOSP, pp. 85–100. ACM (2011)
34. Samarati, P.: Protecting data privacy in outsourcing scenarios: invited talk. In: Daniel, F., Delcambre, L.M.L., Fotouhi, F., Garrigós, I., Guerrini, G., Mazón, J.-N., Mesiti, M., Müller-Feuerstein, S., Trujillo, J., Truta, T.M., Volz, B., Waller, E., Xiong, L., Zimányi, E. (eds.) EDBT/ICDT Workshops. ACM International Conference Proceeding Series. ACM (2010)
35. Samarati, P., di Vimercati, S.D.C.: Data protection in outsourcing scenarios: issues and directions. In: Feng, D., Basin, D.A., Liu, P. (eds.) ASIACCS, pp. 1–14. ACM (2010)
36. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint report 2004/332 (November 30, 2004)
37. Tang, Q.: Towards Public Key Encryption Scheme Supporting Equality Test with Fine-Grained Authorization. In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 389–406. Springer, Heidelberg (2011)
38. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic Public Key Encryption with Equality Test. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119–131. Springer, Heidelberg (2010)

A Proof of Theorem 2

The generic group of asymmetric bilinear group is modeled as follows. Let q be a prime, $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T$ be cyclic groups of order q such that allowed operations are additions in $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T$, bilinear map $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$, homomorphism $\sigma : \mathcal{G}_2 \to \mathcal{G}_1$, and generations of random elements in $\mathcal{G}_1, \mathcal{G}_T$.

Let $(p_1, \ldots, p_5) = (1, x_1, x_2, x_3, x_1\gamma), (q_1, \ldots, q_5) = (1, x_1\alpha, x_3\alpha, x_2\beta, x_3\beta)$ where $x_1, x_2, x_3, \alpha, \beta, \gamma \in \mathbb{Z}_q$ are randomly chosen. Note that these variables represent the exponents of elements, excluding w_5, in W.

Then consider polynomials $D_1(p_6)$ and $D_2(p_6)$ with a set $\mathcal{F}$ of coefficients $\{a_i\}_{i=1,\ldots,6}, \{b_i\}_{i=1,\ldots,5}$, $\{c_{ij}\}_{i=1,\ldots,6;j=1,\ldots,5}, \{d_{ij}\}_{i=1,\ldots,5;j=1,\ldots,5}, e_1$ in $\mathbb{Z}_q$ as follows.

$$D_1(p_6) = \sum_{i=1}^{6} a_i p_i + \sum_{i=1}^{5} b_i q_i \ , \ D_2(p_6) = \sum_{i=1}^{6}\sum_{j=1}^{5} c_{ij} p_i q_j + \sum_{i,j=1}^{5} d_{ij} q_i q_j + e_1.$$

From the logic demonstrated in [11], Assumption 1 holds in the generic asymmetric bilinear group model when the following conditions are satisfied.

1. There exists no Set $\mathcal{F}$ such that $D_1(x_2\gamma) \equiv 0$ but $D_1(x_2\delta) \not\equiv 0$.
2. There exists no Set $\mathcal{F}$ such that $D_2(x_2\gamma) \equiv 0$ but $D_2(x_2\delta) \not\equiv 0$.

Since only equality check can be used for distinction, distinguisher need to computes that is 0 when $\delta = \gamma$ but is not when δ is independently chosen. The condition 1 comes from the equality check in $\mathcal{G}_1$. The condition 2 comes from the equality check in $\mathcal{G}_T$. Since the equality check in $\mathcal{G}_2$ can be done in $\mathcal{G}_1$, these two conditions are sufficient.

Since Condition 1 holds clearly, we focus on proving that Condition 2 holds.

This is done if we show that for every set $\mathcal{F}$ of coefficients such that $c_{6i} = 0$ for some $i = 1, \ldots, 6$, $D(x_2\gamma) \equiv 0$ does not hold.

Focusing on γ, requiring $D(x_2\gamma) \equiv 0$ implies the following.

$$\begin{aligned} 0 \equiv\ & c_{51}x_1 + c_{52}x_1x_1\alpha + c_{53}x_1x_3\alpha + c_{54}x_1x_2\beta + c_{55}x_1x_3\beta \\ & + c_{61}x_2 + c_{62}x_2x_1\alpha + c_{63}x_2x_3\alpha + c_{64}x_2x_2\beta + c_{65}x_2x_3\beta \\ \equiv\ & c_{52}\alpha x_1x_1 + x_1(x_2(c_{62}\alpha + c_{54}\beta) + x_3(c_{53}\alpha + c_{55}\beta) + c_{51}) \\ & + c_{64}\beta x_2x_2 + x_2(x_3(c_{63}\alpha + c_{65}\beta) + c_{61}) \end{aligned}$$

Hence, all coefficients here needs to be 0. Therefore, the theorem follows.

B Proofs of Lemmas 1, 2, 3, 4, and 6

Proof. (Lemma 1) *mkey* appears only as the input to the random oracles. Hence, the game is aborted only when the random guess of *mkey* is successful, which probability is negligible. From the fact that $\mathsf{Adv}^{\kappa}_{C_1,A^*} = \mathsf{Adv}^{\kappa}_{C,A^*}$ as long as the game is not aborted and Lemma 1 (Difference Lemma) in [36], the lemma follows.

Proof. (Lemma 2) $R_2 \cap R_2^* \neq \emptyset$ only with negligible probability since their elements are randomly chosen. Hence, the lemma follows from Difference Lemma in [36].

Proof. (Lemma 3) Since the employed encryption scheme is secure from the premise, the lemma follows.

Note that the adversary never queries decryption queries since we are not considering chosen ciphertext attacks here.

Proof. (Lemma 4) Since the validness of the game guarantees that A^* never receives $(\mathsf{prokey}, \{lab^\dagger, lab\})$ for any $lab^\dagger \in \mathcal{L}^* \cap \bar{\mathcal{L}}$ and any *lab*, $\mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab^\dagger)$ is never given to A^*. Among the elements of (r_1, c, d), which is the response to $(\mathsf{encrypt}, lab^\dagger, msg^*_\beta)$ by C_3, only c depends on b. But, as $\mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab^\dagger)$ is never given to A^*, $\mathsf{Hash}_{\mathcal{G}_1}(\mathsf{Hash}_{\mathcal{K}}(1, param, mkey, lab^\dagger), r_1)$ will never be generated. Hence, the indistinguishability is due to the random oracle. Therefore, the lemma follows.

Proof. (Lemma 6) Since f for msg_{1-b} is also chosen randomly, the output of C_5 does not depends on b. Hence the lemma follows.

Stronger Security Model for Public-Key Encryption with Equality Test

Yao Lu[1,2], Rui Zhang[1], and Dongdai Lin[1]

[1] SKLOIS, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, 100093, P.R. China
[2] Graduate University of Chinese Academy of Sciences, Beijing, 100049, P.R. China
lywhhit@gmail.com, {r-zhang,ddlin}@iie.ac.cn

Abstract. In CT-RSA 2010, Yang et al. suggested a new category of probabilistic public-key encryption (PKE) schemes, called public-key encryption with equality test (PKET), which supports searching on ciphertexts without decrypting them. Typical applications include management of encrypted data in an outsourced database. They presented a construction in bilinear groups, and proved that it is one-way against chosen ciphertext attack (OW-CCA) in the random oracle model. We argue that OW-CCA security may be too weak for database applications, because partial information leakage from the ciphertext is not considered in the model. In this paper, we revisit the security models for PKET, and introduce a number of new security definitions. To remark, the weakest of our definitions is still stronger than OW-CCA. We then investigate relations among these security definitions. Finally, to illustrate the usefulness of our definitions, we analyze the security of a PKET scheme [24], showing the scheme actually provides much stronger security than that was proven previously.

Keywords: Public-Key Encryption with Equality Test, Deterministic Encryption, Searchable Encryption, Semantic Security.

1 Introduction

Background. Public-Key Encryption with Equality Test (PKET), proposed by Yang et al. [24], allows anyone to test whether two ciphertexts contain the same message. PKET has many important applications, e.g., a database can use it to implement searchable encrypted databases to with enhanced privacy. Compared with the previous public-key solutions [9,3], a remarkable property of PKET is that it supports searching on ciphertexts produced under different public keys. It is worth reminding that one has to assume the plaintext space of PKET must be large, otherwise, an adversary can simply "guess" the correct content of the message. Fortunately, this requirement is fulfilled by all interesting applications.

In [24], Yang et al. showed that no PKET scheme can achieve indistinguishability against even chosen plaintext attack (IND-CPA), and naturally, only one-wayness was considered. However, one-wayness can merely guarantee the whole

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 65–82, 2013.

plaintext is unrecoverable, and cannot capture any partial information leakage of the plaintext (See detailed explanations in Lemma 1). For most interesting applications of PKET, one-wayness is far from sufficient. In order to motivate more applications and to understand this new primitive better, in the paper, we investigate stronger security notions regarding PKET.

In particular, our goal is to have security definitions that are natural, convenient to use and "properly" strong. We noticed that PKET and deterministic encryption (DE) [3] are similar in functionalities. They can both be used as searchable encryption, and neither of them can satisfy any meaningful notion of security if the plaintext is distributed over a small space. DE was somehow well-studied [3,7,5,21] and a semantic security style definition of privacy, called PRIV, was well-known [3] in the literature. Adapting PRIV security to PKET will be an immediate solution to our problems, but this also introduces some undesirable issues.

Let us take a closer look at PRIV security. Informally speaking, PRIV requires that any polynomial-time adversary $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ should not win the following game: In phase 1, $\mathcal{A}_m$ selects a plaintext m from a space of large min-entropy and a partial information t about m, and sets t as the challenge. In phase 2, $\mathcal{A}_g$ tries to find the exact value of t, given the target ciphertext c, where c is the encryption of message m. We insist that $(\mathcal{A}_m, \mathcal{A}_g)$ share no common random tape and do not communicate. For a DE scheme, the ciphertext can be computed efficiently using the public key, which leaks non-trivial information about the plaintext. Thus in the formulation of PRIV security the public key is not included in the input for $\mathcal{A}_m$, and PRIV security is meaningful only if the plaintext is independent from the public key.

However, for a PKET scheme, thanks to the probabilistic encryption algorithm, every valid ciphertext is masked by additional randomness, the information regarding the plaintext that the adversary tries to extract from a ciphertext might be negligible. In this case, the above constraint disappears, we can pass the public key to $\mathcal{A}_m$, then get a stronger security notion. To summarize, we can expect stronger security from PKET than DE.

Related Work. In [9], Boneh et al. introduced the notion of public-key encryption with keyword search (PEKS) and several constructions that achieve semantic security. Informally, PEKS provides a mechanism that allows senders to store encrypted messages at a server, to each message one or more tags are attached that are keywords encrypted with the receiver's public key, the receiver may send a trapdoor, generated based on the receiver's private key, to the server so that the latter can search the tags attached to each encrypted message, while the server and other parties exclude receiver do not learn anything else about the tags. Abdalla et al. [1] provided a transform from any anonymous Identity-Based Encryption (IBE) scheme to a secure PEKS scheme. Crescenzo et al. [14] proposed a PEKS construction based on Jacobi symbols.

In [3], Bellare et al. formally studied a notion of security for deterministic public-key encryption that essentially guarantees semantic security for high-entropy messages, and showed how to achieve it in the random oracle model.

Boldyreva et al. [7] introduced a slightly weaker notion of security, which no partial information about encrypted messages should be leaked as long as each message is a-priori hard-to-guess given the others, and give general constructions without random oracles. Subsequent works by Bellare et al. [5] provided alternative security definitions and proved definitional equivalences for DE. Recently a similar formalization of security notions was presented in [11] for "plaintext-checkable encryption" (PCE) which is a probabilistic public-key encryption scheme with an additional functionality that anyone can test whether a ciphertext c is the encryption of a given plaintext m under a public encryption key pk.

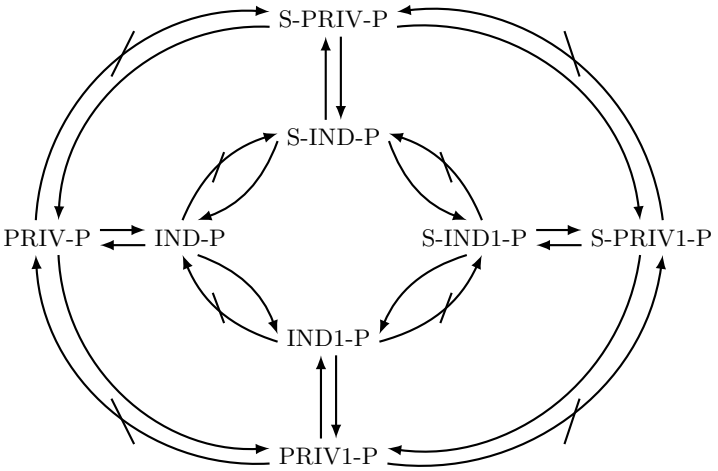
Our Contributions. We establish various security definitions for PKET, which fall in two flavors: semantic security style notions and sources indistinguishability-based style notions. To distinguish our new definitions from those for DE, we use "PRIV-P" to denote PRIV security in the setting of probabilistic encryption, where "-P" stands for "probabilistic". As mentioned above, PRIV security assume that plaintexts are chosen independently from the public key, thus only capture a weak security. We remove such assumptions, and obtain a stronger security notion, called "S-PRIV-P" security, where "S-" stands for "strong". Note that a single-message actually results in weaker security than the multi-message, so we additionally consider weaker notions such as "PRIV1-P" and "S-PRIV1-P" where "1" stands for single-message.

Furthermore, we consider a sources indistinguishability-based notion for PKET (called IND-P), which asks that a scheme hides the "source" from which the data is drawn, meaning it is hard to distinguish ciphertext whose corresponding plaintexts are drawn from one of two possible distributions. Similar to semantic security style notion, we get notions: S-IND-P, IND1-P, and S-IND1-P.

The above discussions are within security goals, and one can further combine attack models such as CPA or CCA to describe security requirements for concrete systems. Similar results as [4] can be gained. However, we consider it less important and omit it here. We analyze relations among the eight notions discussed above, and our results are summarized in Fig.1. We can see that the semantic security style notion and sources indistinguishability-based style notion are equivalent. The weakest notion is PRIV1-P which equals to IND1-P, however, it still stronger than OW security.

Finally we review the two schemes for PKET in [24]: for the first one, we explain why it cannot be proved using general strategies; for the modified one designed for encrypting long messages, we prove that it can actually achieve S-PRIV-P security in the random oracle model.

Organization. The paper is organized as follows. In section 2 we give the preliminary. In section 3 we establish our security models and analyze the relations among them. In section 4 we review the schemes in [24] and prove that the second one actually can achieve S-PRIV-P security.



An arrow $X \rightarrow Y$ means any scheme secure under definition X is also secure under definition Y, and $X \nrightarrow Y$ means a scheme secure under definition X may not be secure under definition Y.

Fig. 1. The Relations Among the Security Notions for PKET

2 Preliminaries

In this section, we review the model of PKET and some mathematical assumptions.

Notations. 1^k denotes the string of k ones. If x is a string $|x|$ denotes its length. If S is a set, $x \leftarrow S$ denotes that x is sampled at random from S. We let $x[1]$ denote the most significance bit (MSB) of x, and let $x[i]$ denote the i^{th} MSB in x. Vectors are denoted in boldface, for example $\mathbf{x}$. If $\mathbf{x}$ is a vector then $|\mathbf{x}|$ denotes the number of components of $\mathbf{x}$ and $\mathbf{x}[i]$ denotes its i^{th} component for $1 \leq i \leq |\mathbf{x}|$. A function $f(k)$: $\mathbb{N} \rightarrow (0,1)$ is called negligible if it approaches zero faster than k^{-c}, where $c \in \mathbb{N}$ is a constant. We use $PtSp(k)$ to denote the plaintext space.

2.1 Public-Key Encryption with Equality Test (PKET)

Syntax. A probabilistic public-key encryption with equality test scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ consists of the following algorithms:

- $\mathcal{K}$, a probabilistic key generation algorithm, takes a security parameter $k \in \mathbb{N}$ as input and outputs a public/private key pair (pk, sk).
- $\mathcal{E}$, a probabilistic encryption algorithm, takes a message m and the public key pk as input, and outputs a ciphertext c.

- $\mathcal{D}$, a deterministic decryption algorithm, takes sk and c as input, and outputs m or $\perp$ (which indicates decryption failure).
- $\mathcal{T}$, a deterministic ciphertext comparison algorithm, takes two ciphertexts c_1 and c_2 which generated under public keys pk and $pk^{'}$ as input, outputs 1 if and only if c_1 and c_2 are encrypting the same message, otherwise 0.

In [24], it was shown that IND-ATK cannot be satisfied by any PKET schemes because of the equality test algorithm. Therefore only one-wayness for PKET schemes was considered.

Definition 1. (**OW-ATK**) *$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ is a PKET scheme. $\mathcal{A}$ is a polynomial-time adversary. For $atk \in \{cpa, cca\}$ and $k \in \mathbb{N}$, let*

$$\mathbf{Adv}^{ow\text{-}atk}_{\mathcal{A},\Pi} = Pr\begin{bmatrix} m^{'} = m \mid (pk, sk) \leftarrow \mathcal{K}(1^k), m \leftarrow PtSp(k) \\ y \leftarrow \mathcal{E}(1^k, pk, m), m^{'} \leftarrow \mathcal{A}^{\mathcal{O}}(1^k, pk, y) \end{bmatrix}$$

where

$$\begin{array}{ll} \textit{If} \;\; atk{=}cpa \;\; \textit{then} & \mathcal{O}(\cdot) = \epsilon \\ \textit{If} \;\; atk{=}cca \;\; \textit{then} & \mathcal{O}(\cdot) = \mathcal{D}(\cdot) \end{array}$$

In the case of cca, we insist that $\mathcal{A}$ does not query $\mathcal{D}$ on y. Π is OW-ATK secure if $\mathbf{Adv}^{ow\text{-}atk}_{\mathcal{A},\Pi}(k)$ is negligible for every polynomial-time adversary $\mathcal{A}$.

2.2 Mathematical Assumptions

Let $\mathbb{G}_1$, $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p and g be a generator of $\mathbb{G}_1$. A bilinear map $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfies the following properties:

1. Bilinear : For any $x, y \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}_q$, $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$;
2. Non-degenerate: $\hat{e}(g, g) \neq 1$;
3. Computable: There is an efficient algorithm to compute $\hat{e}(x, y)$ for any x, $y \in \mathbb{G}_1$.

Computational Diffie-Hellman (CDH) Problem : We say that CDH problem is (ϵ, T)-hard in $\mathbb{G}$, if given 3-tuple $(g, g^a, g^b) \in (\mathbb{G})^3$ as input, any randomized algorithm $\mathcal{A}$ with running time at most T, computes g^{ab} with advantage at most ϵ.

$$\mathbf{Adv}^{cdh}_{\mathcal{A},\mathbb{G}} \stackrel{\text{def}}{=} Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}]$$

We say that the CDH assumption holds if for any polynomial-time algorithm $\mathcal{A}$, its advantage $\mathbf{Adv}^{cdh}_{\mathcal{A},\mathbb{G}}$ is negligible.

3 New Security Definitions for PKET

In this section we present two types of security definitions for PKET, i.e., semantic security style definitions and sources indistinguishability-based style definitions, and investigate relations among these security definitions.

3.1 Semantic Security Style Definitions

Loosely speaking, an encryption scheme is semantically secure if it is infeasible to learn any information about the plaintext from the ciphertext. In the context of DE, Bellare et al. [3] formalize a semantic security style notion PRIV that captures the intuition. Due to the ciphertext comparability in PKET scheme, the security notions we consider here have some connections with those for DE. Similar to the notion PRIV for DE, we can define PRIV-P for PKET, where "-P" stands for "probabilistic". Furthermore, we extend the definition PRIV-P, and obtain a stronger definition called S-PRIV-P, where "-S" stands for "strong"

A priv-p-adversary $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ is a 2-tuple algorithm. $\mathcal{A}_m$ takes 1^k as input, and returns a vector of challenge message $\mathbf{x}$ together with a test string t that represents some partial information about $\mathbf{x}$. $\mathcal{A}_g$ takes 1^k, pk, $\mathbf{c}$ (the encryption of $\mathbf{x}$ under pk) as input, and tries to compute t. The adversary is legitimate if it obeys the following rules. First, there must exist function $v(\cdot)$, $l(\cdot)$ such that $|\mathbf{x}| = v(k)$ and $|\mathbf{x}[i]| = l(k)$ for all k, all $(\mathbf{x}, t)$ output by $\mathcal{A}_m(1^k)$. Second, all plaintext vectors must have the same equality pattern, meaning for all $1 \leq i, j \leq v(k)$ there is a symbol $\Diamond \in \{=, \neq\}$ such that $\mathbf{x}[i] \Diamond \mathbf{x}[j]$ for all $(\mathbf{x}, t)$ output by $\mathcal{A}_m(1^k)$.

We say that an adversary $\mathcal{A}$ has min-entropy μ if

$$Pr[\mathbf{x}[i] = x : (\mathbf{x}, t) \leftarrow \mathcal{A}_m(1^k)] \leq 2^{-\mu(k)}$$

for all $1 \leq i \leq v(k)$, all k, and all $x \in \{0,1\}^*$. $\mathcal{A}$ is said to have high min-entropy if it has min-entropy μ with $\mu(k) \in \omega(log(k))$.

Definition 2. (**PRIV-P**) *$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ is a PKET scheme. $\mathcal{A} = (\mathcal{A}_m, \mathcal{A}_g)$ is a polynomial-time adversary. $\mathcal{A}_m$ and $\mathcal{A}_g$ share neither coins nor state. For $atk \in \{cpa, cca\}$ and $k \in N$, let*

$$\mathbf{Adv}^{priv\text{-}p\text{-}atk}_{\mathcal{A},\Pi}(k) = 2Pr[\mathbf{Exp}^{priv\text{-}p}_{\mathcal{A},\Pi}(k) \Rightarrow true] - 1$$

where:

$$Pr[\mathbf{Exp}^{priv\text{-}p}_{\mathcal{A},\Pi}(k) \Rightarrow true] = Pr\left[\begin{array}{l} b^{'} = b \mid b \leftarrow \{0,1\}, (pk, sk) \leftarrow \mathcal{K}(1^k), \\ (\mathbf{x}_0, t_0) \leftarrow \mathcal{A}_m(1^k), (\mathbf{x}_1, t_1) \leftarrow \mathcal{A}_m(1^k), \\ \mathbf{c} \leftarrow \mathcal{E}(1^k, pk, \mathbf{x}_b), h \leftarrow \mathcal{A}^{\mathcal{O}}_g(1^k, pk, \mathbf{c}), \\ \textit{If } h = t_1 \textit{ then } b^{'} \leftarrow 1 \quad \textit{Else } b^{'} \leftarrow 0 \end{array}\right]$$

and

$$\begin{array}{ll} \textit{If} & \textit{atk=cpa} \quad \textit{then} \quad \mathcal{O}(\cdot) = \epsilon \\ \textit{If} & \textit{atk=cca} \quad \textit{then} \quad \mathcal{O}(\cdot) = \mathcal{D}(\cdot) \end{array}$$

In the case of cca, we insist that $\mathcal{A}_g$ can query on any ciphertext not having appeared in $\mathbf{c}$. Π is **PRIV-P** *secure if $\mathbf{Adv}^{priv\text{-}p\text{-}atk}_{\mathcal{A},\Pi}(k)$ is negligible for every polynomial-time adversary $\mathcal{A}$ with high min-entropy.*

The above definition is for the multi-message case. We can define an another notion PRIV1-P accordingly: Π is **PRIV1-P** secure if $\mathbf{Adv}^{priv\text{-}atk}_{\mathcal{A},\Pi}(k)$ is negligible for every polynomial-time privacy adversary $\mathcal{A}$ with $Pr[|\mathbf{x}| = 1 : (\mathbf{x}, t) \leftarrow \mathcal{A}_m(1^k)] = 1$ for all $k \in \mathbb{N}$.

Lemma 1. *For a PKET scheme, OW security is strictly weaker than PRIV1-P security.*

Proof. To prove this, we construct an encryption scheme $\Pi^{'}$ which is OW secure but not PRIV1-P secure. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ be a OW secure PKET scheme. We define $\Pi^{'} = (\mathcal{K}, \mathcal{E}^{'}, \mathcal{D}^{'}, \mathcal{T}^{'})$ as follows:

Algorithm $\mathcal{E}^{'}(1^k, pk, x)$	**Algorithm** $\mathcal{D}^{'}(1^k, pk, sk, c)$	**Algorithm** $\mathcal{T}^{'}(1^k, c_1, c_2)$
$y \leftarrow \mathcal{E}(1^k, pk, x)$	$y \parallel d \leftarrow c$	$y_1 \parallel d_1 \leftarrow c_1$
return $y \parallel x[1]$	$x \leftarrow \mathcal{D}(1^k, pk, sk, y)$	$y_2 \parallel d_2 \leftarrow c_2$
	if $x[1] = d$ then return x	return
	else return $\perp$	$\mathcal{T}(y_1, y_2) \wedge \neg(d_1 \oplus d_2)$

The assumption that Π is OW secure implies that Π' is OW secure. However, the following attack shows Π' is not PRIV1-P secure. Consider $\mathcal{A}_m(1^k)$ outputs (x, t) where $t = x[1]$, then $\mathcal{A}_g(1^k, pk, c\ (c = y \parallel d))$ return d. Therefore, the advantage of the priv1-p adversary is $\mathbf{Adv}^{priv\text{-}p}_{\mathcal{A},\Pi'} \geqslant 1/2$. □

Just like DE scheme [3], the single-message security definition and the multi-message version have different security level for PKET scheme.

Lemma 2. *For a PKET scheme, PRIV1-P security is strictly weaker than PRIV-P security.*

Proof. To prove this, we give an example of an encryption scheme $\Pi^{'}$ which is PRIV1-P secure but not PRIV-P secure. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ be a PRIV-P secure PKET scheme. We define $\Pi^{'} = (\mathcal{K}, \mathcal{E}^{'}, \mathcal{D}^{'}, \mathcal{T}^{'})$ as follows:

Algorithm $\mathcal{E}^{'}(1^k, pk, x)$	**Algorithm** $\mathcal{D}^{'}(1^k, pk, sk, c)$	**Algorithm** $\mathcal{T}^{'}(1^k, c_1, c_2)$
$y \leftarrow \mathcal{E}(1^k, pk, x)$	$y \parallel z \leftarrow c$	$y_1 \parallel z_1 \leftarrow c_1$
$z \leftarrow \mathcal{E}(1^k, pk, \bar{x})$	$x \leftarrow \mathcal{D}(1^k, pk, sk, y)$	$y_2 \parallel z_2 \leftarrow c_2$
return $y \parallel z$	$x^{'} \leftarrow \mathcal{D}(1^k, pk, sk, z)$	return
	if $x^{'} = \bar{x}$ then return x	$\mathcal{T}(y_1, y_2) \wedge \mathcal{T}(z_1, z_2)$
	else return $\perp$	

Here $\bar{x}$ denotes the bitwise complement of a string s. The assumption that Π is PRIV-P secure implies that Π' is PRIV1-P secure. However, the following attack shows Π' is not PRIV-P secure. Consider $\mathcal{A}_m(1^k)$ that picks x_1, x_2 from $\{0,1\}^k$ and outputs $(\mathbf{x}_1(\mathbf{x}_1 = (x_1, \bar{x}_1)), 1)$, $(\mathbf{x}_2(\mathbf{x}_2 = (x_1, x_2)), 0)$. Let $\mathcal{A}_g(1^k, pk, (y_1 \parallel z_1, y_2 \parallel z_2)$ outputs $\mathcal{T}(z_1, y_2)$, and we have $\mathbf{Adv}^{priv\text{-}p}_{\mathcal{A},\Pi'} \geq 1/2$. □

In the previous definitions for PRIV security [3], $\mathcal{A}_m$ cannot be given pk, otherwise the ciphertext itself leaks partial information about the plaintext, as a result, the PRIV security would be unachievable. However, for probabilistic encryption schemes this restriction no longer exists, since ciphertext $\mathbf{c}$ is masked by additional randomness used in the encryption algorithm. Therefore, we can give a strong privacy regarding probabilistic definition, namely, S-PRIV-P.

Definition 3. *$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ is a PKET scheme. Compared with Definition 2, $\mathcal{A}_m$ is additionally given pk, We say that Π is* **S-PRIV-P** *secure if for any polynomial-time attacker $\mathcal{A}$, we have that $\mathbf{Adv}^{s\text{-}priv\text{-}p}_{\mathcal{A},\Pi}(k)$ is negligible.*

For the single-message scenario, we define **S-PRIV1-P** security accordingly.

We note that the above definition is strictly stronger than PRIV security, for simplicity, we consider the single-message scenario. We can get similar result for the multi-message scenario.

Lemma 3. *For a PKET scheme, PRIV1-P security is strictly weaker than S-PRIV1-P security.*

Proof. To prove this, we construct an encryption scheme Π' which is PRIV1-P secure but not S-PRIV1-P secure. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ be a S-PRIV1-P secure PKET scheme. $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be a collision resistant hash function with the property that $H(pk) \in PtSp(k)$ for all $pk \in \{0,1\}^*$. We define $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}', \mathcal{T}')$ as follows:

Algorithm $\mathcal{E}'(1^k, pk, x)$	**Algorithm** $\mathcal{D}'(1^k, pk, sk, c)$	**Algorithm** $\mathcal{T}'(1^k, c_1, c_2)$
$r \leftarrow \{0,1\}^k$	$y \parallel z \leftarrow c$	$y_1 \parallel z_1 \leftarrow c_1$
$y \leftarrow \mathcal{E}(1^k, pk, x)$	$x \leftarrow \mathcal{D}(1^k, pk, sk, y)$	$y_2 \parallel z_2 \leftarrow c_2$
if $x = H(pk)$ then	return x	return $\mathcal{T}(y_1, y_2)$
return $y \parallel x$		
else return $y \parallel r$		

We can see that Π' is a PKET scheme with PRIV1-P security. However, the following attack shows that Π' is not S-PRIV1-P secure. Consider that $\mathcal{A}_m(1^k, pk)$ outputs (x_0, t_0), (x_1, t_1), where $x_1 = H(pk)$, $t_b = x_b (b \in \{0,1\})$. Let $\mathcal{A}_g(1^k, pk, c(c = y \parallel z))$ outputs z, and $\mathbf{Adv}^{s\text{-}priv1\text{-}p}_{\mathcal{A},\Pi} \geq 1 - \epsilon(k)$ (since H is a collision resistant hash function, $\epsilon(k)$ is negligible). □

3.2 Sources Indistinguishability-Based Style Definitions

We consider sources indistinguishability-based definitions, which are easier to deal with. It requires that the adversary is unable to distinguish encryption of plaintexts drawn from two adversary-specified, high-entropy message spaces. Similar to the semantic security style definitions, we get four sources indistinguishability-based definitions.

An ind-p-adversary $\mathcal{I} = (\mathcal{I}_m, \mathcal{I}_g)$ is a 2-tuple algorithm. $\mathcal{I}_m$ takes 1^k, a bit b as input, and returns a vector of message $\mathbf{x}$. $\mathcal{I}_g$ takes 1^k, pk, $\mathbf{c}$ (the encryption

of $\mathbf{x}$ under pk) as input, and tries to guess the bit b. The adversary is legitimate if it obeys the following rules. First, there must exist function $v(\cdot)$, $l(\cdot)$ such that $|\mathbf{x}| = v(k)$ and $|\mathbf{x}[i]| = l(k)$ for all k, all $(\mathbf{x}, t)$ output by $\mathcal{I}_m(1^k, b)$ $(b \in \{0,1\})$, and all $1 \leq i \leq v$. Second, all plaintext vectors must have the same equality pattern, which was explained above.

We say that an adversary $\mathcal{I}$ has min-entropy μ if

$$Pr[\mathbf{x}[i] = x : (\mathbf{x}, t) \leftarrow \mathcal{I}_m(1^k, b)] \leq 2^{-\mu(k)}$$

for all $1 \leq i \leq v(k)$, all k, and all $x \in \{0,1\}^*$. $\mathcal{I}$ is said to have high min-entropy if it has min-entropy μ with $\mu(k) \in \omega(log(k))$.

Definition 4. (**IND-P**) *$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ is a PKET scheme. $\mathcal{I} = (\mathcal{I}_m, \mathcal{I}_g)$ is a polynomial-time adversary. $\mathcal{I}_m, \mathcal{I}_g$ share neither coins nor state. For $atk \in \{cpa, cca\}$ and $k \in \mathbb{N}$, let*

$$\mathbf{Adv}^{ind\text{-}p\text{-}atk}_{\mathcal{I},\Pi}(k) = 2Pr\left[\begin{array}{l} b^{'} = b \mid b \leftarrow (0,1), \\ (pk, sk) \leftarrow \mathcal{K}(1^k), \mathbf{x}_b \leftarrow \mathcal{I}_m(1^k, b), \\ \mathbf{c} \leftarrow \mathcal{E}(1^k, pk, \mathbf{x}_b), b^{'} \leftarrow \mathcal{I}_g^{\mathcal{O}}(1^k, pk, \mathbf{c}) \end{array}\right] - 1$$

and

$$\begin{array}{lll} \textit{If} & \textit{atk=cpa} & \textit{then} \quad \mathcal{O}(\cdot) = \epsilon \\ \textit{If} & \textit{atk=cca} & \textit{then} \quad \mathcal{O}(\cdot) = \mathcal{D}(\cdot) \end{array}$$

In the case of cca, we insist that $\mathcal{I}_g$ can query on any ciphertext not having appeared in $\mathbf{c}$. Π is **IND-P** *secure if $\mathbf{Adv}^{ind\text{-}p\text{-}atk}_{\mathcal{I},\Pi}(k)$ is negligible for every polynomial-time adversary $\mathcal{I}$ with high min-entropy.*

The above definition is for the multi-message case. We can define an another notion IND1-P accordingly: Π is **IND1-P** secure if $\mathbf{Adv}^{ind\text{-}p\text{-}atk}_{\mathcal{I},\Pi}(k)$ is negligible for every privacy adversary $\mathcal{I}$ with $Pr[|\mathbf{x}| = 1 : (\mathbf{x}, t) \leftarrow \mathcal{I}_m(1^k, b)] = 1$ for all $k \in N$.

Similar to Definition 3, if $\mathcal{I}_m$ is given pk, we can get a stronger security definition.

Definition 5. *$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ is a PKET scheme. Compared with Definition 4, $\mathcal{I}_m$ is additionally given pk, we say that Π is* **S-IND-P** *secure if for any polynomial-time attacker $\mathcal{I}$, we have that $\mathbf{Adv}^{ind\text{-}p}_{\mathcal{I},\Pi}(k)$ is negligible.*

For the single-message scenario, we can define **S-IND1-P** similarly. Using similar techniques in section 3.1, we get the following results.

Lemma 4. *For a PKET scheme, IND1-P security is strictly weaker than IND-P security.*

Lemma 5. *For a PKET scheme, IND1-P security is strictly weaker than S-IND1-P security.*

3.3 Equivalence of the Security Definitions

In this section we show the relations among the semantic security style definitions and the sources indistinguishability-based security style definitions. In particular, four pairs of the definitions: PRIV1-P and IND1-P, PRIV-P and IND-P, S-PRIV1-P and S-IND1-P, S-PRIV-P and S-IND-P are equivalent. We just need to show that PRIV-P is equivalent to IND-P, other pairs follow the similar technique, since the techniques we used are independent no matter whether pk is given to $\mathcal{A}_m$. Now we give the main theorem of the section.

Theorem 1. *$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ is a PKET scheme, let $\mathcal{A}$ be a PRIV-P adversary against Π, then there is a IND-P adversary such that for all $k \in \mathbb{N}$*

$$\mathbf{Adv}^{s\text{-}priv\text{-}p}_{\mathcal{A},\Pi}(k) \leq 6\mathbf{Adv}^{s\text{-}ind\text{-}p}_{\mathcal{I},\Pi}(k) + (\frac{2}{3})^k$$

and the running-time of $\mathcal{I}$ is the time for at most that for k executions of $\mathcal{A}$.

Here we focus on the IND-P to PRIV-P implication, for the opposite direction is straightforward. The proof is identical to [21] apart from a few minor differences. We give the entire proof in appendix.
In the rest part of this paper, without loss of generality we limit the adversary $\mathcal{A}$ as a 0-balanced boolean function, which means the probability the partial information is 1 or 0 is 1/2.

4 Revisiting Security of Yang et al.'s Schemes

The description of Yang et al.'s scheme is given in Figure 2. We remark that the scheme seems difficult to be proved PRIV1-P secure (the weakest notion we define in the section 3). Because in the security proof, to show a meaningful reduction, the simulator should generate a valid challenge cipertext, but it cannot compute m^r, thus fails.

In [24], the authors also present a modified scheme to encrypt long messages and prove that it is OW-CCA secure. We review it here and show that it satisfies S-PRIV1-P-CCA security. We give the description of the modified scheme in Figure 3 (The different parts of the two schemes are labeled out with boxes).

Theorem 2. *The PKET scheme in Figure 3 is S-PRIV-P-CCA secure in the random oracle model assuming PRG is a secure pseudo-random bit generator and CDH problem is intractable.*

Proof. Let $\mathcal{A}$ is an algorithm that has advantage ϵ in breaking the above PKET scheme. Suppose that $\mathcal{A}$ runs in time t and makes at most q_G hash G function queries, q_H hash H function queries and q_D decryption queries. Now we construct an algorithm $\mathcal{B}$ that can solve CDH problem with probability at least ϵ', where

$$\epsilon' \geq \epsilon \left(1 - \frac{q_D}{2^k} - \frac{q_D}{2^{k+l}}\right) \left(1 - \frac{v}{2^\mu}\right)^{q_G}$$

$\mathcal{K}(1^k)$:	$\mathcal{D}(sk, C)$:
$x \leftarrow \mathbb{Z}_q^*$;	$C = (U, V, W)$;
$y = g^x$;	$m\|\|r \leftarrow H(U, V, U^x) \oplus W$;
$pk = (y)$;	If $(m \in \mathbb{G}_1^* \wedge r \in \mathbb{Z}_q^* \wedge U = g^r \wedge V = m^r)$;
$sk = (x)$;	return m;
return (pk,sk);	otherwise, return $\perp$;
$\mathcal{E}(pk, m)$:	$\mathcal{T}(C_1, C_2)$:
$r \leftarrow \mathbb{Z}_q^*$;	$C_1 = (U_1, V_1, W_1)$;
$U \leftarrow g^r$;	$C_2 = (U_2, V_2, W_2)$;
$V \leftarrow m^r (m \in \mathbb{G}_1^*)$;	if $\hat{e}(U_1, V_2) = \hat{e}(U_2, V_1)$;
$W \leftarrow H(U, V, y^r) \oplus m\|\|r$;	return 1;
$C = (U, V, W)$;	otherwise, return 0;
return(C);	

H is a hash function: $\mathbb{G}_1^3 \rightarrow \{0,1\}^{k+l}$, where k and l are security parameters such that elements of $\mathbb{G}_1$ can be represented with k bits and elements of $\mathbb{Z}_q$ can be represented with l bits.

Fig. 2. Yang et al.'s PKET Scheme

$\mathcal{K}(1^k)$:	$\mathcal{D}(sk, C)$:
$x \leftarrow \mathbb{Z}_q^*$;	$C = (U, V, W)$;
$y = g^x$;	$K \leftarrow H(U, V, U^x)$
$pk = (y)$;	$m\|\|r \leftarrow PRG(K) \oplus W$;
$sk = (x)$;	If $(r \in \mathbb{Z}_q^* \wedge U = g^r \wedge \boxed{V = G(m)^r})$;
return (pk,sk);	return m;
	otherwise, return $\perp$;
$\mathcal{E}(pk, m)$:	$\mathcal{T}(C_1, C_2)$:
$r \leftarrow \mathbb{Z}_q^*$;	$C_1 = (U_1, V_1, W_1)$;
$U \leftarrow g^r$;	$C_2 = (U_2, V_2, W_2)$;
$\boxed{V \leftarrow G(m)^r;}$	if $\hat{e}(U_1, V_2) = \hat{e}(U_2, V_1)$;
$K \leftarrow H(U, V, y^r)$	return 1;
$W \leftarrow PRG(K) \oplus m\|\|r$;	otherwise, return 0;
$C = (U, V, W)$; ;	
return(C).	

G is a collision resistant hash function: $\{0,1\}^* \rightarrow \mathbb{G}_1^*$. H is a hash function: $\mathbb{G}_1^3 \rightarrow \{0,1\}^{k+l}$, where k and l are security parameters such that elements of $\mathbb{G}_1$ can be represented with k bits and elements of $\mathbb{Z}_q$ can be represented with l bits. PRG is a pseudo-random bit generator.

Fig. 3. The Modified Scheme

Let g be a generator of $\mathbb{G}_1$. Algorithm $\mathcal{B}$ is given a tuple $(g, g^a, g^c) \in \mathbb{G}_1^3$. Its goal is to compute g^{ac}. We describe algorithm $\mathcal{B}$ as follows:

SetUp. Algorithm $\mathcal{B}$ sets the public key $pk = y$ which $y = g^a$, the secret key a which is unknown to $\mathcal{B}$. $\mathcal{O}_G$, $\mathcal{O}_H$ are random oracles which are controlled by $\mathcal{B}$;

Challenge. First, $(\mathbf{m}^*, t^*) \leftarrow \mathcal{A}_m(g, y, 1^k)$, and $(\mathbf{m}^{'}, t^{'}) \leftarrow \mathcal{A}_m(g, y, 1^k)$. Here $\mathbf{m}^*$ and $\mathbf{m}^{'}$ have the same equality pattern, we assume $|\mathbf{m}^*| = v$, $|\mathbf{m}^*[i]| = d (1 \leqslant i \leqslant v, d \geqslant k)$, and $\mathbf{m}^*[i] \neq \mathbf{m}^*[j]$ for all $1 \leqslant i, j \leqslant v$ (other message patterns can be handled using similar technique). Then $\mathcal{B}$ sets the target ciphertext $\mathbf{C}^* = (\mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*)$ which $\mathbf{U}^*[i] = g^c$, $\mathbf{V}^*[i] \leftarrow \{0,1\}^k$, $\mathbf{W}^*[i] \leftarrow \{0,1\}^{d+l}$, for all i which $1 \leqslant i \leqslant v$.

Guess $t^{'} \leftarrow \mathcal{A}_g^{\mathcal{O}_G, \mathcal{O}_H, \mathcal{O}_D}(g, y, \mathbf{U}^*, \mathbf{V}^*, \mathbf{W}^*)$. In the phase, the oracles for $\mathcal{A}_g$ are simulated as follows:

- $\mathcal{O}_G$: On input $m \in \mathbb{G}_1$, if $m = \mathbf{m}^*[i]$ for any $1 \leqslant i \leqslant v$, the algorithm $\mathcal{B}$ aborts with failure. If there is an entry (m, h_1) in the hash table T_1 maintained by $\mathcal{B}$, h_1 is returned; otherwise, a random value h_1 is selected from $\mathbb{G}_1^*$ and returned, and (m, h_1) is added into T_1.
- $\mathcal{O}_H$: On input $(U, V, Z) \in \mathbb{G}_1^3$, if $U = U^*$, $\mathcal{B}$ checks if $\hat{e}(g, Z) = \hat{e}(y, U^*)$, if the equation holds, $\mathcal{B}$ outputs Z and aborts the algorithm. Or if there is an entry (U, V, Z, h_2) in the hash table T_2 maintained by $\mathcal{B}$, h_2 is returned; otherwise, a random value h_2 is selected and returned, and (U, V, Z, h_2) is added into T_2.
- $\mathcal{O}_D$: On input a ciphertext $\mathbf{C} = (\mathbf{U}, \mathbf{V}, \mathbf{W})$, we deal with the messages separately.
  For each $\mathbf{C}[j] = (\mathbf{U}[j], \mathbf{V}[j], \mathbf{W}[j])$, if the input is $\mathbf{U}[j] = \mathbf{U}^*[i]$, $\mathbf{V}[j] = \mathbf{V}^*[i]$ and $\mathbf{W}[j] \neq \mathbf{W}^*[i]$ for any $1 \leqslant i \leqslant v$, $\mathcal{B}$ returns $\perp$. Otherwise $\mathcal{B}$ searches T_2 for an entry of the form $(U, V, \cdot, \cdot)$. For each item (U, V, Z, h), $\mathcal{B}$ computes $m||r = PRG(h) \oplus W$ and proceeds as follows:
  1. $\mathcal{B}$ searches T_1 for m, if m is not in the list, $\mathcal{B}$ returns $\perp$; when h_1 is returned, check if $U = g^r$, $V = h_1^r$ and $Z = y^r$. If the equations holds, $\mathcal{B}$ obtains m.
  2. Otherwise, $\mathcal{B}$ continues to search T_2 for the next entry of the form $(U, V, \cdot, \cdot)$.

  If nothing is returned to $\mathcal{B}$ in the above loop for all entries $(U, V, \cdot, \cdot)$ in T_2, $\mathcal{B}$ returns $\perp$. Finally, combining all the messages, $\mathcal{B}$ returns $\mathbf{m}$.

This completes the description of $\mathcal{B}$. As $\mathcal{O}_G$ is a random oracle, the challenge ciphertext is always valid. Next we show that $\mathcal{B}$ correctly output g^{ac} with probability at least $\epsilon^{'}$. First we define three events:

E In the simulation the adversary $\mathcal{A}$ queries $\mathcal{O}_H$ on input $(g^c, \cdot, g^{ac})$.
F In the simulation the adversary $\mathcal{A}$ queries $\mathcal{O}_G$ on input $\mathbf{m}^*[i]$.
I In the simulation a valid ciphertext is rejected.

Claim 1: $Pr[\mathbf{E}] = \epsilon^{'}$

This is straightforward, we don't give further explanation.

Claim 2: $Pr[\mathbf{I}] \leq \frac{q_D}{2^k} + \frac{q_D}{2^{k+l}}$

Proof. Algorithm $\mathcal{B}$ will simulate the decryption oracle perfectly except for the following two cases.

- Case 1: (U, V, U^a) has never been queried to $\mathcal{O}_H$ before a decryption query (U, V, W) is issued. In the case, $\perp$ is returned by the decryption oracle. The simulation fails if (U, V, W) is a valid ciphertext. Due to the idealness of the random oracle, this happens with probability $1/2^{k+l}$.
- Case 2: (U, V, U^a) has been queried to $\mathcal{O}_H$ before a decryption query (U, V, W) is issued, and m can be computed from $W \oplus H(U, V, U^a)$, but m has never been queried to $\mathcal{O}_G$. In the case, $\perp$ is returned by the decryption oracle. The simulation fails if (U, V, W) is a valid ciphertext. Due to the idealness of the random oracle, this happens with probability $1/2^k$.

Combining the above two cases, we get the final result $Pr[\mathbf{I}] \leq \frac{q_D}{2^k} + \frac{q_D}{2^{k+l}}$ □

Claim 3: $Pr[\mathbf{F}] \leq 1 - (1 - \frac{v}{2^\mu})^{q_G}$

Proof. In the simulation, if event $\mathbf{E}$ occurs, the algorithm $\mathcal{B}$ aborts. So if event $\mathbf{F}$ occurs before event $\mathbf{E}$ occurs, then the adversary $\mathcal{A}$ can not get any information for $\mathbf{m}$. We assume that the distribution of plaintext messages has min-entropy μ, then we get the result of $Pr[F]$.

$$Pr[\mathbf{F}] = Pr[\mathbf{F}|\neg\mathbf{E}] \leq 1 - (1 - \frac{v}{2^\mu})^{q_G}$$

□

Claim 4: Suppose that in a real attack $\mathcal{A}_m$ is given the public key $pk = g^a$, and selects $(\mathbf{m}_0, t_0), (\mathbf{m}^*, t^*)$, $\mathcal{A}_g$ is given the target ciphertext $\mathbf{C}^*$, and guesses b' at last. Then in the real attack $\mathcal{A}_g$ queries $\mathcal{O}_H$ for $(g^c, \cdot, g^{ac})$ with probability at least ϵ.

Proof. Denote $\mathbf{E}'$ be the event that in the real attack $\mathcal{A}_g$ queries $\mathcal{O}_H$ for $(g^c, \cdot, g^{ac})$. If $\mathbf{E}'$ does not occur, we have that the bit $t \in \{0, 1\}$ is independent of $\mathcal{A}_g$'s view, since f is a balanced boolean function, then $\mathcal{A}_g$ outputs b' which satisfies $b = b'$ with probability at most $1/2$. By the assumption of $\mathcal{A}_g$, we know that in the real attack $2Pr[b = b'] - 1 = \epsilon$. Combining with the two facts, we show that $Pr[\mathbf{E}'] \geqslant \epsilon$.

$$\begin{aligned} Pr[b = b'] &= Pr[b = b'|\mathbf{E}']Pr[\mathbf{E}'] + Pr[b = b'|\neg\mathbf{E}']Pr[\neg\mathbf{E}'] \\ &\leq Pr[\mathbf{E}'] + \frac{1}{2}Pr[\neg\mathbf{E}'] \\ &\leq \frac{1}{2} + \frac{1}{2}Pr[\mathbf{E}'] \end{aligned}$$

In the end, we get the result:

$$Pr[\mathbf{E}^{'}] \geqslant 2Pr[b = b^{'}] - 1 \geqslant \epsilon$$

□

At last, we get the probability that solving CDH problem:

$$\epsilon^{'} \geq \epsilon \left(1 - \frac{q_D}{2^k} - \frac{q_D}{2^{k+l}}\right) \left(1 - \frac{v}{2^{\mu}}\right)^{q_G}$$

So we get a non-negligible advantage solving CDH problem, this is a contradiction to the assumption. This completes the proof of Theorem 2. □

Acknowledgments. We would like to thank the anonymous reviewers and especially Damien Verguaud for helpful comments. This work is supported by the National 973 Program of China under Grant 2011CB302400, the National Natural Science Foundation of China under Grant 60970152, the Grand Project of Institute of Software under Grant YOCX285056, One Hundred Person Project of the Chinese Academy of Sciences under Grant NSFC61100225.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Baek, J., Safavi-Naini, R., Susilo, W.: Public Key Encryption with Keyword Search Revisited. In: Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2008, Part I. LNCS, vol. 5072, pp. 1249–1259. Springer, Heidelberg (2008)
3. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and Efficiently Searchable Encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
4. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
5. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic Encryption: Definitional Equivalences and Constructions without Random Oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008)
6. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing effcient protocols. In: Ashby (ed.) ACM CCS 1993, pp. 62–73. ACM Press (1993)
7. Boldyreva, A., Fehr, S., O'Neill, A.: On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
8. Boneh, D., Boyen, X.: Secure Identity Based Encryption Without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)

9. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
10. Brakerski, Z., Segev, G.: Better security for deterministic public-key encryption: the auxiliary-input setting, Cryptology ePrint Archieve, Report 2011/209 (2011)
11. Canard, S., Fuchsbauer, G., Gouget, A., Laguillaumie, F.: Plaintext-Checkable Encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 332–348. Springer, Heidelberg (2012)
12. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
13. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
14. Di Crescenzo, G., Saraswat, V.: Public Key Encryption with Searchable Keywords Based on Jacobi Symbols. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 282–296. Springer, Heidelberg (2007)
15. Dodis, Y., Smith, A.: Entropic Security and the Encryption of High Entropy Messages. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 556–577. Springer, Heidelberg (2005)
16. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
17. Hofheinz, D., Weinreb, E.: Searchable encryption with decryption in the standard model, Cryptology ePrint Archieve, Report 2008/423 (2008)
18. Hwang, Y.H., Lee, P.J.: Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007)
19. Koblitz, N., Menezes, A.: Another look at "provable security". Journal of Cryptology 20, 3–37 (2007)
20. Menezes, A.: A introduction to pairing-based cryptography. Recent Trends in Cryptography 30, 1–19 (2005)
21. O'Neill, A.: Deterministic public-key encryption revisted. Cryptology ePrint Archieve, Report 2010/533 (2010)
22. Tang, Q., Chen, L.: Public-Key Encryption with Registered Keyword Search. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 163–178. Springer, Heidelberg (2010)
23. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
24. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic Public Key Encryption with Equality Test. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119–131. Springer, Heidelberg (2010)

A Proof of Theorem 1

We first show that is suffices to sonsider boolean PRIV adversaries (call a PRIV-P adversary $\mathcal{A}$ boolean if it outputs test strings of length 1).

Proposition 1. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ be a PKET scheme, and $\mathcal{A}$ be a PRIV-P adversary that outputs test strings of length l. Then there exists a boolean PRIV-P adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{priv-p}_{\Pi,\mathcal{A}}(k) \leqslant 2 \cdot \mathbf{Adv}^{priv-p}_{\Pi,\mathcal{B}}(k)$$

$\mathcal{B}$ has same message space as $\mathcal{A}$ and its running time is the time to run $\mathcal{A}$ plus $\mathcal{O}(l)$.

Proof. The proof is identical to the argument in [5,21]. To make sure the length of the test strings is 1, we suppose two public parameters: $(r,\ s)(r \leftarrow \{0,1\}^l$, $s \leftarrow \{0,1\})$. Adversary $\mathcal{B}$ works as follows:

$\mathbf{Algorithm}\ \mathcal{B}_m(1^k)$	$\mathbf{Algorithm}\ \mathcal{B}_g(1^k, pk, \mathbf{c})$
$(\mathbf{x}, t) \leftarrow \mathcal{A}_m(1^k)$	$g \leftarrow \mathcal{A}_g(1^k, pk, \mathbf{c})$
return $(\mathbf{x}, (\langle r, t\rangle \bigoplus s))$	return $\langle r, g\rangle \bigoplus s$

Then $\mathcal{B}$ is boolean, let A_d denotes the event $\mathbf{Exp}^{priv-p-d}_{\Pi,\mathcal{A}}(k) \Rightarrow true$ and similarly B_d denotes $\mathbf{Exp}^{priv-p-d}_{\Pi,\mathcal{B}}(k) \Rightarrow true$ $(d \in \{0,1\})$. Then

$$\begin{aligned}
\mathbf{Adv}^{priv-p}_{\Pi,\mathcal{B}}(k) &= Pr[B_1] - Pr[B_0] \\
&= \left(Pr[A_1] + \frac{1}{2} \cdot (1 - Pr[A_1])\right) - \left(Pr[A_0] + \frac{1}{2} \cdot (1 - Pr[A_0])\right) \\
&= \frac{1}{2} \cdot (Pr[A_1] - Pr[A_0]) \\
&= \frac{1}{2} \cdot \mathbf{Adv}^{priv-p}_{\Pi,\mathcal{A}}(k)
\end{aligned}$$

In the next step, we show that it in fact suffices to consider PRIV-P adversaries for which $\mathcal{B}$ is not just boolean but also balanced, meaning the probability the partial information is 1 or 0 cannot be negligible. Namly, call a boolean PRIV-P adversary $\mathcal{B}$ δ-balanced if for all $b \in \{0,1\}$

$$\mid Pr[t = b :\ (\mathbf{x}, t) \leftarrow \mathcal{B}_m(1^k)] - \frac{1}{2} \mid \leq \delta$$

Proposition 2. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ be a PKET scheme, and $\mathcal{B}$ be a boolean $1/2$-balanced PRIV-P adversary. Then for any $0 \leqslant \delta < 1/2$ there is a δ-balanced boolean PRIV-P adversary $\mathcal{B}'$ such that*

$$\mathbf{Adv}^{priv-p}_{\Pi,\mathcal{B}}(k) \leqslant 2\delta \cdot \mathbf{Adv}^{priv-p}_{\Pi,\mathcal{B}'}(k)$$

$\mathcal{B}'$ has same message space as $\mathcal{B}$ and its running time is the time to run $\mathcal{B}$ plus $\mathcal{O}(1/\delta)$.

Proof. For simplicity we assume $1/2\delta$ is odd. Adversary $\mathcal{B}'$ works as follows:

Algorithm $\mathcal{B}'_m(1^k)$	**Algorithm** $\mathcal{B}'_g(1^k, pk, \mathbf{c})$
$(\mathbf{x}, t) \leftarrow \mathcal{B}_m(1^k)$	$g \leftarrow \mathcal{B}_g(1^k, pk, \mathbf{c})$
$i \leftarrow [1, \cdots, 1/2\delta]$	return g
if $i \leqslant 1/4\delta - 1/2$ then return $(\mathbf{x}, 0)$	
else if $i \leqslant 1/2\delta - 1$ then return $(\mathbf{x}, 1)$	
else return $(\mathbf{x}, t)$	

After some calculations, we can get

$$| \, Pr[t = b : \ (\mathbf{x}, t) \leftarrow \mathcal{B}'_m(1^k)] - \frac{1}{2} \, | \leq \delta$$

Let B_d denotes the event $\mathbf{Exp}^{priv-p-d}_{\Pi,\mathcal{B}}(k) \Rightarrow true$, similarly B'_d denotes $\mathbf{Exp}^{priv-p-d}_{\Pi,\mathcal{B}'}(k) \Rightarrow true$ ($d \in \{0, 1\}$), and E denotes that $\mathcal{B}'_m$ picks $i = 1/2\delta$. Then

$$\begin{aligned}
\mathbf{Adv}^{priv-p}_{\Pi,\mathcal{B}'}(k) &= Pr[B'_1] - Pr[B'_0] \\
&= Pr[E] \cdot (Pr[B_1|E] - Pr[B_0|E]) + Pr[\bar{E}] \cdot (Pr[B_1|\bar{E}] - Pr[B_0|\bar{E}]) \\
&= Pr[E] \cdot (Pr[B_1|E] - Pr[B_0|E]) + Pr[\bar{E}] \cdot (\frac{1}{2} - \frac{1}{2}) \\
&= \frac{1}{2\delta} \cdot \mathbf{Adv}^{priv-p}_{\Pi,\mathcal{B}}(k)
\end{aligned}$$

The final component for the proof is as follows.

Proposition 3. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ be a PKET scheme, and $\mathcal{B}'$ be a δ-balanced boolean PRIV-P adversary ($0 \leqslant \delta < 1/2$). Then there is an IND-P adversary $\mathcal{I}$ with min-entropy $\mu - log(1 - 2\delta) + 1$ such that*

$$\mathbf{Adv}^{priv-p}_{\Pi,\mathcal{B}'}(k) \leqslant \mathbf{Adv}^{ind-p}_{\Pi,\mathcal{I}}(k) + (\frac{1}{2} + \delta)^k$$

its running time is the time at most k executions of $\mathcal{B}$.

Proof. Algorithm $\mathcal{I}$ works as follows:

Algorithm $\mathcal{I}_m(1^k, b)$	**Algorithm** $\mathcal{I}_g(1^k, pk, \mathbf{c})$
For $i = 1, \ldots, n$ do	$g \leftarrow \mathcal{B}'_g(1^k, pk, \mathbf{c})$
$(\mathbf{x}, t) \leftarrow \mathcal{B}'_m(1^k)$	return g
if $t = b$ then return $\mathbf{x}$	
return $\mathbf{x}$	

Let F_d denotes the event that $d = b$ when the final return statement is executed. B'_d denotes the event $\mathbf{Exp}^{priv-p-d}_{\Pi,\mathcal{B}'}(k) \Rightarrow true$ ($d \in \{0, 1\}$), similarly I_b denotes $\mathbf{Exp}^{ind-p-b}_{\Pi,\mathcal{I}}(k) \Rightarrow true$. Then

$$\begin{aligned}
\mathbf{Adv}^{ind-p}_{\Pi,\mathcal{I}}(k) &= Pr[I_1] - Pr[I_0] \\
&= Pr[E_1] \cdot Pr[I_1|E_1] - Pr[E_0] \cdot Pr[I_0|E_0] + Pr[\bar{E}] \cdot Pr[I_1|\bar{E}] - Pr[\bar{E}] \cdot Pr[I_0|\bar{E}] \\
&= (1 - (\frac{1}{2} + \delta)^k) \cdot Pr[B_1^{'}] - (1 - (\frac{1}{2} - \delta)^k) \cdot Pr[B_0^{'}] \\
&\geqslant \mathbf{Adv}^{priv-p}_{\Pi,\mathcal{B}^{'}}(k) - (\frac{1}{2} + \delta)^k
\end{aligned}$$

Next we explain that the min-entropy of $\mathcal{I}$ is $\mu - log(1-2\delta) + 1$.

Let $b \in \{0,1\}$, Denote $Pr[t = b : (\mathbf{x}, t) \leftarrow \mathcal{B}^{'}_m(1^k)]$ by $P_{B'}(b)$, $Pr[\mathbf{x}[i] = x : (\mathbf{x}, t) \leftarrow \mathcal{B}^{'}_m(1^k)]$ by $P_{B'}(x, i)$, and $Pr[\mathbf{x}[i] = x \wedge t = b : (\mathbf{x}, t) \leftarrow \mathcal{B}^{'}_m(1^k)]$ by $P_{B'}(x, i, b)$, According to the definition of min-entropy of $\mathcal{B}^{'}$, we have

$$\begin{aligned}
Pr[\mathbf{x}[i] = x : (\mathbf{x}, t) \leftarrow \mathcal{I}_m(1^k, b)] &= \sum_{i=1}^{k-1} P_{B'}(\bar{b})^{i-1} P_{B'}(x, i, b) + P_{B'}(\bar{b})^{k-1} P_{B'}(x, i, \bar{b}) \\
&= P_{B'}(x, i, b) \frac{1 - P_{B'}(\bar{b})^k}{P_{B'}(b)} + P_{B'}(\bar{b})^{k-1} P_{B'}(x, i, \bar{b}) \\
&\leqslant \frac{1}{P_{B'}(b)} \cdot (P_{B'}(x, i, b) + P_{B'}(x, i, \bar{B})) \\
&= \frac{1}{P_{B'}(b)} \cdot P_{B'}(x, i) \\
&= \frac{1}{1/2 - \delta} \cdot 2^{-\mu}
\end{aligned}$$

Theorem 1 follows by combining Propositions 1, 2, and 3 with $\delta = 1/6$.

Forward-Secure Hierarchical Predicate Encryption

Juan Manuel González Nieto[1], Mark Manulis[2], and Dongdong Sun[1]

[1] Queensland University of Technology, Brisbane QLD 4001, Australia
j.gonzaleznieto@qut.edu.au, dd.sun@student.qut.edu.au
[2] Department of Computer Science, University of Surrey, United Kingdom
mark@manulis.eu

Abstract. Secrecy of decryption keys is an important pre-requisite for security of any encryption scheme. *Forward Security (FS)* reduces damage from compromised keys by guaranteeing confidentiality of messages that were encrypted prior to the compromise event. In this paper we introduce FS to the powerful setting of *Hierarchical Predicate Encryption (HPE)*, proposed by Okamoto and Takashima (Asiacrypt 2009). Our FS-HPE scheme guarantees forward security for plaintexts and for attributes that are hidden in HPE ciphertexts. It further allows delegation of decrypting abilities at any point in time, independent of FS time evolution. It realizes zero-inner-product predicates and is proven adaptively secure under standard assumptions. As the "cross-product" approach taken in FS-HIBE is not directly applicable to the HPE setting, our construction resorts to techniques that are specific to existing HPE schemes and extends them with what can be seen as a reminiscent of binary tree encryption from FS-PKE.

Keywords: Forward Security, Predicate Encryption, Inner Product.

1 Introduction

Predicate Encryption. We focus on the notion of Predicate Encryption (PE), formalized by Katz, Sahai, and Waters [21], building on Hidden Vector Encryption (HVE) [6], and further studied in [22, 24, 25, 27, 28, 33, 34]. In PE schemes users' decryption keys are associated with *predicates* f and ciphertexts encode *attributes* a that are specified during the encryption procedure. A user can successfully decrypt if and only if $f(a) = 1$. Otherwise, the decryption process preserves *plaintext hiding* and thus leaks no information about the encrypted message. Unlike Attribute-Based Encryption (ABE) [2, 11, 15, 29] that imposes the same requirement, PE schemes have a distinguished privacy goal of *attribute hiding* to prevent ciphertext leaking attributes. Existing PE schemes typically realize concrete predicates f. For example, predicates based on the inner product of vectors (over a field or ring) — Inner-Product Encryption (IPE) [21] — are particularly powerful since they can be used to evaluate a large class of predicates, including conjunctions or disjunctions of equality tests, comparisons, and

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 83–101, 2013.

subset tests, or more generally, arbitrary CNF or DNF formulae. In IPE schemes, attributes are represented by a vector $\overrightarrow{y}$ while the choice of another vector $\overrightarrow{x}$ defines the predicate $f_{\overrightarrow{x}}$ such that $f_{\overrightarrow{x}}(\overrightarrow{y}) = 1$ iff the inner product $\overrightarrow{x} \cdot \overrightarrow{y} = 0$. While the original scheme from [21] was proven to be selectively secure under non-standard assumptions, recent result of Lewko *et al.* [22] provided more sophisticated PE constructions achieving (stronger) adaptive security under non-standard assumptions. Furthermore, Okamoto and Takashima [25] investigated Functional Encryption that is adaptive security under standard assumptions. In [22,24] the authors also explored constructions of Hierarchical PE (HPE) schemes providing their users with the ability to delegate their decryption keys down the hierarchy by restricting predicates associated to the delegated keys and by this restricting the abilities of lower-level users to decrypt. It should be noted that existing PE (and ABE) schemes emerged from Identity-Based Encryption (IBE) [5,32] and the majority of these schemes are pairing-based.

Forward Security. Forward Security (FS) offers meaningful protection in cryptographic applications with long-term (aka. static) private keys in the unfortunate case when these keys become compromised. Being a standard requirement in authenticated key exchange protocols, where it also takes its origin [12,16], forward security has further been explored in digital signatures [1,18] and in public key encryption (PKE) [8]; see [18] for a nice survey and strong motivation of forward security. The concept of *time evolution* is central to forward security since from the moment the private key is exposed the intended security goals can no longer be guaranteed and the key must be changed. FS aims to tame potential damage by offering protection with respect to *earlier* time periods. For example, in forward secure digital signatures signing keys that are exposed in one time period cannot be used to forge signatures related to prior time periods. Similarly, in the case of forward secure encryption decryption keys used in one time period cannot be used to decrypt ciphertexts generated in the past.

The first forward-secure PKE scheme, due to Canetti, Halevi, and Katz [8], was built from the technical tool, called *binary tree encryption* [20], which in turn is implied by Hierarchical IBE (HIBE) [14,17] by considering identities as nodes of the tree and restricting the intermediate nodes to have exactly two descendants: a parent node with identity string $\mathtt{id} \in \{0,1\}^{\ell}$ is split into two child nodes with identities $\mathtt{id}0$, $\mathtt{id}1 \in \{0,1\}^{\ell+1}$. For each node $\mathtt{id}$ there exists a secret key $SK_{\mathtt{id}}$, which can be used to derive secret keys $SK_{\mathtt{id}0}$ and $SK_{\mathtt{id}1}$ in a one-way fashion. The intuition behind FS-PKE is to split the entire lifetime of the scheme into N time periods and construct a binary tree with depth $\log N$, where each node corresponds to a unique time period. In order to encrypt a message for some time period $i \in [1, N]$ one uses the master public key of HIBE and the identity string $\mathtt{id}_i$ of the node i. At any period $i \in [1, N]$ the private decryption key of the user contains the secret key $SK_{\mathtt{id}_i}$ as well as secret keys for all right siblings of the nodes on the path from the root to node i. The latter keys can be used to derive secret keys $SK_{\mathtt{id}_j}$ for all subsequent periods $j \in [i, N]$. The actual FS property is obtained by erasing $SK_{\mathtt{id}_i}$ (and all secret keys that can be used to derive it) from the private key upon transition to period $i + 1$.

These ideas were extended by Yao *et al.* [36] to obtain FS in the identity-based setting. More precisely, they came up with a forward-secure HIBE (FS-HIBE) constructed via a "cross-product" combination of two HIBE schemes, in the random oracle model. Boneh, Boyen, and Goh [3] offered more efficient FS-HIBE constructions, with selective security in the standard model and with adaptive security in the random oracle model. The first adaptively secure FS-HIBE scheme in the standard model is due to Lewko and Waters [23]. As mentioned by Boyen and Waters [7] and also explored in [10,13,30,31,34] FS is also achievable for *anonymous* HIBE systems, whose ciphertexts hide the (hierarchy of) identities for which messages were encrypted. Since HIBE generalizes IBE (anonymous) FS-HIBE covers (anonymous) FS-IBE.

Forward Security in ABE/PE. A message encrypted with an ABE/PE scheme can potentially be decrypted by many users. Exposure of some user's private key in these schemes is likely to cause more damage in comparison to PKE or IBE schemes since the adversary could obtain messages that were encrypted for more than one user. Adding forward security to ABE/PE schemes is thus desirable to alleviate this problem. A naïve approach, i.e., to change all keys (incl. public ones) for each new time period, has already been ruled out as being impractical in PKE and IBE schemes, and it seems even more complicated in the ABE/PE setting. In this work we formalize and construct the first forward-secure hierarchical predicate encryption (FS-HPE). Since HPE includes PE/ABE [22,24], our FS-HPE scheme also implies constructions of first forward secure ABE/PE schemes.

Although forward-secure HIBE constructions exist, formalizing and designing FS-HPE is challenging due to a number of advanced properties that must be considered. In HPE schemes predicates (and by this indirectly private keys) are organized in a hierarchy — any ciphertext that can be decrypted by a low-level predicate must also be decryptable by a high-level predicate but the converse may not be true. In contrast to HIBE, where delegation is performed by extending the parent identity with a substring, predicates in HPE have more complex structures and their delegation requires different techniques. Moreover, predicates should be delegatable at any period in time, irrespective of time evolution for FS. Another aspect is that encryption of messages in forward-secure HPE must be possible only using the master public key, the set of attributes, and the current time period, without having á priori knowledge of predicates at any level of the hierarchy, whereas in FS-HIBE schemes encryption is performed with respect to a given identity at one of the hierarchy levels. We note that obtaining forward security in HPE schemes by applying techniques from existing FS-PKE [8] and FS-HIBE [36] results in a number of obstacles. For example, a "cross-product" combination of two HPE schemes [22,24], akin to the case of two HIBE schemes for FS-HIBE in [36], seems not feasible due to the unique delegation and randomization mechanisms used in those HPE schemes. Finally, an FS-HPE scheme should still provide attribute-hiding, which could be threatened if (public) time periods for FS are mixed up with attributes during the encryption.

1.1 Our Contributions

FS-HPE: Model and Scheme. We formalize and design the first forward-secure hierarchical predicate encryption (FS-HPE) scheme, for zero-inner-product predicates [21]. Our scheme is secure (adaptively attribute-hiding) in the standard model under the well-known Decision Linear (DLIN) assumption [4] in bilinear groups of prime order. We first present a new syntax and security definitions that are specific to FS-HPE, in particular definition of attribute hiding had to be extended in order to account for FS, in a more complex way than in FS-HIBE definitions from [23,36], as explained in Section 3.3. Our FS-HPE scheme offers some desirable properties: time-independent delegation of predicates (to support dynamic behavior for delegation of decrypting rights to new users), local update for users' private keys (i.e., no master authority needs to be contacted), forward security, and the scheme's encryption process doesn't require knowledge of predicates at any level including when those predicates join the hierarchy. Considering the relationships amongst the encryption flavors, we can restrict our scheme to level-1 hierarchy and obtain first adaptively-secure FS-PE/ABE construction, or we can set the inner-product predicate to perform the equality test, in which case we would obtain the first adaptively-secure anonymous FS-HIBE scheme under the basic DLIN assumption (as an alternative to [10] that works in bilinear groups of composite order and requires new hardness assumptions).

Techniques. Our FS-HPE scheme is built based on the dual system encryption approach introduced by Waters [35] and uses the concept of dual pairing vector spaces (DPVS) of Okamoto and Takashima [24]. Techniques underlying forward security of the scheme can be seen as reminiscent of binary tree encryption [8] that was invented for FS-PKE and doesn't apply immediately to the more complex HPE setting. We had to resort to those techniques and modify them for integration with HPE since obtaining FS-HPE in a more direct way, e.g. by adopting the "cross-product" idea from [36], seems not feasible with existing HPE constructions [22,24]. On a high level, we modify the existing HPE scheme from [22] and combine two of its instances in a non-trivial way to achieve a FS-HPE scheme. One of the HPE schemes handles predicate/attibute hierarchy while another one is used for maintaining time periods using the concept behind binary tree encryption [8]. The modification of the scheme in [22] is necessary to prove security the stringent security definitions involving FS. The combination of two schemes is non-trivial due to the delegation and randomization components inherited from HPE. Our scheme perfectly synchronizes all private key components (decryption, delegation and randomization) from both HPE instances. These components are updated at each new time period and they are also used for time-independent delegation of predicates. We apply game-hopping proofs, following the general proof strategy from [25], i.e. we first define several hard problems and prove that security of our scheme relies on them, then we prove that those hard problems can individually be used to solve the DLIN problem.

2 Background on Dual Pairing Vector Spaces and Complexity Assumption

GROUPS. Let $\mathcal{G}_{\mathsf{bpg}}$ be an algorithm that on input a security parameter 1^λ outputs a description of the symmetric bilinear group setting $(q, \mathbb{G}, \mathbb{G}_T, G, e)$ where q is a prime, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order q, G is the generator of $\mathbb{G}$, e is a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, i.e., $e(sG, tG) = e(G, G)^{st}$ and $e(G, G) \neq 1$. We also define cyclic additive group $\mathbb{G}$ and multiplicative group $\mathbb{G}_T$ of order q.

VECTOR SPACES. Let $\mathbb{V} = \overbrace{\mathbb{G} \times \cdots \times \mathbb{G}}^{N}$ be a *vector space* and each element in $\mathbb{V}$ be expressed by *N-dimensional vector*. $\boldsymbol{x} = (x_1 G, \ldots, x_N G)$ ($x_i \in \mathbb{F}_q$ for $i = 1, \ldots, N$). The *canonical base* $\mathbb{A}$ of $\mathbb{V}$ is $\mathbb{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N)$, where $\boldsymbol{a}_1 = (G, 0, \ldots, 0)$, $\boldsymbol{a}_2 = (0, G, 0, \ldots, 0), \ldots, \boldsymbol{a}_N = (0, \ldots, 0, G)$. Given two vectors $\boldsymbol{x} = (x_1 G, \ldots, x_N G) = x_1 \boldsymbol{a}_1 + \cdots + x_N \boldsymbol{a}_N \in \mathbb{V}$ and $\boldsymbol{y} = (y_1 G, \ldots, y_N G) = y_1 \boldsymbol{a}_1 + \cdots + y_N \boldsymbol{a}_N \in \mathbb{V}$, where $\overrightarrow{x} = (x_1, \ldots, x_N)$ and $\overrightarrow{y} = (y_1, \ldots, y_N)$, the pairing operation is defined as $e(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{N} e(x_i G, y_i G) = e(G, G)^{\sum_{i=1}^{N} x_i y_i} = g_T^{\overrightarrow{x}\,\overrightarrow{y}} \in \mathbb{G}_T$.

Definition 1 (Dual Pairing Vector Space (DPVS) [24]). *Let $(q, \mathbb{G}, \mathbb{G}_T, G, e)$ be a symmetric bilinear pairing group. A* Dual Pairing Vector Space *$(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$, generated by an algorithm denoted $\mathcal{G}_{\mathsf{dpvs}}$, is a tuple containing a prime q, an N-dimensional vector space $\mathbb{V}$ over $\mathbb{F}_q$, a cyclic group $\mathbb{G}_T$ of order q, a canonical base $\mathbb{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N)$ of $\mathbb{V}$, and a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ that satisfy the following conditions:*

1. NON-DEGENERATE BILINEAR PAIRING: *There exists a polynomial-time computable non-degenerate bilinear pairing $e(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{N} e(G_i, H_i)$ where $\boldsymbol{x} = (G_1, \ldots, G_N) \in \mathbb{V}$ and $\boldsymbol{y} = (H_1, \ldots, H_N) \in \mathbb{V}$. This is non-degenerate bilinear pairing i.e., $e(s\boldsymbol{x}, t\boldsymbol{y}) = e(\boldsymbol{x}, \boldsymbol{y})^{st}$ and if $e(\boldsymbol{x}, \boldsymbol{y}) = 1$ for all $\boldsymbol{y} \in \mathbb{V}$, then $\boldsymbol{x} = \boldsymbol{0}$.*
2. DUAL ORTHONORMAL BASES: *$\mathbb{A}$ and e satisfy that $e(\boldsymbol{a_i}, \boldsymbol{a_j}) = g_T^{\delta_{i,j}}$ for all i and j, where $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise, and $g_T \neq 1 \in \mathbb{G}_T$.*
3. DISTORTION MAPS: *Linear transformations $\phi_{i,j}$ on $\mathbb{V}$ s.t. $\phi_{i,j}(\boldsymbol{a}_j) = \boldsymbol{a}_i$ and $\phi_{i,j}(\boldsymbol{a}_k) = \boldsymbol{0}$ if $k \neq j$ are polynomial-time computable. We call $\phi_{i,j}$ "distortion maps".*

ORTHONORMAL BASES. Let $\mathbb{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N)$ be a basis of vector space $\mathbb{V}$ which is obtained from its canonical basis $\mathbb{A}$ using a uniformly chosen linear transformation $\Lambda = (\lambda_{i,j}) \xleftarrow{U} GL(N, \mathbb{F}_q)$. Note that $GL(N, \mathbb{F}_q)$ creates a matrix of size $N \times N$ in which each element is uniformly selected from $\mathbb{F}_q$ such that $\boldsymbol{b}_i = \sum_{j=1}^{N} \lambda_{i,j} \boldsymbol{a}_j$, for $i = 1, \ldots, N$. Similarly, let $\mathbb{B}^* = (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_N^*)$ be another basis of $\mathbb{V}$ which is also obtained from $\mathbb{A}$ using $\mu_{i,j} = (\Lambda^T)^{-1}$ as $\boldsymbol{b}_i^* = \sum_{j=1}^{N} \mu_{i,j} \boldsymbol{a}_j$, for $i = 1, \ldots, N$. It can be shown that $e(\boldsymbol{b}_i, \boldsymbol{b}_j^*) = g_T^{\delta_{i,j}}$, where $\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ if $i \neq j$. That is $\mathbb{B}$ and $\mathbb{B}^*$ are dual orthonormal bases of $\mathbb{V}$. In our

scheme we will use the following probabilistic algorithm $\mathcal{G}_{\mathsf{ob}}$ to generate group and DPSV parameters and the two dual orthonormal bases:

$$\begin{aligned}
\mathcal{G}_{\mathsf{ob}}(1^\lambda\ ,\ \overrightarrow{n} = (d; n_1, \ldots, n_d)) : \mathsf{param}_{\mathbb{G}} &= (q, \mathbb{G}, \mathbb{G}_T, G, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bpg}}(1^\lambda), \\
&\psi \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times, N_0 = 5, N_t = 3n_t + 1 \text{ for } t = 1, \ldots, d; \\
&\text{For } t = 0, \ldots, d: \\
&\quad \mathsf{param}_{\mathbb{V}_t} = (q, \mathbb{V}_t, \mathbb{G}_T, \mathbb{A}_t, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, N_t, \mathsf{param}_{\mathbb{G}}), \\
&\quad \varLambda^{(t)} = (\lambda_{i,j}^{(t)}) \xleftarrow{\mathsf{U}} GL(N_t, \mathbb{F}_q),\ (\mu_{i,j}^{(t)}) = \psi \cdot (\varLambda^{(t)^T})^{-1}, \\
&\quad \boldsymbol{b}_i{}^{(t)} = \sum_{j=1}^{N_t} \lambda_{i,j}^{(t)} \boldsymbol{a}_j^{(t)} \text{ for } i = 1, \ldots, N_t, \mathbb{B}^{(t)} = (\boldsymbol{b}_1^{(t)}, \ldots, \boldsymbol{b}_{N_t}^{(t)}), \\
&\quad \boldsymbol{b}_i^{*(t)} = \sum_{j=1}^{N_t} \mu_{i,j}^{(t)} \boldsymbol{a}_j^{(t)} \text{ for } i = 1, \ldots, N_t, \mathbb{B}^{*(t)} = (\boldsymbol{b}_1^{*(t)}, \ldots, \boldsymbol{b}_{N_t}^{*(t)}), \\
&g_T = e(G, G)^\psi, \mathsf{param}_{\overrightarrow{n}} = (\{\mathsf{param}_{\mathbb{V}_t}\}_{t=0,\ldots,d}, g_T), \\
&\text{Output } (\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}^{(t)}, \mathbb{B}^{*(t)}\}_{t=0,\ldots,d}).
\end{aligned}$$

Note that $g_T = e(\boldsymbol{b}_i{}^{(t)}, \boldsymbol{b}_i^{*(t)})$ for $t = 0, \ldots, d$; $i = 1, \ldots, N_t$.

Definition 2 (Decisional Linear Assumption (DLIN) [4]). *The DLIN problem is to decide on bit $\beta \in \{0, 1\}$, given the output $(\mathsf{param}_{\mathbb{G}}, G, aG, bG, acG, bdG, Y_\beta)$ of the probabilistic algorithm*

$$\begin{aligned}
\mathcal{G}_\beta^{\mathrm{DLIN}}(1^\lambda) : \mathsf{param}_{\mathbb{G}} &= (q, \mathbb{G}, \mathbb{G}_T, G, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bpg}}(1^\lambda), a, b, c, d \xleftarrow{\mathsf{U}} \mathbb{F}_q, \\
&Y_0 = (c + d)G,\ Y_1 \xleftarrow{\mathsf{U}} \mathbb{G},\ \beta \xleftarrow{\mathsf{U}} \{0, 1\}; \\
&\text{Output } (\mathsf{param}_{\mathbb{G}}, G, aG, bG, acG, bdG, Y_\beta).
\end{aligned}$$

The advantage $\mathsf{Adv}_{\mathcal{D}}^{\mathrm{DLIN}}(\lambda)$ of a probabilistic polynomial-time DLIN solver $\mathcal{D}$ is defined as follows:

$$\left| \mathsf{Pr}\left[\mathcal{D}(1^\lambda, \varpi) \to 1 \;\middle|\; \varpi \xleftarrow{\mathsf{R}} \mathcal{G}_0^{\mathrm{DLIN}}(1^\lambda)\right] - \mathsf{Pr}\left[\mathcal{D}(1^\lambda, \varpi) \to 1 \;\middle|\; \varpi \xleftarrow{\mathsf{R}} \mathcal{G}_1^{\mathrm{DLIN}}(1^\lambda)\right] \right|.$$

The DLIN assumption states that for any $\mathcal{D}$ this advantage is negligible in λ.

3 Forward-Secure Hierarchical Predicate Encryption

In this section we present our model for forward secure hierarchical predicate encryption (FS-HPE). First, we highlight the idea behind FS-HPE concept and introduce some notations. In FS-HPE private keys are associated with predicate vectors and evolve over the time. At any time period i a user may join the hierarchy and receive delegated private keys. These keys are computed by the parent user for time period i and together with further secret information that

is necessary to derive private keys for later time periods is handed over to the joined user. Once the user receives this secret information, at the end of each period the user updates his private key locally and erases secrets that are no longer needed. Additionally, at any time $j \geq i$ the user may delegate its private key down the hierarchy without contacting its parent. In any time period i a message can be encrypted using public parameters, the attribute vectors, and i. In order to decrypt for time period i users must possess private keys satisfying attributes from the ciphertext for that time.

3.1 Notations

Time Period. Let the total number of time periods $N = 2^\kappa$, where $\kappa \in \mathbb{N}$.

Hierarchical Inner-Product Predicate Encryption. We borrow some notations from [22] to describe our HPE with inner-product predicates. Let $\overrightarrow{\mu} = (n; d, \mu_1, \ldots, \mu_d)$ be a tuple of positive integers such that $\mu_0 = 0 < \mu_1 < \mu_2 < \cdots < \mu_d = n$. We call $\overrightarrow{\mu}$ a *format of hierarchy of depth d attribute spaces*. With Σ_l, $l = 1, \ldots, d$ we denote *attribute sets* and each $\Sigma_l = \mathbb{F}_q^{\mu_l - \mu_{l-1}} \setminus \{0\}$. A *hierarchical attribute* $\Sigma = \cup_{l=1}^{d}(\Sigma_1 \times \ldots \times \Sigma_l)$ is defined using the disjoint union. For $\overrightarrow{v}_i \in \mathbb{F}_q^{\mu_i - \mu_{i-1}} \setminus \{\overrightarrow{0}\}$, a hierarchical attribute $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) \in \Sigma$ is said to satisfy a *hierarchical predicate* $f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}$ iff $l \leq h$ and $\overrightarrow{x}_i \cdot \overrightarrow{y}_i = 0$ for $1 \leq i \leq l$, which we denote as $f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) = 1$. The space of hierarchical predicates is $\mathcal{F} = \{f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)} | \overrightarrow{x}_i \in \mathbb{F}_q^{\mu_i - \mu_{i-1}} \setminus \{\overrightarrow{0}\}\}$. We call h (resp. l) the *level* of $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ (resp. $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$). Throughout the paper we will assume that an attribute vector $\overrightarrow{y}_1 = (y_1, \ldots, y_{\mu_1})$ is normalized such that $y_1 = 1$ (note that $\overrightarrow{y}_1$ can be normalized via $(1/y_1) \cdot \overrightarrow{y}_1$, assuming that y_1 is non-zero). By $\overrightarrow{e}_i^{(k)}$ we denote the *canonical basis vector* $(\overbrace{0, \ldots, 0}^{i-1}, 1, \overbrace{0, \ldots, 0}^{n_k - i}) \in \mathbb{F}_q^{n_k}$ for $k = 1, 2$ and $i = 1, \ldots, n_k$.

Keys. We use two notations for secret keys: $sk_{w,(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}$ is the key associated with some prefix w of the bit representation of a time period i and a hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$, whereas $SK_{i,(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}$ denotes the key associated with time i and a hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. That is, $SK_{i,(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)} = \{sk_{i,(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}, sk_{w1,(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)} : w0 \text{ is a prefix of } i\}$.

3.2 Syntax

Definition 3 (FS-HPE). *A forward secure hierarchical predicate encryption scheme is a tuple of five algorithms* (RootSetup, Delegate, Update, Encrypt, Decrypt) *described in the following:*

RootSetup$(1^\lambda, N, \overrightarrow{\mu})$ *This algorithm takes as input a security parameter 1^λ, the total number of time periods N and the format of hierarchy $\overrightarrow{\mu}$. It outputs public parameters of the system, incl. public key PK, and a root secret key $SK_{0,1}$, which is assumed to be known only to the master authority of the hierarchy.*

$\mathsf{Delegate}(SK_{i,l}, i, \overrightarrow{x}_{l+1})$ *This algorithm takes as input a secret key* $SK_{i,l}$ *associated with time* i *on hierarchy level* l *and an* $(l+1)$*-th level predicate vector* $\overrightarrow{x}_{l+1}$*. It outputs the delegated secret key* $SK_{i,l+1}$*. This key is intended for the direct descendant at level* $l+1$*. It is assumed that predicate vector* $\overrightarrow{x}_{l+1}$ *is added to the predicate hierarchy during the time period* i*.*

$\mathsf{Update}(SK_{i,l}, i)$ *This algorithm takes as input a secret key* $SK_{i,l}$ *and the current time period* i*. It outputs an updated secret key* $SK_{i+1,l}$ *for the following time period* $i+1$ *and erases* $SK_{i,l}$*.*

$\mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h), i, M)$ *This algorithm takes as input the public key* PK*, hierarchical attribute vectors* $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$*, a time period* i*, and a message* M *from the associated message space. It outputs a ciphertext* C*. We assume that* i *is included in* C*.*

$\mathsf{Decrypt}(C, SK_{i,l})$ *This algorithm takes as input a ciphertext* C *and a secret key* $SK_{i,l}$ *for the time period* i *and predicate vectors* $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$*. It outputs either a message* M *or the distinguished symbol* $\perp$ *(to indicate a failure).*

Correctness. *For all correctly generated* PK *and* $SK_{i,l}$ *associated with predicate vectors* $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ *and a time period* i*, let* $C \xleftarrow{R} \mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h), i, M)$ *and* $M' = \mathsf{Decrypt}(C, SK_{i,l})$*. Then, if* $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) = 1$ *then* $M = M'$*; otherwise,* $M \neq M'$ *with all but negligible probability.*

3.3 Security Definition

Definition 4. *A* FS-HPE *scheme is* adaptively attribute hiding against chosen plaintext attacks *if for all PPT adversaries* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following game is negligible in the security parameter:*

Setup. RootSetup *algorithm is run by the challenger* $\mathcal{C}$ *to generate public key* PK *and root secret key* $SK_{0,1}$*.* PK *is given to* $\mathcal{A}$*.*

Queries I. $\mathcal{A}$ *may adaptively make a polynomial number of* delegation queries *by asking* $\mathcal{C}$ *to create a secret key for any given time period* i *and hierarchical predicate vectors* $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$*. In response,* $\mathcal{C}$ *computes the secret key* $SK_{i,l}$ *and reveals it to* $\mathcal{A}$*. (Note that* $\mathcal{C}$ *computes* $SK_{i,l}$ *with the help of algorithms* Delegate *and* Update *that it may need to execute several times, i.e. depending on the input time period* i *and hierarchy level* l*.)*

Challenge. $\mathcal{A}$ *outputs its challenge: two attribute vectors* $(Y^{(0)}, Y^{(1)}) = ((\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h^{(0)}}^{(0)}), (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h^{(1)}}^{(1)}))$*, two plaintexts* $(M^{(0)}, M^{(1)})$*, and a time period* I*, such that either* $i > I$*, or* $i \leq I$ *and* $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h^{(0)}}^{(0)}) = f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h^{(1)}}^{(1)}) = 0$ *for each revealed key for* $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$ *and time period* i*.* $\mathcal{C}$ *then flips a random coin* b*. If* $b = 0$ *then* $\mathcal{A}$ *is given* $C = \mathsf{Encrypt}(PK, Y^{(0)}, I, M^{(0)})$ *and if* $b = 1$ *then* $\mathcal{A}$ *is given* $C = \mathsf{Encrypt}(PK, Y^{(1)}, I, M^{(1)})$*.*

Query phase 2. *Repeat the* **Query phase 1** *subject to the restrictions as in the challenge phase.*

Guess. $\mathcal{A}$ *outputs a bit* b'*, and succeeds if* $b' = b$*.*

We define the advantage of $\mathcal{A}$ *as a quantity* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FS\text{-}HPE}}(\lambda) = |\mathsf{Pr}[b = b'] - 1/2|$.

Remark 1. In Definition 4, adversary $\mathcal{A}$ is not allowed to ask a key query for time period i and hierarchical predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ such that $i \leq I$ and $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h^{(b)}}^{(b)}) = 1$ for some $b \in \{0,1\}$, i.e., the queried key is not allowed to decrypt the challenge ciphertext. Recently, Okamoto and Takashima [28] proposed a PE (HPE) which allow such key query, provided that $M^{(0)} = M^{(1)}$. The technique of Okamoto and Takashima [28] can be applied in our scheme to achieve strong security.

Remark 2. In Definition 4, $\mathcal{A}$ may ask delegation queries and obtain the resulting keys. This contrasts slightly with the HPE security definition in [22], where $\mathcal{A}$ may ask the challenger to create and delegate private keys but will not be given any of them, unless it explicitly asks a separate reveal query. This is because HPE in [22] has two algorithms for computing secret keys, either directly (using the master secret key) or through delegation (using secret key of the parent node). In our FS-HPE syntax we compute secret keys through delegation only and in the security definition we are mainly concerned with maintaining time evolution for delegated keys.

Remark 3. Definition 4 can be easily extended to address chosen-ciphertext attacks (CCA) by allowing decryption queries. The usual restriction is that decryption queries cannot be used for the challenge ciphertext. Our CPA-secure FS-HPE scheme from Section 4 can be strengthened to resist CCA by applying the well-known CHK transformation from [9] that uses one-time signatures to authenticate the ciphertext.

4 Our Forward-Secure HPE Scheme

High-Level Description. For simplicity of presentation, our FS-HPE makes use of a version of FS-PKE scheme by Katz [19]. In Katz's scheme, time periods are associated with the leaf nodes of a binary tree while in Canetti *et al.* scheme [8], time periods correspond to all nodes of the tree. Our scheme can also be realized based on the FS-PKE scheme by Canetti et al., which will give faster key update time. We utilize a full binary tree of height κ, whose root is labeled ϵ and all other nodes are labeled recursively: if the label of a node is w, then its left child is $w0$, and its right child is $w1$. Each time period $i \in \{0, \ldots, N-1\}$ corresponds to a leaf identified via the binary representation of i. We denote the k-bit prefix of a d-length word $w = w_1 w_2 \ldots w_d$ by $w|_k$, i.e. $w|_k = w_1 w_2 \ldots w_k$ for $k \leq d$. Let $w|_0 = \epsilon$ and $w = w|_d$.

We use two HPE schemes in parallel. Private keys in each scheme contain three components: decryption, delegation and randomness. Private key of a user contains private keys from both schemes that are linked together using secret sharing. One HPE scheme is used to handle predicate/attribute hierarchy, while the other one is used to handle time evolution. Each of the two HPE schemes is

a modification of the scheme in [22], in a way that allows us to prove attribute-hiding property under more sophisticated conditions involving time evolution. The efficiency of the modified scheme is still comparable to the one in [22], i.e. it increases the ciphertext by an additional component (master component) that is used to combine both HPE schemes and is crucial for the security proof. This change implies that the length of the orthonormal bases grows from $(2n+3)\cdot|G|$ in [22] to $(3n+1)\cdot|G|$ in our scheme, where n is the dimension of the attribute vectors, and $|G|$ is the length of a group element from G.

At time period i, the entity at level l with a hierarchical predicate $(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)$ holds a secret key $SK_{i,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$, denoted for simplicity as $SK_{i,l}$. It contains secret keys $sk_{i,l}$ and $\{sk_{w,l}\}$) for each label w corresponding to a right sibling node (if one exists) on the path from l to the root. We view $sk_{i,l}$ as a decryption key, which is associated with current time i and the predicate $(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)$. The secret keys in $\{sk_{w,l}\}$ contain auxiliary information used to update $sk_{i,l}$ for future time periods and to derive its lower-level predicates. The initial keys $sk_{0,1}$ and $sk_{1,1}$ are computed in the RootSetup algorithm and are associated with the predicate $\overrightarrow{x}_1$. In general, each $sk_{w,l}$ contains three secret components: the *decryption component* $(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}},\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}},\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}})$, the *randomness component* $(\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},1},\ldots,\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},l+1},\boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},1},\ldots,\boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},|w|+1})$ and the *delegation component* $(\boldsymbol{k}^{(1)}_{w,l,\mathsf{del},\mu_l+1},\ldots,\boldsymbol{k}^{(1)}_{w,l,\mathsf{del},n},\boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2|w|+1},\ldots,\boldsymbol{k}^{(2)}_{w,l,\mathsf{del},L})$. All above components are constructed using orthonormal bases $\mathbb{B}^*$ specified in Section 2. There are three different bases in the system. The superscript of each key component denotes its base. $\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}}$ is the mentioned master component that links $\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}$ and $\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}}$ using the secret sharing techniques. In turn, $\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}$ and $\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}}$ are used in respective HPE schemes. If w represents a leaf of the binary tree then the decryption component $(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}},\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}},\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}})$ is used for decryption at time represented by w.

Delegation and randomization of private keys are processed similarly as in [22], except that upon derivation of keys for lower level predicates, we also delegate and randomize their time-dependent part. In particular, the delegation component of the l-th level key is essential to compute the $(l+1)$-th level child key, and the randomness component of the l-th level key is used to re-randomize the latter's coefficients. To handle time hierarchy we deploy "dummy" nodes. Similarly, we will compute the dummy child for predicate hierarchy when time evolves. In this way, all derived keys are re-randomized.

We define a helper algorithm ComputeNext that will be called from RootSetup and Update. Given a secret key $sk_{w,l}$ for node w and a hierarchical predicate $(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)$ it outputs $sk_{(wb),l}$, $b\in\{0,1\}$ for the nodes $w0$ and $w1$ by updating the three components of $sk_{w,l}$. The algorithm Update computes secret keys for the next time period through the internal call to ComputeNext and erases all secret information that was used to derive the key for the current time period. The update procedure involves all three components of the secret key. For example, for a given secret key $SK_{i,l}=(sk_{i,l},\{sk_{w,l}\})$, forward security is achieved

by deleting its component $sk_{i,l}$ and using all three components of $\{sk_{w,l}\}$, where w is now the label of an internal node, to derive $SK_{i+1,l}$ for the following time period with the help of ComputeNext.

In algorithm Delegate, a secret key $sk_{w,l}$ for a string w is used to derive $sk_{w,u}$ for a lower hierarchy level $u > l$ and a hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_u)$ that has restricted capabilities in comparison to $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. As mentioned, the delegation component for hierarchical predicates of $sk_{w,l}$ is essential for the derivation of $sk_{w,u}$, whose coefficients are re-randomized with the randomization component.

The algorithm Encrypt requires only a time period t and a hierarchical attribute $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ to encrypt the message. We note that during encryption attributes $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ are extended with random elements from level $h+1$ down to the leaf, i.e., the scheme encrypts attribute vectors on all levels in the hierarchy instead of encrypting only the input vectors. In this way, parent keys can directly decrypt ciphertexts produced for their children without taking effort to derive child keys first.

The algorithm Decrypt uses the decryption key $sk_{i,l}$, which is associated with time period i and hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. The message is decrypted iff the attributes in the ciphertext satisfy the predicates in the decryption component of the key and the ciphertext is created at time i.

Detailed Description. The five algorithms of our FS-HPE scheme are detailed in the following:
$\mathsf{RootSetup}\big(1^\lambda, N = 2^\kappa, \overrightarrow{\mu} = (n; d, \mu_1, \ldots, \mu_d)\big)$:
Let $\overrightarrow{x}_1$ be the root predicate and let $L = 2\kappa$ and $\overrightarrow{n} = (2; n, L)$. Compute

$$(\mathsf{param}_{\overrightarrow{n}}, \mathbb{B}^{(0)}, \mathbb{B}^{*(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{*(1)}, \mathbb{B}^{(2)}, \mathbb{B}^{*(2)}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n}),$$
$$\widetilde{\mathbb{B}}^{(0)} = (\boldsymbol{b}_1^{(0)}, \boldsymbol{b}_3^{(0)}, \boldsymbol{b}_5^{(0)}),\ \widetilde{\mathbb{B}}^{(1)} = (\boldsymbol{b}_1^{(1)}, \ldots, \boldsymbol{b}_n^{(1)}, \boldsymbol{b}_{3n+1}^{(1)}),\ \widetilde{\mathbb{B}}^{(2)} = (\boldsymbol{b}_1^{(2)}, \ldots, \boldsymbol{b}_L^{(2)}, \boldsymbol{b}_{3L+1}^{(2)}),$$
$$\widetilde{\mathbb{B}}^{*(0)} = (\boldsymbol{b}_1^{*(0)}, \boldsymbol{b}_3^{*(0)}),\ \widetilde{\mathbb{B}}^{*(1)} = (\boldsymbol{b}_1^{*(1)}, \ldots, \boldsymbol{b}_n^{*(1)}),\ \widetilde{\mathbb{B}}^{*(2)} = (\boldsymbol{b}_1^{*(2)}, \ldots, \boldsymbol{b}_L^{*(2)}),$$
$$\widehat{\mathbb{B}}^{*(1)} = (\boldsymbol{b}_{2n+1}^{*(1)}, \ldots, \boldsymbol{b}_{3n}^{*(1)}),\ \widehat{\mathbb{B}}^{*(2)} = (\boldsymbol{b}_{2L+1}^{*(2)}, \ldots, \boldsymbol{b}_{3L}^{*(2)}).$$

The master authority needs to generate not only the secret key associated with the current time period 0 but also secret keys corresponding to the internal nodes on the binary tree whose bit representations are all 0 except for the last bit. The secret key for time 0 and predicate $\overrightarrow{x}_1$ is denoted as $sk_{0^\kappa,1}$. Secret keys that will be used to derive keys for future time periods are denoted as $\{sk_{1,1}, sk_{(01),1}, \ldots, sk_{0^{\kappa-1}1,1}\}$. These values are generated recursively as follows, starting with $sk_{0,1}$ and $sk_{1,1}$.

Computing $sk_{0,1}$: Pick $\psi, \psi', \alpha_{\mathsf{dec}}, \alpha_{\mathsf{dec}}^{(1)}, \alpha_{\mathsf{dec}}^{(2)} \xleftarrow{\mathsf{U}} \mathbb{F}_q$ such that $\alpha_{\mathsf{dec}} = \alpha_{\mathsf{dec}}^{(1)} + \alpha_{\mathsf{dec}}^{(2)}$. Pick $\eta_{\mathsf{dec}}^{(0)}, \beta_{\mathsf{dec},1}^{(1)}, \beta_{\mathsf{dec},1}^{(2)}, \beta_{\mathsf{ran},j,1}^{(1)} (j = 1, 2), \beta_{\mathsf{ran},j,1}^{(2)} (j = 1, 2), \beta_{\mathsf{del},j,1}^{(1)} (j = 1, \ldots, n)$, $\beta_{\mathsf{del},j,1}^{(2)} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\overrightarrow{\eta}_{\mathsf{dec}}^{(2)}, \overrightarrow{\eta}_{\mathsf{ran},j}^{(2)} (j = 1, 2), \overrightarrow{\eta}_{\mathsf{del},j}^{(2)} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathbb{F}_q^L$,

$\overrightarrow{\eta}^{(1)}_{\mathsf{dec}}, \overrightarrow{\eta}^{(1)}_{\mathsf{ran},j}(j=1,2), \overrightarrow{\eta}^{(1)}_{\mathsf{del},j}(j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathbb{F}_q^n$. Compute

$$\begin{aligned}
\boldsymbol{k}^{(0)}_{0,1,\mathsf{dec}} &= (-\alpha_{\mathsf{dec}}, 0, 1, \eta^{(0)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(0)}},\\
\boldsymbol{k}^{(1)}_{0,1,\mathsf{dec}} &= (\alpha^{(1)}_{\mathsf{dec}} \overrightarrow{e}_1^{(1)} + \beta^{(1)}_{\mathsf{dec},1} \overrightarrow{x}_1, 0^{2n-\mu_1}, \overrightarrow{\eta}^{(1)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(1)}},\\
\boldsymbol{k}^{(2)}_{0,1,\mathsf{dec}} &= (\alpha^{(2)}_{\mathsf{dec}}, \beta^{(2)}_{\mathsf{dec},1}, 0^{2L-2}, \overrightarrow{\eta}^{(2)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(2)}},\\
\boldsymbol{k}^{(1)}_{0,1,\mathsf{ran},j} &= (\beta^{(1)}_{\mathsf{ran},j,1} \overrightarrow{x}_1, 0^{2n-\mu_1}, \overrightarrow{\eta}^{(1)}_{\mathsf{ran},j}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j=1,2,\\
\boldsymbol{k}^{(2)}_{0,1,\mathsf{ran},j} &= (0, \beta^{(2)}_{\mathsf{ran},j,1}, 0^{2L-2}, \overrightarrow{\eta}^{(2)}_{\mathsf{ran},j}, 0)_{\mathbb{B}^{*(2)}}, \text{ for } j=1,2,\\
\boldsymbol{k}^{(1)}_{0,1,\mathsf{del},j} &= (\beta^{(1)}_{\mathsf{del},j,1} \overrightarrow{x}_1, 0^{j-\mu_1-1}, \psi, 0^{2n-j}, \overrightarrow{\eta}^{(1)}_{\mathsf{del},j}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j=\mu_1+1,\ldots,n,\\
\boldsymbol{k}^{(2)}_{0,1,\mathsf{del},j} &= (0, \beta^{(2)}_{\mathsf{del},j,1}, 0^{j-3}, \psi', 0^{2L-j}, \overrightarrow{\eta}^{(2)}_{\mathsf{del},j}, 0)_{\mathbb{B}^{*(2)}}, \text{ for } j=3,\ldots,L.
\end{aligned}$$

$$\begin{aligned}
\text{Let } sk_{0,1} = (&\boldsymbol{k}^{(0)}_{0,1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{0,1,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{0,1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{0,1,\mathsf{ran},1}, \boldsymbol{k}^{(1)}_{0,1,\mathsf{ran},2}, \boldsymbol{k}^{(2)}_{0,1,\mathsf{ran},1}, \boldsymbol{k}^{(2)}_{0,1,\mathsf{ran},2},\\
&\boldsymbol{k}^{(1)}_{0,1,\mathsf{del},\mu_1+1}, \ldots, \boldsymbol{k}^{(1)}_{0,1,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{0,1,\mathsf{del},3}, \ldots, \boldsymbol{k}^{(2)}_{0,1,\mathsf{del},L}).
\end{aligned}$$

Computing $sk_{1,1}$: Pick $\pi, \pi', \delta_{\mathsf{dec}}, \delta^{(1)}_{\mathsf{dec}}, \delta^{(2)}_{\mathsf{dec}} \xleftarrow{\mathsf{U}} \mathbb{F}_q$ such that $\delta_{\mathsf{dec}} = \delta^{(1)}_{\mathsf{dec}} + \delta^{(2)}_{\mathsf{dec}}$. Pick $\gamma^{(0)}_{\mathsf{dec}}, \theta^{(1)}_{\mathsf{dec},1}, \theta^{(2)}_{\mathsf{dec},1}, \theta^{(1)}_{\mathsf{ran},j,1}(j=1,2), \theta^{(2)}_{\mathsf{ran},j,1}(j=1,2), \theta^{(1)}_{\mathsf{del},j,1}(j=1,\ldots,n), \theta^{(2)}_{\mathsf{del},j,1}(j=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\overrightarrow{\gamma}^{(1)}_{\mathsf{dec}}, \overrightarrow{\gamma}^{(1)}_{\mathsf{ran},j}(j=1,2), \overrightarrow{\gamma}^{(1)}_{\mathsf{del},j}(j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathbb{F}_q^n$, $\overrightarrow{\gamma}^{(2)}_{\mathsf{dec}}, \overrightarrow{\gamma}^{(2)}_{\mathsf{ran},j}(j=1,2), \overrightarrow{\gamma}^{(2)}_{\mathsf{del},j}(j=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q^L$. Compute

$$\begin{aligned}
\boldsymbol{k}^{(0)}_{1,1,\mathsf{dec}} &= (-\delta_{\mathsf{dec}}, 0, 1, \gamma^{(0)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(0)}},\\
\boldsymbol{k}^{(1)}_{1,1,\mathsf{dec}} &= (\delta^{(1)}_{\mathsf{dec}} \overrightarrow{e}_1^{(1)} + \theta^{(1)}_{\mathsf{dec},1} \overrightarrow{x}_1, 0^{2n-\mu_1}, \overrightarrow{\gamma}^{(1)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(1)}},\\
\boldsymbol{k}^{(2)}_{1,1,\mathsf{dec}} &= (\delta^{(2)}_{\mathsf{dec}} + \theta^{(2)}_{\mathsf{dec},1}, \theta^{(2)}_{\mathsf{dec},1}, 0^{2L-2}, \overrightarrow{\gamma}^{(2)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(2)}},\\
\boldsymbol{k}^{(1)}_{1,1,\mathsf{ran},j} &= (\theta^{(1)}_{\mathsf{ran},j,1} \overrightarrow{x}_1, 0^{2n-\mu_1}, \overrightarrow{\gamma}^{(1)}_{\mathsf{ran},j}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j=1,2,\\
\boldsymbol{k}^{(2)}_{1,1,\mathsf{ran},j} &= (\theta^{(2)}_{\mathsf{ran},j,1}, \theta^{(2)}_{\mathsf{ran},j,1}, 0^{2L-2}, \overrightarrow{\gamma}^{(2)}_{\mathsf{ran},j}, 0)_{\mathbb{B}^{*(2)}}, \text{ for } j=1,2,\\
\boldsymbol{k}^{(1)}_{1,1,\mathsf{del},j} &= (\theta^{(1)}_{\mathsf{del},j,1} \overrightarrow{x}_1, 0^{j-\mu_1-1}, \pi, 0^{2n-j}, \overrightarrow{\gamma}^{(1)}_{\mathsf{del},j}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j=\mu_1+1,\ldots,n,\\
\boldsymbol{k}^{(2)}_{1,1,\mathsf{del},j} &= (\theta^{(2)}_{\mathsf{del},j,1}, \theta^{(2)}_{\mathsf{del},j,1}, 0^{j-3}, \pi', 0^{2L-j}, \overrightarrow{\gamma}^{(2)}_{\mathsf{del},j}, 0)_{\mathbb{B}^{*(2)}}, \text{ for } j=3,\ldots,L.
\end{aligned}$$

$$\begin{aligned}
\text{Let } sk_{1,1} = (&\boldsymbol{k}^{(0)}_{1,1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{1,1,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{1,1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{1,1,\mathsf{ran},1}, \boldsymbol{k}^{(1)}_{1,1,\mathsf{ran},2}, \boldsymbol{k}^{(2)}_{1,1,\mathsf{ran},1}, \boldsymbol{k}^{(2)}_{1,1,\mathsf{ran},2},\\
&\boldsymbol{k}^{(1)}_{1,1,\mathsf{del},\mu_1+1}, \ldots, \boldsymbol{k}^{(1)}_{1,1,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{1,1,\mathsf{del},3}, \ldots, \boldsymbol{k}^{(2)}_{1,1,\mathsf{del},L}).
\end{aligned}$$

Recursion: Use $sk_{0,1}$ to recursively invoke algorithm $\mathsf{ComputeNext}$, i.e. compute

$$(sk_{w00,1}, sk_{w01,1}) = \mathsf{ComputeNext}(PK, sk_{w0,1}, w0), \text{ for all } 1 \le |w0| \le \kappa - 1.$$

Output: Output public key $PK = \big(1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\widetilde{\mathbb{B}}^{(k)}\}_{k=0,1,2}, \widehat{\mathbb{B}}^{*(1)}, \widehat{\mathbb{B}}^{*(2)}, \boldsymbol{b}_4^{*(0)}\big)$ and the root secret key $SK_{0,1} = (sk_{0^\kappa,1}, \{sk_{1,1}, sk_{(01),1}, \ldots, sk_{(0^{\kappa-1}1),1}\})$.

$\mathsf{ComputeNext}(PK, sk_{w,l}, w)$: This is a helper method and is called by the Root Setup and Update algorithms. It takes a public key PK, a secret key $sk_{w,l}$, a node w, and outputs keys $sk_{w0,l}$, $sk_{w1,l}$ for time nodes $w0$ and $w1$ of predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. Parse w as $w_1, \ldots, w_r$, where $|w| = r$. Parse $sk_{w,l}$ as $(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},l+1}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},r+1}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},\mu_l+1}, \ldots, \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},(2r+1)}, \ldots, \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},L})$.

Computing $sk_{w0,l}$: Pick $\psi, \psi', \epsilon^{(0)}_{\mathsf{dec}}, \epsilon^{(1)}_{\mathsf{dec},t}, \epsilon^{(1)}_{\mathsf{ran},j,t} (j = 1, \ldots, l+1), \epsilon^{(1)}_{\mathsf{del},j,t} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t = 1, \ldots, l+1$. Pick $\epsilon^{(2)}_{\mathsf{dec},t}, \sigma_{\mathsf{dec}}, \epsilon^{(2)}_{\mathsf{ran},j,t} (j = 1, \ldots, r+2), \sigma_{\mathsf{ran},j} (j = 1, \ldots, r+2), \epsilon^{(2)}_{\mathsf{del},j,t} (j = 1, \ldots, L), \sigma_{\mathsf{del},j} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t = 1, \ldots, r+1$. $\boldsymbol{r}^{(1)}_{\mathsf{dec}}, \boldsymbol{r}^{(1)}_{\mathsf{ran},j} (j = 1, \ldots, l+1), \boldsymbol{r}^{(1)}_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(1)}_{2n+1}, \ldots, \boldsymbol{b}^{*(1)}_{3n} \rangle$, $\boldsymbol{r}^{(2)}_{\mathsf{dec}}, \boldsymbol{r}^{(2)}_{\mathsf{ran},j} (j = 1, \ldots, r+2), \boldsymbol{r}^{(2)}_{\mathsf{del},j} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(2)}_{2L+1}, \ldots, \boldsymbol{b}^{*(2)}_{3L} \rangle$.
Compute

$$\boldsymbol{k}^{(0)}_{w0,l,\mathsf{dec}} = \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \epsilon^{(0)}_{\mathsf{dec}} \boldsymbol{b}^{*(0)}_4,$$

$$\boldsymbol{k}^{(1)}_{w0,l,\mathsf{dec}} = \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{dec},t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(1)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(2)}_{w0,l,\mathsf{dec}} = \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{dec},t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{dec}} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \boldsymbol{r}^{(2)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(1)}_{w0,l,\mathsf{ran},j} = \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(1)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, l+1,$$

$$\boldsymbol{k}^{(2)}_{w0,l,\mathsf{ran},j} = \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{ran},j} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \boldsymbol{r}^{(2)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, r+2,$$

$$\boldsymbol{k}^{(1)}_{w0,l,\mathsf{del},j} = \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{del},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \psi \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(1)}_{\mathsf{del},j}, \text{ for } j = \mu_l + 1, \ldots, n,$$

$$\boldsymbol{k}^{(2)}_{w0,l,\mathsf{del},j} = \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{del},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{del},j} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \psi' \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(2)}_{\mathsf{del},j},$$
$$\text{for } j = 2(r+1)+1, \ldots, L.$$

Let $sk_{w0,l} = (\boldsymbol{k}^{(0)}_{w0,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w0,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w0,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w0,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w0,l,\mathsf{ran},l+1}, \boldsymbol{k}^{(2)}_{w0,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(2)}_{w0,l,\mathsf{ran},r+2}, \boldsymbol{k}^{(1)}_{w0,l,\mathsf{del},\mu_l+1}, \ldots, \boldsymbol{k}^{(1)}_{w0,l,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{w0,l,\mathsf{del},(2(r+1)+1)}, \ldots, \boldsymbol{k}^{(2)}_{w0,l,\mathsf{del},L})$.

Computing $sk_{w1,l}$: Pick $\tau, \tau', \varepsilon^{(0)}_{\mathsf{dec}}, \varepsilon^{(1)}_{\mathsf{dec},t}, \varepsilon^{(1)}_{\mathsf{ran},j,t} (j = 1, \ldots, l+1), \varepsilon^{(1)}_{\mathsf{del},j,t} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t = 1, \ldots, l+1$. Pick $\varepsilon^{(2)}_{\mathsf{dec},t}, \varsigma_{\mathsf{dec}}, \varepsilon^{(2)}_{\mathsf{ran},j,t} (j = 1, \ldots, r+2), \varsigma_{\mathsf{ran},j} (j = 1, \ldots, r+2), \varepsilon^{(2)}_{\mathsf{del},j,t} (j = 1, \ldots, L), \varsigma_{\mathsf{del},j} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t = 1, \ldots, r+1$. $\boldsymbol{t}^{(1)}_{\mathsf{dec}}, \boldsymbol{t}^{(1)}_{\mathsf{ran},j} (j = 1, \ldots, l+1), \boldsymbol{t}^{(1)}_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(1)}_{2n+1}, \ldots, \boldsymbol{b}^{*(1)}_{3n} \rangle$, $\boldsymbol{t}^{(2)}_{\mathsf{dec}}$,

$\boldsymbol{t}^{(2)}_{\mathsf{ran},j}(j=1,\ldots,r+2), \boldsymbol{t}^{(2)}_{\mathsf{del},j}(j=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(2)}_{2L+1},\ldots,\boldsymbol{b}^{*(2)}_{3L}\rangle$. Compute

$$\boldsymbol{k}^{(0)}_{w1,l,\mathsf{dec}} = \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \varepsilon^{(0)}_{\mathsf{dec}} \boldsymbol{b}^{*(0)}_{4},$$

$$\boldsymbol{k}^{(1)}_{w1,l,\mathsf{dec}} = \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{l+1} \varepsilon^{(1)}_{\mathsf{dec},t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{t}^{(1)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(2)}_{w1,l,\mathsf{dec}} = \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{r+1} \varepsilon^{(2)}_{\mathsf{dec},t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{dec}} \left(\sum_{i=2r+1}^{2r+2} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i} \right) + \boldsymbol{t}^{(2)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(1)}_{w1,l,\mathsf{ran},j} = \sum_{t=1}^{l+1} \varepsilon^{(1)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{t}^{(1)}_{\mathsf{ran},j}, \text{ for } j=1,\ldots,l+1,$$

$$\boldsymbol{k}^{(2)}_{w1,l,\mathsf{ran},j} = \sum_{t=1}^{r+1} \varepsilon^{(2)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{ran},j} \left(\sum_{i=2r+1}^{2r+2} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i} \right) + \boldsymbol{t}^{(2)}_{\mathsf{ran},j}, \text{ for } j=1,\ldots,r+2,$$

$$\boldsymbol{k}^{(1)}_{w1,l,\mathsf{del},j} = \sum_{t=1}^{l+1} \varepsilon^{(1)}_{\mathsf{del},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \tau \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},j} + \boldsymbol{t}^{(1)}_{\mathsf{del},j}, \text{ for } j=\mu_l+1,\ldots,n,$$

$$\boldsymbol{k}^{(2)}_{w1,l,\mathsf{del},j} = \sum_{t=1}^{r+1} \varepsilon^{(2)}_{\mathsf{del},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{del},j} \left(\sum_{i=2r+1}^{2r+2} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i} \right) + \tau' \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},j} + \boldsymbol{t}^{(2)}_{\mathsf{del},j}, \text{ for } j=2(r+1)+1,\ldots,L.$$

Let $sk_{w1,l} = (\boldsymbol{k}^{(0)}_{w1,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w1,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w1,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w1,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w1,l,\mathsf{ran},l+1}, \boldsymbol{k}^{(2)}_{w1,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(2)}_{w1,l,\mathsf{ran},r+2}, \boldsymbol{k}^{(1)}_{w1,l,\mathsf{del},\mu_l+1}, \ldots, \boldsymbol{k}^{(1)}_{w1,l,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{w1,l,\mathsf{del},(2(r+1)+1)}, \ldots, \boldsymbol{k}^{(2)}_{w1,l,\mathsf{del},L})$.

Output: Output $(sk_{w0,l}, sk_{w1,l})$.

$\mathsf{Delegate}(SK_{i,l}, i, \overrightarrow{x}_{l+1} = (x_{\mu_l+1},\ldots,x_{\mu_{l+1}}))$: Parse i as $i_1,\ldots,i_\kappa$ where $\kappa = \log_2 N$. Parse $SK_{i,l}$ as $(sk_{i,l}, \{sk_{i|_{k-1},l}\}_{i_k=0})$. For each $sk_{w,l}$ in $SK_{i,l}$ compute $sk_{w,l+1}$ as follows:
Parse w as $w_1,\ldots,w_r$, where $|w|=r$. Pick $\psi, \psi', \gamma^{(0)}_{\mathsf{dec}}, \gamma^{(1)}_{\mathsf{dec},t}, \gamma^{(1)}_{\mathsf{ran},j,t} (j=1,\ldots,l+2), \gamma^{(1)}_{\mathsf{del},j,t} (j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t=1,\ldots,l+1$. Pick $\gamma^{(2)}_{\mathsf{dec},t}, \sigma_{\mathsf{dec}}, \gamma^{(2)}_{\mathsf{ran},j,t} (j=1,\ldots,r+1), \sigma_{\mathsf{ran},j} (j=1,\ldots,l+2), \gamma^{(2)}_{\mathsf{del},j,t} (j=1,\ldots,L), \sigma_{\mathsf{del},j} (j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t=1,\ldots,r+1$. $\boldsymbol{r}^{(1)}_{\mathsf{dec}}, \boldsymbol{r}^{(1)}_{\mathsf{ran},j} (j=1,\ldots,l+2), \boldsymbol{r}^{(1)}_{\mathsf{del},j} (j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(1)}_{2n+1},\ldots,\boldsymbol{b}^{*(1)}_{3n}\rangle$, $\boldsymbol{r}^{(2)}_{\mathsf{dec}}, \boldsymbol{r}^{(2)}_{\mathsf{ran},j} (j=1,\ldots,r+1), \boldsymbol{r}^{(2)}_{\mathsf{del},j} (j=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(2)}_{2L+1},\ldots,\boldsymbol{b}^{*(2)}_{3L}\rangle$. Compute

$$\boldsymbol{k}^{(0)}_{w,l+1,\mathsf{dec}} = \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \gamma^{(0)}_{\mathsf{dec}} \boldsymbol{b}^{*(0)}_{4},$$

$$\boldsymbol{k}^{(1)}_{w,l+1,\mathsf{dec}} = \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{l+1} \gamma^{(1)}_{\mathsf{dec},t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{dec}} \left(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},i} \right) + \boldsymbol{r}^{(1)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(2)}_{w,l+1,\mathsf{dec}} = \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{r+1} \gamma^{(2)}_{\mathsf{dec},t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(2)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(1)}_{w,l+1,\mathsf{ran},j} = \sum_{t=1}^{l+1} \gamma^{(1)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{ran},j} \left(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},i} \right) + \boldsymbol{r}^{(1)}_{\mathsf{ran},j}, \quad \text{for } j = 1, \ldots, l+2,$$

$$\boldsymbol{k}^{(2)}_{w,l+1,\mathsf{ran},j} = \sum_{t=1}^{r+1} \gamma^{(2)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(2)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, r+1,$$

$$\boldsymbol{k}^{(1)}_{w,l+1,\mathsf{del},j} = \sum_{t=1}^{l+1} \gamma^{(1)}_{\mathsf{del},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{del},j} \left(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},i} \right) + \psi \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(1)}_{\mathsf{del},j}, \quad \text{for } j = \mu_{l+1}+1, \ldots, n,$$

$$\boldsymbol{k}^{(2)}_{w,l+1,\mathsf{del},j} = \sum_{t=1}^{r+1} \gamma^{(2)}_{\mathsf{del},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \psi' \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(2)}_{\mathsf{del},j}, \text{ for } j = 2r+1, \ldots, L.$$

Let $sk_{w,l+1} = (\boldsymbol{k}^{(0)}_{w,l+1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{ran},l+2}, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{ran},r+1}, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{del},\mu_{l+1}+1}, \ldots, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{del},2r+1}, \ldots, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{del},L})$.

Output $SK_{i,l+1} = (sk_{i,l+1}, \{sk_{i|_{k-1}1,l+1}\}_{i_k=0})$ and erase all other information.

$\mathsf{Update}(SK_{i,l}, i)$: This algorithm follows the concept from [8,19] to compute a private key for the next time period $i+1$. Parse i as $i_1, \ldots, i_\kappa$ where $|i| = \kappa$. Parse $SK_{i,l}$ as $(sk_{i,l}, \{sk_{i|_{k-1}1,l}\}_{i_k=0})$. Erase $sk_{i,l}$. If $i_\kappa = 0$, simply output the remaining keys as the key $SK_{(i+1),l}$ for the next period. Otherwise, let $\tilde{k}$ be the largest value such that $i_{\tilde{k}} = 0$. Let $i' = i|_{\tilde{k}-1}1$. Using $sk_{i',l}$, which is part of $SK_{i,l}$, recursively apply algorithm $\mathsf{ComputeNext}$ to generate keys $sk_{(i'0^d1),l}$ for $0 \leq d \leq l - \tilde{k} - 1$ and $sk_{(i'0^{d-\tilde{k}},l)}$. (The key $sk_{(i'0^{d-\tilde{k}},l)}$ will be used for decryption in the next time period $i+1$, whereas other generated secret keys will be used to compute private key of the next period.) Erase $sk_{i',l}$ and output the remaining keys as $SK_{(i+1),l}$.

$\mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) = ((y_1, \ldots, y_{\mu_1}), \ldots, (y_{\mu_{h-1}+1}, \ldots, y_{\mu_h})), i, M \in \mathbb{G}_T)$:

Parse i as $i_1, \ldots, i_\kappa$. Pick $(\overrightarrow{y}_{h+1}, \ldots, \overrightarrow{y}_d) \xleftarrow{\cup} \mathbb{F}_q^{\mu_{h+1}-\mu_h} \times \ldots \times \mathbb{F}_q^{n-\mu_{d-1}}$, $\delta, \zeta, \varphi, \varphi^{(1)}, \varphi^{(2)} \xleftarrow{\cup} \mathbb{F}_q$, compute

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (\delta, 0, \zeta, 0, \varphi)_{\mathbb{B}^{(0)}}, \\
\boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_d), 0^{2n}, \varphi^{(1)})_{\mathbb{B}^{(1)}}, \\
\boldsymbol{c}^{(2)} &= (\delta((1, -i_1), \ldots, (1, -i_\kappa)), 0^{2L}, \varphi^{(2)})_{\mathbb{B}^{(2)}}, \\
\boldsymbol{c}^{(M)} &= g_T^{\zeta} M.
\end{aligned}$$

Output ciphertext $C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)})$.

$\mathsf{Decrypt}(C, SK_{i,l})$: Parse ciphertext C as $(\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)})$ and secret key $SK_{i,l}$ as $(sk_{i,l}, \{sk_{i|_{k-1}1,l}\}_{i_k=0})$. Use $sk_{i,l}$ to decrypt and output

$$M = \frac{\boldsymbol{c}^{(M)}}{e(\boldsymbol{c}^{(0)}, \boldsymbol{k}^{(0)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(1)}, \boldsymbol{k}^{(1)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(2)}, \boldsymbol{k}^{(2)}_{i,l,\mathsf{dec}})}.$$

Correctness. To see why the scheme is correct, let C and $SK_{i,l}$ be as above. If $\overrightarrow{x}_i \cdot \overrightarrow{y}_i = 0$ for $1 \le i \le l$, and C and $SK_{i,l}$ are encoded with the same time period i then M can be recovered by computing $\boldsymbol{c}^{(M)} / e(\boldsymbol{c}^{(0)}, \boldsymbol{k}^{(0)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(1)}, \boldsymbol{k}^{(1)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(2)}, \boldsymbol{k}^{(2)}_{i,l,\mathsf{dec}})$, since

$$e(\boldsymbol{c}^{(0)}, \boldsymbol{k}^{(0)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(1)}, \boldsymbol{k}^{(1)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(2)}, \boldsymbol{k}^{(2)}_{i,l,\mathsf{dec}}) = g_T^{-\alpha_{\mathsf{dec}}\delta+\zeta} g_T^{\alpha^{(1)}_{\mathsf{dec}}\delta} g_T^{\alpha^{(2)}_{\mathsf{dec}}\delta} = g_T^{-\alpha\delta+\zeta} g_T^{\alpha\delta}.$$

Remark 4. Recently, Okamoto and Takashima [27] proposed a PE with short secret keys. We note that their scheme can be easily applied to our system to achieve better efficiency in key size. Moreover, in an updated version [26], Okamoto and Takashima devised a payload-hiding HIPE with compact secret keys. The technique [26] can also be applied in our system, specifically, for the time period subtree.

Theorem 1. *Our FS-HPE scheme is adaptively attribute-hiding against chosen plaintext attacks under the DLIN assumption. For any adversary $\mathcal{A}$, there exists a PPT machine $\mathcal{D}$ such that for any security parameter λ,*

$$\mathsf{Adv}^{\mathsf{FS\text{-}HPE}}_{\mathcal{A}}(\lambda) \le (2\nu(\kappa+1)(n+L+1)+1)\mathsf{Adv}^{\mathrm{DLIN}}_{\mathcal{D}}(\lambda) + \psi,$$

where ν is the maximum number of $\mathcal{A}$'s key queries, κ is the depth of the time tree, and $\psi = (20\nu(\kappa+1)(n+L+1)+9)/q$.

The proof of Theorem 1 is provided in the full version.

5 Conclusion

In this paper, we introduced the notion of forward security to the powerful setting of hierarchical predicate encryption. The resulting FS-HPE scheme offers

time-independent delegation of predicates, autonomous update for users' private keys, and its encryption process doesn't require knowledge of time periods at which particular predicates joined the predicate hierarchy. The scheme is forward-secure and adaptively attribute-hiding under chosen plaintext attacks, under the DLIN assumption in the standard model. Using level-1 hierarchy we obtain first adaptively-secure FS-PE/ABE construction. By setting the inner-product predicate to perform the equality test, we achieve the first adaptively-secure anonymous FS-HIBE scheme under the DLIN assumption.

References

1. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society (2007)
3. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
4. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
5. 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)
6. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
7. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
8. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
9. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
10. De Caro, A., Iovino, V., Persiano, G.: Fully Secure Anonymous HIBE and Secret-Key Anonymous IBE with Short Ciphertexts. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 347–366. Springer, Heidelberg (2010)
11. Chase, M.: Multi-authority Attribute Based Encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
12. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Designs, Codes and Cryptography 2, 107–125 (1992)
13. Ducas, L.: Anonymity from Asymmetry: New Constructions for Anonymous HIBE. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 148–164. Springer, Heidelberg (2010)
14. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)

15. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS 2006, pp. 89–98. ACM (2006)
16. Günther, C.G.: An Identity-Based Key-Exchange Protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)
17. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
18. Itkis, G., Reyzin, L.: Forward-Secure Signatures with Optimal Signing and Verifying. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 332–354. Springer, Heidelberg (2001)
19. Katz, J.: A forward-secure public-key encryption scheme. Cryptology ePrint Archive, Report 2002/060 (2002), http://eprint.iacr.org/
20. Katz, J.: Binary Tree Encryption: Constructions and Applications. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 1–11. Springer, Heidelberg (2004)
21. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
22. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
23. Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
24. Okamoto, T., Takashima, K.: Hierarchical Predicate Encryption for Inner-Products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
25. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
26. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. Cryptology ePrint Archive, Report 2010/563 (2010), http://eprint.iacr.org/
27. Okamoto, T., Takashima, K.: Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 138–159. Springer, Heidelberg (2011)
28. Okamoto, T., Takashima, K.: Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012)
29. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
30. Seo, J.H., Cheon, J.H.: Fully secure anonymous hierarchical identity-based encryption with constant size ciphertexts. Cryptology ePrint Archive, Report 2011/021 (2011), http://eprint.iacr.org/
31. Seo, J.H., Kobayashi, T., Ohkubo, M., Suzuki, K.: Anonymous Hierarchical Identity-Based Encryption with Constant Size Ciphertexts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 215–234. Springer, Heidelberg (2009)

32. 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)
33. Shen, E., Shi, E., Waters, B.: Predicate Privacy in Encryption Systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)
34. Shi, E., Waters, B.: Delegating Capabilities in Predicate Encryption Systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)
35. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
36. Yao, D., Fazio, N., Dodis, Y., Lysyanskaya, A.: Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In: ACM CCS 2004, pp. 354–363. ACM (2004)

Fully Secure Hidden Vector Encryption

Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano

Dipartimento di Informatica ed Applicazioni,
Università di Salerno, 84084 Fisciano (SA), Italy
{decaro,iovino,giuper}@dia.unisa.it

Abstract. Predicate encryption is an important cryptographic primitive (see [3,5,9,11]) that enables fine-grained control on the decryption keys. Roughly speaking, in a predicate encryption scheme the owner of the master secret key Msk can derive secret key Sk_P, for any predicate P from a specified class of predicates $\mathbb{P}$. In encrypting a message M, the sender can specify an *attribute* vector $\boldsymbol{x}$ and the resulting ciphertext $\tilde{X}$ can be decrypted only by using keys Sk_P such that $P(\boldsymbol{x}) = 1$. Security is modeled by means of a game between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ that sees the public key, is allowed to ask for keys of predicates P of his choice and gives two *challenge vectors* $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. $\mathcal{A}$ then receives a *challenge ciphertext* (an encryption of a randomly chosen challenge vector) and has to guess which of the two challenge vectors has been encrypted. The adversary $\mathcal{A}$ is allowed to ask queries even after seeing the challenge ciphertext. In the *unrestricted* queries model, it is required the adversary $\mathcal{A}$ to ask for keys of predicates P that do not discriminate the two challenge vectors; that is, for which $P(\boldsymbol{x}_0) = P(\boldsymbol{x}_1)$. It can be readily seen that this condition is necessary. In this paper, we consider *hidden vector encryption* (HVE in short), a notable case of predicate encryption introduced by Boneh and Waters [5] and further developed in [16,10,15]. In a HVE scheme, the ciphertext attributes are vectors $\boldsymbol{x} = \langle x_1, \ldots, x_\ell \rangle$ of length ℓ over alphabet Σ, keys are associated with vectors $\boldsymbol{y} = \langle y_1, \ldots, y_\ell \rangle$ of length ℓ over alphabet $\Sigma \cup \{\star\}$ and we consider the $\mathsf{Match}(\boldsymbol{x}, \boldsymbol{y})$ predicate which is true if and only if, for all i, $y_i \neq \star$ implies $x_i = y_i$. In [5], it is shown that HVE implies predicate encryption schemes for conjunctions, comparison, range queries and subset queries. We describe also constructions of secure predicate encryption for Boolean predicates that can be expressed as k-CNF and k-DNF (for any constant k) over *binary variables*.

Our main contribution is a *very simple*, in terms of construction and security proof, implementation of the HVE primitive that can be proved *fully* secure against probabilistic polynomial-time adversaries in the *unrestricted* queries model under non-interactive constant sized (that is independent of ℓ) hardness assumptions on bilinear groups of composite order. Our proof employs the dual system methodology of Waters [18], that gave one of the first fully secure construction in this area, blended with a careful design of intermediate security games that keep into account the relationship between challenge ciphertext and key queries.

Keywords: predicate encryption, HVE, full security, pairing-based cryptography.

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 102–121, 2013.

1 Introduction and Related Work

Predicate encryption is an important cryptographic primitive (see [3,5,9,11]) that enables fine-grained control on the decryption keys. Roughly speaking, in a predicate encryption scheme for a class $\mathbb{P}$ of ℓ-ary predicates, the owner of the master secret key Msk can derive secret key Sk_P for any predicate $P \in \mathbb{P}$. In encrypting a message M, the sender can specify an *attribute* vector $\boldsymbol{x}$ of length ℓ and the resulting ciphertext X can be decrypted only by using keys Sk_P such that $P(\boldsymbol{x}) = 1$. Thus a predicate encryption scheme enables the owner of the master secret key to delegate the decryption of different types of ciphertexts to different entities by releasing the appropriate key. In the context of predicate encryption, security is modeled by means of a game between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ that sees the public key, is allowed to ask for keys of predicates P of his choice and gives two *challenge vectors* $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. $\mathcal{A}$ then receives a *challenge ciphertext* (an encryption of a randomly chosen challenge vector) and has to guess which of the two challenge vectors has been encrypted. The adversary $\mathcal{A}$ is allowed to ask queries even after seeing the challenge ciphertext. In the *unrestricted* queries model, it is required the adversary $\mathcal{A}$ to ask for keys of predicates P that do not discriminate the two challenge vectors; that is, for which $P(\boldsymbol{x}_0) = P(\boldsymbol{x}_1)$. It can be readily seen that this condition is necessary. Many previous works restricted the proof of security to adversaries that could ask only *non-satisfying* queries (*restricted* queries model); that is, ask for keys of predicates P such that $P(\boldsymbol{x}_0) = P(\boldsymbol{x}_1) = 0$.

We consider *hidden vector encryption* (HVE in short), a notable case of predicate encryption introduced by [5]. In a HVE scheme, the ciphertext attributes are vectors $\boldsymbol{x} = \langle x_1, \ldots, x_\ell \rangle$ of length ℓ over alphabet Σ and predicates are described by vectors $\boldsymbol{y} = \langle y_1, \ldots, y_\ell \rangle$ of length ℓ over alphabet $\Sigma \cup \{\star\}$. The class $\mathbb{P}$ of predicates for HVE consists of all predicates $\mathsf{Match}_{\boldsymbol{y}}$ defined as follows: $\mathsf{Match}_{\boldsymbol{y}}(\boldsymbol{x})$ is true if and only if, for all i, $y_i \neq \star$ implies $x_i = y_i$. In the rest of the paper we will adopt the writing $\mathsf{Match}(\boldsymbol{x}, \boldsymbol{y})$ instead of $\mathsf{Match}_{\boldsymbol{y}}(\boldsymbol{x})$. Besides being one of the first predicates for which constructions have been given, HVE can be used as building block for several other predicates. Specifically in [5], it is shown that HVE implies predicate encryption schemes for conjunctions, comparison, range queries and subset queries. For completeness, in Appendix 6, we describe also constructions of secure predicate encryption for Boolean predicates that can be expressed as k-CNF and k-DNF (for any constant k).

Our main contribution is a *very simple*, in terms of construction and security proof, implementation of the HVE primitive that can be proved *fully* secure against probabilistic polynomial-time adversaries in the *unrestricted* queries model under non-interactive constant sized (that is independent of ℓ) hardness assumptions on bilinear groups of composite order. Specifically, our two assumptions posit the difficulty of a subgroup decision problem and of a problem that can be seen as the generalization of Decision Diffie-Hellman to groups of composite order.

Related Work. The first implementation of HVE is due to [5] that proved the security of their construction under assumptions on bilinear groups of composite order in the selective model. In this security model (introduced by [6] in the context of IBE), the adversary must announce to its challenge vectors before seeing the public key of the HVE scheme. In a recent series of papers Waters [18] and Lewko and Waters [12] introduced the concept of a dual system encryption scheme that was used to construct efficient and fully secure Identity Based Encryption (IBE) and Hierarchical IBE from simple assumptions. Previous fully secure constructions of these primitives either used a partitioning strategy (see [2],[17]) or used complexity assumptions of non-constant size (see [7],[8]). Partitioning strategy and the approaches of [7] and [8] do not seem to be helpful in proving full security of more complex primitives like HVE. Fully secure constructions of HVE in the *unrestricted* queries model can be already derived, via the reduction given in [11], from the fully secure constructions for inner-product encryption given by [13]. Anyway, we stress that the main goal of this paper is to present a *very simple*, in terms of construction and security proof, and direct implementation of the HVE primitive that can be still proved *fully* secure in the *unrestricted* queries model.

Proof Technique. Our proof of security is based on the dual system encryption methodology introduced by Waters [18] and gives extra evidence of the power of this proof technique. However, to overcome the difficulty of having to deal also with the unrestricted queries model, we have to carefully look at the space of matching queries and at how they relate to the challenge vectors. This enables us to craft a new security game in which the challenge ciphertext is constructed in a way that guarantees that keys obtained by the adversary give the expected result when tested against the challenge ciphertext and, at the same time, the challenge ciphertext is independent from the challenge vector used to construct it. Then we show, by means of a sequence of intermediate security games, that the real security game is computationally indistinguishable from this new game

2 Hidden Vector Encryption

In this section we give formal definitions for Hidden Vector Encryption (HVE) and its security properties. For sake of simplicity, we present predicate-only definitions and constructions for HVE instead of full-fledged ones. For the same reason, we give our definitions and constructions for binary alphabets.

Following standard terminology, we call a function $\nu(\lambda)$ *negligible* if for all constants $c > 0$ and sufficiently large λ, $\nu(\lambda) < 1/\lambda^c$ and denote by $[n]$ the set of integers $\{1, \ldots, n\}$. Moreover the writing "$a \leftarrow A$", for a finite set A, denotes that a is randomly and uniformly selected from A.

Hidden Vector Encryption. Let $\boldsymbol{x}$ be a binary vector of length ℓ and $\boldsymbol{y}$ a vector of the same length over $\{0, 1, \star\}$. We remind that predicate $\mathsf{Match}(\boldsymbol{x}, \boldsymbol{y})$ is defined to be true if and only if the two vectors agree in all positions i where

$y_j \neq \star$. A Hidden Vector Encryption scheme is a tuple of four efficient probabilistic algorithms (Setup, $\mathsf{Encrypt}$, KeyGen, Test) with the following semantics.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: takes as input a security parameter λ and a length parameter ℓ (given in unary), and outputs public parameters Pk and master secret key Msk.

$\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$: takes as input the master secret key Msk and a vector $\boldsymbol{y} \in \{0,1,\star\}^\ell$, and outputs a secret key $\mathsf{Sk}_{\boldsymbol{y}}$.

$\mathsf{Encrypt}(\mathsf{Pk}, \boldsymbol{x})$: takes as input the public parameters Pk and a vector $\boldsymbol{x} \in \{0,1\}^\ell$ and outputs a ciphertext Ct.

$\mathsf{Test}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk}_{\boldsymbol{y}})$: takes as input the public parameters Pk, a ciphertext Ct encrypting $\boldsymbol{x}$ and a secret key $\mathsf{Sk}_{\boldsymbol{y}}$ and outputs $\mathsf{Match}(\boldsymbol{x}, \boldsymbol{y})$.

For correctness we require that, for pairs $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, it holds that for all vectors $\boldsymbol{x} \in \{0,1\}^\ell$ and $\boldsymbol{y} \in \{0,1,\star\}^\ell$, we have that $\mathsf{Test}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, \boldsymbol{x}), \mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})) = \mathsf{Match}(\boldsymbol{x}, \boldsymbol{y})$ with very hight probability.

Security Definitions for HVE. In this section we formalize our security requirement by means of a security game GReal between a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. GReal consists of a Setup phase and of a Query Answering phase. In the Query Answering phase, the adversary can issue a polynomial number of Key Queries and one Challenge Construction query and at the end of this phase $\mathcal{A}$ outputs a guess. We stress that key queries can be issued by $\mathcal{A}$ even after he has received the challenge from $\mathcal{C}$. In GReal the adversary is restricted to queries for vectors $\boldsymbol{y}$ such that $\mathsf{Match}(\boldsymbol{y}, x_0) = \mathsf{Match}(\boldsymbol{y}, x_1)$.

More precisely, we define game GReal in the following way.

Setup. $\mathcal{C}$ runs the Setup algorithm on input the security parameter λ and the length parameter ℓ (given in unary) to generate public parameters Pk and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering$(\boldsymbol{y})$. $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$.

Challenge Query Answering$(\boldsymbol{x}_0, \boldsymbol{x}_1)$. $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns the challenge ciphertext computed by executing $\mathsf{Encrypt}(\mathsf{Pk}, \boldsymbol{x}_\eta)$.

Winning Condition. Let η' be $\mathcal{A}$'s output. $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and for all $\boldsymbol{y}$ for which $\mathcal{A}$ has issued a Key Query, it holds $\mathsf{Match}(\boldsymbol{x}_0, \boldsymbol{y}) = \mathsf{Match}(\boldsymbol{x}_1, \boldsymbol{y})$.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{HVE}}(\lambda)$ of $\mathcal{A}$ in GReal to be the probability of winning minus $1/2$.

Definition 1. *An Hidden Vector Encryption scheme is* secure *if for all probabilistic polynomial time adversaries $\mathcal{A}$, we have that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{HVE}}(\lambda)$ is a negligible function of λ.*

Larger Alphabet. One can easily observe that an HVE scheme for a general alphabet Σ can be obtained with an expansion of $\log_2 |\Sigma|$: the encryption simply encrypts bit by bit by using the binary HVE, and the key generation procedure proceeds analogously. We stress that this reduction is *black-box* and does not depend on our specific scheme.

3 Complexity Assumptions

We work with *symmetric* bilinear groups of composite order. Our construction can be adapted to the asymmetric setting in a straightforward way. Composite order bilinear groups were first used in Cryptography by [4] (see also [1]). We suppose the existence of an efficient group generator algorithm $\mathcal{G}$ which takes as input the security parameter λ and outputs a description $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ of a bilinear setting, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order N, and $\mathbf{e} : \mathbb{G}^2 \to \mathbb{G}_T$ is a map with the following properties:

1. (Bilinearity) $\forall\ g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_N$ it holds that $\mathbf{e}(g^a, h^b) = \mathbf{e}(g, h)^{ab}$.
2. (Non-degeneracy) $\exists\ g \in \mathbb{G}$ such that $\mathbf{e}(g, g)$ has order N in $\mathbb{G}_T$.

We assume that the group descriptions of $\mathbb{G}$ and $\mathbb{G}_T$ include generators of the respective cyclic subgroups. We require that the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map $\mathbf{e}$ are computable in deterministic polynomial time in λ. In our construction we will make hardness assumptions for bilinear settings whose order N is product of four distinct primes each of length $\Theta(\lambda)$. For an integer m dividing N, we let $\mathbb{G}_m$ denote the subgroup of $\mathbb{G}$ of order m. From the fact that the group is cyclic, it is easy to verify that if g and h are group elements of co-prime orders then $\mathbf{e}(g, h) = 1$. This is called the *orthogonality* property and is a crucial tool in our constructions. We are now ready to give our complexity assumptions.

Assumption 1. The first assumption is a subgroup-decision type assumption for bilinear settings. More formally, we have the following definition. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_3 \leftarrow \mathbb{G}_{p_3},\ A_{13} \leftarrow \mathbb{G}_{p_1p_3},\ A_{12} \leftarrow \mathbb{G}_{p_1p_2},\ A_4 \leftarrow\in \mathbb{G}_{p_4},\ T_1 \leftarrow \mathbb{G}_{p_1p_3},\ T_2 \leftarrow \mathbb{G}_{p_2p_3}$, and set $D = (\mathcal{I}, A_3, A_4, A_{13}, A_{12})$. We define the advantage of any $\mathcal{A}$ in breaking Assumption 1 to be $\mathsf{Adv}_1^{\mathcal{A}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D, T_1) = 1] - \mathrm{Prob}[\mathcal{A}(D, T_2) = 1]|$

Assumption 1. *We say that Assumption* 1 *holds for generator* $\mathcal{G}$ *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$, $\mathsf{Adv}_1^{\mathcal{A}}(\lambda)$ *is a negligible function of* λ.

Assumption 2. Our second assumption can be seen as the Decision Diffie-Hellman Assumption for composite order groups. More formally, we have the following definition. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_1 \leftarrow \mathbb{G}_{p_1}, A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4, C_4, D_4 \leftarrow \mathbb{G}_{p_4}, \alpha, \beta \leftarrow \mathbb{Z}_{p_1}, T_2 \leftarrow \mathbb{G}_{p_1p_4}$, and set $T_1 = A_1^{\alpha\beta} \cdot D_4$ and $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4)$. We define the advantage of any $\mathcal{A}$ in breaking Assumption 2 to be $\mathsf{Adv}_2^{\mathcal{A}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D, T_1) = 1] - \mathrm{Prob}[\mathcal{A}(D, T_2) = 1]|$

Assumption 2. *We say that Assumption* 2 *holds for generator* $\mathcal{G}$ *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$, $\mathsf{Adv}_2^{\mathcal{A}}(\lambda)$ *is a negligible function of* λ.

4 Constructing HVE

In this section we describe our HVE scheme. To make our description and proofs simpler, we add to all vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ two dummy components and set both of them equal to 0. We can thus assume that all vectors have at least two non-star positions.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization, and random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$, and, for $i \in [\ell]$ and $b \in \{0,1\}$, random $t_{i,b} \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$. The public parameters are $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and the master secret key is $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$.

$\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$: Let $S_{\boldsymbol{y}}$ be the set of indices i such that $y_i \neq \star$. The key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ for $i \in S_{\boldsymbol{y}}$ under the constraint that $\sum_{i\in S_{\boldsymbol{y}}} a_i = 0$. For $i \in S_{\boldsymbol{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ and sets $Y_i = g_{12}^{a_i/t_{i,y_i}} \cdot W_i$. The algorithm returns the tuple $(Y_i)_{i\in S_{\boldsymbol{y}}}$. Here we use the fact that $S_{\boldsymbol{y}}$ has size at least 2.

$\mathsf{Encrypt}(\mathsf{Pk}, \boldsymbol{x})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$. For $i \in [\ell]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets $X_i = T_{i,x_i}^s \cdot Z_i$, and returns the tuple $(X_i)_{i\in[\ell]}$.

$\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\boldsymbol{y}})$: The test algorithm computes $T = \prod_{i\in S_{\boldsymbol{y}}} \mathbf{e}(X_i, Y_i)$. It returns TRUE if $T = 1$, FALSE otherwise.

Correctness. It easy to see that the scheme is correct.

Remark 1. In our construction the Match predicate is computed in the subgroup of $\mathbb{G}$ of order p_1, $\mathbb{G}_{p_1}$. Notice that the other subgroups do not interfere during the evaluation of the predicate due to the orthogonality property. The subgroup of $\mathbb{G}$ of order p_2, $\mathbb{G}_{p_2}$, represents the semi-functional space and it is used to prove the security of the scheme. Notice that at this stage it can be removed from our construction at the expense of introducing another game in the security proof. To simplify the proof we have decide to include that subgroup directly. Finally, the subgroups $\mathbb{G}_{p_3}$ and $\mathbb{G}_{p_4}$ are used to re-randomize the public key and the ciphertexts, and the secret keys respectively. Their main role is to create enough room to manipulate the semi-functional space.

Remark 2. Let $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and $\mathsf{Msk} = [g_1g_2, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ be a pair of public parameter and master secret key output by the Setup algorithm and consider $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and $\mathsf{Msk}' = [\hat{g}_1 \cdot g_2, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ with $T'_{i,b} = \hat{g}_1^{t_{i,b}} \cdot R'_{i,b}$ for some $\hat{g}_1 \in \mathbb{G}_{p_1}$ and $R'_{i,b} \in \mathbb{G}_{p_3}$. We make the following easy observations: (1) For every $\boldsymbol{y} \in \{0,1,\star\}^\ell$, $\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$ and $\mathsf{KeyGen}(\mathsf{Msk}', \boldsymbol{y})$ are identical distributed. (2) Similarly, for every $\boldsymbol{x} \in \{0,1\}^\ell$, $\mathsf{Encrypt}(\mathsf{Pk}, \boldsymbol{x})$ and $\mathsf{Encrypt}(\mathsf{Pk}', \boldsymbol{x})$ are identical distributed.

5 Security of Our HVE Scheme

We start by giving an informal description of the ideas behind our proof of security and show how we overcome the main technical difficulty of having to deal with adversaries that possess keys that match the challenge ciphertext.
The first step of our proof strategy consists in projecting the public key (and thus the ciphertexts the adversary constructs by himself) into the semi functional space and thus to a different subgroup from the one of the challenge ciphertext. Specifically, we defined a new security game GPK in which the $t_{i,b}$'s are encoded in the $\mathbb{G}_{p_2}$ part of the $T_{i,b}$'s from the public key (instead of the the $\mathbb{G}_{p_1}$ part as in normal public key in the real game). The challenge ciphertext and the answers to the key queries are instead constructed as in the real security game GReal. Thus, ciphertexts constructed by the adversary are completely independent from the challenge ciphertext (as they encode information in two different subgroups). We observe that since keys are constructed as in the real security game, they carry information about $\boldsymbol{y}$ both in the $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_2}$ parts. Thus when the adversary tests a ciphertext he has constructed by using the public key against a key obtained by means of a query, he obtains the expected result because of the information encoded in the $\mathbb{G}_{p_2}$ part of the key and of the ciphertext. The challenge ciphertext instead interacts with the $\mathbb{G}_{p_1}$ part of the keys. The only difference between the two games is in the public key but, under Assumption 1 (a natural subgroup decision hardness assumption), we can prove that the two games are indistinguishable. Here a crucial role is played by $\mathbb{G}_{p_3}$ as it enables the challenger of GPK to move the public key to the semi-functional space and to create the challenge ciphertext with the respect to an independent public key. Without $\mathbb{G}_{p_3}$, a normal public key (as seen by the adversary in GReal) would be in $\mathbb{G}_{p_1}$ whereas a semi-functional public key (as seen by the adversary in GPK) would be in $\mathbb{G}_{p_2}$. Moreover, in both games, the adversary will be given a challenge ciphertext in $\mathbb{G}_{p_1}$ and thus, by orthogonality, it would be able to distinguish the two games. **The second step** proves that the keys obtained from queries do not help the adversary. Since the challenge ciphertext carries information about the randomly selected challenge vector $\boldsymbol{x}_\eta$ in its $\mathbb{G}_{p_1}$ part, in this informal discussion when we refer to key we mean its $\mathbb{G}_{p_1}$ part. The $\mathbb{G}_{p_2}$ parts of the keys are always correctly computed. In our construction, testing a ciphertext against a non-matching key gives a random value (from the target group) whereas testing it against a matching key returns a specified value (the identity of the target group). If we had to prove security against an adversary that asked only non-matching queries we could consider the experiment in which key queries were replied by returning a key with random $\mathbb{G}_{p_1}$ parts. Such a game can be proved indistinguishable from GPK (under an appropriate complexity assumption) and it is easy to prove that it gives no advantage to an adversary. This approach fails for matching queries as such a key will return the wrong answer with high probability when tested against the challenge ciphertext. Instead we modify the construction of the challenge ciphertext in the following way: the challenge ciphertext is well-formed in all the positions where the two challenge vectors are equal and random in all the other positions. We observe that

testing such a challenge ciphertext against matching and non-matching keys always gives the correct answer and that no adversary (even an all powerful one) can guess which of the two challenge vectors has been used to construct the challenge ciphertext (see the discussion in Section 5.2).

5.1 The First Step of the Proof

We start by defining game $\mathsf{GPK}(\lambda, \ell)$ (see Figure 1) that differs from $\mathsf{GReal}(\lambda, \ell)$ as in the Setup phase, $\mathcal{C}$ prepares two sets of public parameters, Pk and Pk', and one master secret key Msk. Pk is given as input to $\mathcal{A}$, Msk is used to answer $\mathcal{A}$'s key queries and Pk' is used to construct the challenge ciphertext. The next

Game $\mathsf{GPK}(\lambda, \ell)$

Setup. $\mathcal{C}$ chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization and random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3 \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$ and sets $g_{12} = g_1 \cdot g_2$. For each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in Z_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T'_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$ and $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{C}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. Finally, $\mathcal{C}$ sends Pk to $\mathcal{A}$.

Key Query Answering$(\boldsymbol{y})$. $\mathcal{C}$ returns the output of $\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$.

Challenge Query Answering$(\boldsymbol{x}_0, \boldsymbol{x}_1)$. Upon receiving the pair $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ of challenge vectors, $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns the output of $\mathsf{Encrypt}(\mathsf{Pk}', \boldsymbol{x}_\eta)$.

Winning Condition. Like in $\mathsf{GReal}(\lambda, \ell)$.

Fig. 1. A formal description of GPK

lemma shows that, the advantages of an adversary in $\mathsf{GReal}(\lambda, \ell)$ and $\mathsf{GPK}(\lambda, \ell)$ are the same, up to a negligible factor.

Lemma 1. *If Assumption* 1 *holds,* $\mathsf{GReal}(\lambda, \ell) \approx_c \mathsf{GPK}(\lambda, \ell)$

Proof. We show a PPT algorithm $\mathcal{B}$ which receives $(\mathcal{I}, A_3, A_4, A_{13}, A_{12})$ and T and, depending on the nature of T, simulates $\mathsf{GReal}(\lambda, \ell)$ or $\mathsf{GPK}(\lambda, \ell)$ with $\mathcal{A}$. This suffices to prove the Lemma.

Setup. $\mathcal{B}$ starts by constructing public parameters Pk and Pk' in the following way. $\mathcal{B}$ sets $g_{12} = A_{12}, g_3 = A_3, g_4 = A_4$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and sets $T_{i,b} = T^{t_{i,b}}$ and $T'_{i,b} = A_{13}^{t_{i,b}}$. Then $\mathcal{B}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and starts the interaction with $\mathcal{A}$ on input Pk.

Answering Key Query for $(\boldsymbol{y})$**.** $\mathcal{B}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$.

Answering Challenge Query for $(\boldsymbol{x}_0, \boldsymbol{x}_1)$**.** The challenge is created by $\mathcal{B}$ by picking random $\eta \in \{0,1\}$ and running $\mathsf{Encrypt}(\mathsf{Pk}', \boldsymbol{x}_\eta)$.

This concludes the description of algorithm $\mathcal{B}$. Now suppose $T \in \mathbb{G}_{p_1p_3}$, and thus it can be written as $T = h_1 \cdot h_3$ for $h_1 \in \mathbb{G}_{p_1}$ and $h_3 \in \mathbb{G}_{p_3}$. This implies that Pk received in input by $\mathcal{A}$ in the interaction with $\mathcal{B}$ has the same distribution as

Game $\mathsf{GBadQ}(f,k)$

Setup. Like in GPK. That is, $\mathcal{C}$ chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization and random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3 \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$ and sets $g_{12} = g_1 \cdot g_2$. For each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in Z_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T'_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$ and $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{C}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. Finally, $\mathcal{C}$ sends Pk to $\mathcal{A}$.

Answering Key-Query for $\boldsymbol{y} = \langle y_1, \ldots, y_\ell\rangle$. $\mathcal{C}$ answers the first k queries in the following way.

- If $y_f \neq \star$, $\mathcal{C}$ returns a key whose $\mathbb{G}_{p_1}$ parts is random. More specifically, $\mathcal{C}$ chooses, for each $i \in S_{\boldsymbol{y}}$, random $W_i \in \mathbb{G}_{p_4}$, random $C_i \in \mathbb{G}_{p_1}$ and random $a_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i\in S_y} a_i = 0$ and sets $Y_i = C_i \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i$.
- If $y_f = \star$ then $\mathcal{C}$ returns the output of $\mathsf{KeyGen}(\boldsymbol{y}, \mathsf{Msk})$.

The remaining $q-k$ queries are answered by running $\mathsf{KeyGen}(\boldsymbol{y}, \mathsf{Msk})$.

Answering Challenge Query for $(\boldsymbol{x}_0, \boldsymbol{x}_1)$. $\mathcal{C}$ chooses random $s \in \mathbb{Z}_N$ and $\eta \in \{0,1\}$ and sets $\boldsymbol{x} = \boldsymbol{x}_\eta$. For each $i \in [f-1]$ such that $\boldsymbol{x}_{0,i} \neq \boldsymbol{x}_{1,i}$, $\mathcal{C}$ chooses random $X_i \in \mathbb{G}_{p_1p_3}$. Then, for each remaining i, $\mathcal{C}$ chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets $X_i = T'^{s}_{i,x_i} \cdot Z_i$. $\mathcal{C}$ returns the tuple $(X_i)_{i\in[\ell]}$.

Winning Condition. Like in GReal.

Fig. 2. A formal description of game $\mathsf{GBadQ}(f,k)$

in GReal. Moreover, by writing A_{13} as $A_{13} = \hat{h}_1 \cdot \hat{h}_3$ for $\hat{h}_1 \in \mathbb{G}_{p_1}$ and $\hat{h}_3 \in \mathbb{G}_{p_3}$ which is possible since by assumption $A_{13} \in \mathbb{G}_{p_1p_3}$, we notice that that Pk and Pk' are as in the hypothesis of Remark 2 (with $g_1 = h_1$ and $\hat{g}_1 = \hat{h}_1$). Therefore the answers to key queries and the challenge ciphertext given by $\mathcal{B}$ to $\mathcal{A}$ have the same distribution as the answers and the challenge ciphertext received by $\mathcal{A}$ in $\mathsf{GReal}(\lambda,\ell)$. We can thus conclude that, when $T \in \mathbb{G}_{p_1p_3}$, $\mathcal{C}$ has simulated $\mathsf{GReal}(\lambda,\ell)$ with $\mathcal{A}$. Let us discuss now the case $T \in \mathbb{G}_{p_2p_3}$. In this case, Pk provided by $\mathcal{B}$ has the same distribution as the public parameters produced by $\mathcal{C}$ in $\mathsf{GPK}(\lambda,\ell)$. Therefore, $\mathcal{C}$ is simulating $\mathsf{GPK}(\lambda,\ell)$ for $\mathcal{A}$. □

5.2 The Second Step of the Proof

We start the second step of the proof by describing in Figure 2, for $1 \leq f \leq \ell+1$ and $0 \leq k \leq q$, game $\mathsf{GBadQ}(f,k)$ between the challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ that asks q queries. Not to overburden our notation, we omitted λ and ℓ from the name of the games. $\mathsf{GBadQ}(f,k)$ differs from GPK both in the way in which key queries are answered and in the way in which the challenge ciphertext is constructed. Specifically, in $\mathsf{GBadQ}(f,k)$ the first k key queries are answered by distinguishing two cases. Queries for $\boldsymbol{y}$ such that $y_f = \star$ are answered by running $\mathsf{KeyGen}(\mathsf{Msk}, \boldsymbol{y})$. Instead queries for $\boldsymbol{y}$ such that $y_f \neq \star$ are answered by returning keys whose $\mathbb{G}_{p_1}$ part is random for all components. Moreover, in $\mathsf{GBadQ}(f,k)$, the $\mathbb{G}_{p_1}$ part of the first $f-1$ components of the challenge ciphertext corresponding to positions in which the two challenges differ are random.

In the proofs, we will use the shorthand $\mathsf{GBadCh}(f)$ to denote the game $\mathsf{GBadQ}(f,0)$ in which only the challenge ciphertext is modified whereas all the replies to the key queries are correctly computed. We define $\mathsf{GBadQ2}(f,k)$, for $1 \leq f \leq \ell$ and $0 \leq k \leq q$, as a game in which the setup phase is like in $\mathsf{GBadQ}(f,k)$, key queries are answered like in $\mathsf{GBadQ}(f,k)$ and the challenge ciphertext is constructed like in $\mathsf{GBadQ}(f+1,k)$.

Observation 1. $\mathsf{GPK} = \mathsf{GBadQ}(1,0) = \mathsf{GBadCh}(1)$. By definitions of the games.

Observation 2. $\mathsf{GBadQ}(f,q) = \mathsf{GBadQ2}(f,q)$ *for* $f = 1,\ldots,\ell$. From the definitions of the two games, it is clear that all key queries are answered in the same way in both the games and all components X_i for $i \neq f$ of the challenge ciphertext are computed in the same way. Let us now look at X_f and more precisely to its $\mathbb{G}_{p_1}$ part. In $\mathsf{GBadQ}(f,q)$, the $\mathbb{G}_{p_1}$ part of X_f is computed as T'^{s}_{f,x_f} which is exactly how it is computed in $\mathsf{GBadQ2}(f,q)$ when $x_{0,f} = x_{1,f}$. On the other hand, when $x_{0,f} \neq x_{1,f}$, the $\mathbb{G}_{p_1}$ part of X_f is chosen at random. However, observe that exponents $t_{f,0} \mod p_1$ and $t_{f,1} \mod p_1$ have not appeared in the answers to key queries since every query has either a $\star$ in position f (in which case position f of the answer is empty) or a non-$\star$ value in position f (in which case the $\mathbb{G}_{p_1}$ part of the position f of the answer is random since $k = q$). Therefore, we can conclude that the $\mathbb{G}_{p_1}$ part of the component X_f of the answer to the challenge query is also random in $\mathbb{G}_{p_1}$.

Observation 3. $\mathsf{GBadQ2}(f,0) = \mathsf{GBadQ}(f+1,0)$ *for* $f = 1,\ldots,\ell-1$. Indeed, in both games all key queries are answered correctly, and the challenge query in $\mathsf{GBadQ2}(f,0)$ is by definition answered in the same way as in $\mathsf{GBadQ}(f+1,0)$.

Observation 4. *For* $f = 1,\ldots,\ell-1$, *if* $x_{0,f} = x_{1,f}$, $\mathsf{GBadCh}(f) = \mathsf{GBadCh}(f+1)$. By definition, in $\mathsf{GBadCh}(f) = \mathsf{GBadQ}(f,0)$ the f-th component of the challenge ciphertext is well formed, namely $X_f = T'^{s}_{f,x_f} \cdot Z_f$. This is the same in $\mathsf{GBadCh}(f+1) = \mathsf{GBadQ}(f+1,0)$ under the condition that $x_{0,f} = x_{1,f}$.

Observation 5. *In* $\mathsf{GBadCh}(\ell+1) = \mathsf{GBadQ}(\ell+1,0)$ *all adversaries have no advantage.* This follows from the fact that, for positions i such that $x_{0,i} \neq x_{1,i}$, the $\mathbb{G}_{p_1}$ part of X_i is random. Thus the challenge ciphertext of $\mathsf{GBadCh}(\ell+1)$ is independent from η.

Overview of the proof second step. Consider the sequence $\mathsf{GPK} = \mathsf{GBadCh}(1)$, $\mathsf{GBadCh}(2),\ldots,\mathsf{GBadCh}(\ell),\mathsf{GBadCh}(\ell+1)$ of $\ell+1$ experiments. By Observation 5, if an adversary $\mathcal{A}$ has a non negligible advantage in GPK then it must be the case that there exists $1 \leq f \leq \ell$ such that the difference between $\mathcal{A}$'s advantages in $\mathsf{GBadCh}(f)$ and $\mathsf{GBadCh}(f+1)$ is non-negligible. Moreover, by Observation 4, for this to happen it must be the case that $\mathcal{A}$ has non-negligible probability to output two challenges that differ in the f-th component. Then, if $\mathcal{A}$ makes q key queries, consider the following sequence $\mathsf{GBadCh}(f) = \mathsf{GBadQ}(f,0)\ldots\mathsf{GBadQ}(f,q-1)$ $\mathsf{GBadQ}(f,q) = \mathsf{GBadQ2}(f,q)$ $\mathsf{GBadQ2}(f,q-1)\ldots\mathsf{GBadQ2}(f,0) = \mathsf{GBadCh}(f+1)$ of $2q+1$ games. If the difference in advantage between $\mathsf{GBadCh}(f)$ and

$\mathsf{GBadCh}(f+1)$ is non-negligible, then there must exist k such that the difference in advantage between either $\mathsf{GBadQ}(f,k)$ and $\mathsf{GBadQ}(f,k-1)$ or between $\mathsf{GBadQ2}(f,k)$ and $\mathsf{GBadQ2}(f,k-1)$ is non-negligible.

Now it seems that we are stuck as, for all f and k, games $\mathsf{GBadQ}(f,k-1)$ and $\mathsf{GBadQ}(f,k)$ can be distinguished by an adversary $\mathcal{A}$ using the following simple strategy. $\mathcal{A}$ prepares two challenges $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ that coincide in the f-th component and asks as k-th key query the key Y for a vector $\boldsymbol{y}^{(k)}$ such that $\boldsymbol{y}_f^{(k)} = x_{0,f} = x_{1,f}$ and $\mathsf{Match}(\boldsymbol{x}_0, \boldsymbol{y}^{(k)}) = \mathsf{Match}(\boldsymbol{x}_1, \boldsymbol{y}^{(k)}) = 1$. Let X be the challenge ciphertext. Now, in $\mathsf{GBadQ}(f,k-1)$, the answer Y received by $\mathcal{A}$ to its k-th query is well-formed and thus $\mathsf{Test}(X,Y)=1$. Instead in $\mathsf{GBadQ}(f,k)$, Y is random and thus $\mathsf{Test}(X,Y)=0$ except with negligible probability. The above strategy requires the two challenges to coincide in the f-th component. Indeed, if $x_{0,f} \neq x_{1,f}$ then for $\boldsymbol{y}^{(k)}$ to be matching it must the case that $y_f = \star$ but then in this case Y is well-formed in both games. This perfectly fits our strategy as we have to prove that $\mathsf{GBadQ}(f,k)$ is indistinguishable from $\mathsf{GBadQ}(f,k-1)$ only for f for which $\mathcal{A}$ has a non-negligible probability of outputting two challenges that differ in the f-th component. Exactly, the same reasoning holds for $\mathsf{GBadQ2}$.

In the next section we describe a simulator $\mathcal{S}$ that takes as input the pair of integers f and k and an instance of Assumption 2 and, provided that the adversary does not output two challenges that coincide in the f-th component, simulates with some non-negligible probability $\mathsf{GBadQ}(f,k)$ or $\mathsf{GBadQ}(f,k-1)$ depending on the nature of the challenge. A similar simulator can be constructed for games $\mathsf{GBadQ2}(f,k)$ and $\mathsf{GBadQ2}(f,k-1)$.

Description of simulator $\mathcal{S}$ Input to $\mathcal{S}$. Integers $1 \leq f \leq \ell+1$ and $0 \leq k \leq q$, and a randomly chosen instance (D,T) of Assumption 2; recall that $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha} B_4, A_1^{\beta} C_4)$ and $T = A_1^{\alpha\beta} D_4$ or random $\mathbb{G}_{p_1 p_4}$.

Setup. To simulate the Setup phase $\mathcal{S}$ executes the following steps.

1. $\mathcal{S}$ sets $g_1 = A_1$, $g_2 = A_2$, $g_3 = A_3$, $g_4 = A_4$ and $g_{12} = A_1 \cdot A_2$.
2. For each $i \in [\ell]$ and $b \in \{0,1\}$,
   $\mathcal{S}$ chooses random $v_{i,b} \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$, and sets $T_{i,b} = g_2^{v_{i,b}} \cdot R_{i,b}$.
3. $\mathcal{S}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$.
4. $\mathcal{S}$ picks random $\hat{\jmath} \in [\ell]$ and $\hat{b} \in \{0,1\}$ and sets $\hat{c} = 1 - \hat{b}$.
5. For each $i \in [\ell] \setminus \{\hat{\jmath}\}$ and $b \in \{0,1\}$, $\mathcal{S}$ chooses random $r_{i,b} \in \mathbb{Z}_N$ and $R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T'_{i,b} = g_1^{r_{i,b}} \cdot R'_{i,b}$.
6. $\mathcal{S}$ chooses random $r_{\hat{\jmath},\hat{c}} \in \mathbb{Z}_N$ and $R'_{\hat{\jmath},\hat{c}} \in \mathbb{G}_{p_3}$.
   $\mathcal{S}$ sets $T'_{\hat{\jmath},\hat{c}} = g_1^{r_{\hat{\jmath},\hat{c}}} \cdot R'_{\hat{\jmath},\hat{c}}$, $T'_{\hat{\jmath},\hat{b}} = \perp$ and $r_{\hat{\jmath},\hat{b}} = \perp$.
7. $\mathcal{S}$ sets $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and $\mathsf{Msk} = [g_{12}, g_4, (r_{i,b}, v_{i,b})_{i\in[\ell],b\in\{0,1\}}]$.

Notice that the values $r_{\hat{\jmath},\hat{b}}$ and $T'_{\hat{\jmath},\hat{b}}$ are unspecified and thus Pk' and Msk are incomplete. As we shall see below, in answering key queries, $\mathcal{S}$ will implicitly set $r_{\hat{\jmath},\hat{b}} = 1/\beta$. Here β is the exponent of A_1 in $A_1^{\beta} \cdot C_4$ from instance D of Assumption 2 and we stress that $\mathcal{S}$ does not have access to the actual value of β. $\mathcal{S}$ starts the interaction with $\mathcal{A}$ on input Pk.

Answering Key Query for $\boldsymbol{y} = \langle y_1, \ldots, y_\ell \rangle$.

- **First $k-1$ key queries**. We have the following mutually exclusive cases.
  Case A.1: $y_f \neq \star$**.** In this case, $\mathcal{S}$ outputs a key whose $\mathbb{G}_{p_1}$ part is random. More precisely, $\mathcal{S}$ executes the following steps. For each $i \in S_{\boldsymbol{y}}$, $\mathcal{S}$ chooses random a_i'' such that $\sum_{i \in S_{\boldsymbol{y}}} a_i'' = 0$, random $C_i \in \mathbb{G}_{p_1}$, and random $W_i \in \mathbb{G}_{p_4}$. Then, for each $i \in S_{\boldsymbol{y}}$, $\mathcal{S}$ sets $Y_i = C_i \cdot g_2^{a_i''/v_{i,y_i}} \cdot W_i$.
  Case A.2: $y_f = \star$. In this case, $\mathcal{S}$ outputs a key that has the same distribution induced by algorithm KeyGen on input $\boldsymbol{y}$ and Msk. We observe that if $y_{\hat{\jmath}} = \hat{c}$ then Msk includes all the r_{i,y_i}'s and v_{i,y_i}'s that are needed. If instead $y_{\hat{\jmath}} = \hat{b}$, then Msk is missing $r_{\hat{\jmath},\hat{b}}$. In this case $\mathcal{S}$ computes $Y_{\hat{\jmath}}$ by using $A_1^{\beta} \cdot C_4$ from the challenge D of Assumption 2 received in input.
  More precisely, for each $i \in S_{\boldsymbol{y}}$, $\mathcal{S}$ picks random $W_i \in \mathbb{G}_{p_4}$ and random $a_i', a_i'' \in \mathbb{Z}_N$ under the constraint that $\sum_{i \in S_{\boldsymbol{y}}} a_i' = \sum_{i \in S_{\boldsymbol{y}}} a_i'' = 0$. Then for each $i \neq \hat{\jmath}$, $\mathcal{S}$ sets $Y_i = g_1^{a_i'/r_{i,y_i}} \cdot g_2^{a_i''/v_{i,y_i}} \cdot W_i$. Moreover, if $y_{\hat{\jmath}} = \hat{c}$, $\mathcal{S}$ sets $Y_{\hat{\jmath}} = g_1^{a_{\hat{\jmath}}'/r_{\hat{\jmath},\hat{c}}} \cdot g_2^{a_{\hat{\jmath}}''/v_{\hat{\jmath},\hat{c}}} \cdot W_{\hat{\jmath}}$ otherwise, if $y_{\hat{\jmath}} = \hat{b}$, $\mathcal{S}$ sets $Y_{\hat{\jmath}} = (A_1^{\beta} \cdot C_4)^{a_{\hat{\jmath}}'} \cdot g_2^{a_{\hat{\jmath}}''/v_{\hat{\jmath},\hat{b}}} \cdot W_{\hat{\jmath}} = g_1^{a_{\hat{\jmath}}'\beta} \cdot g_2^{a_{\hat{\jmath}}''/v_{\hat{\jmath},\hat{b}}} \cdot (C_4^{a_{\hat{\jmath}}'} \cdot W_{\hat{\jmath}})$. Notice that this setting implicitly defines $r_{\hat{\jmath},\hat{b}} = 1/\beta$ which remains unknown to $\mathcal{S}$.
- **k-th query.** Let $\boldsymbol{y}^{(k)} = (y_1^{(k)}, \ldots, y_\ell^{(k)})$ be the k-th key query.
  Case B.1: $y_f^{(k)} = \star$. $\mathcal{S}$ performs the same steps of Case A.2.
  Case B.2: $y_f^{(k)} \neq \star$ **and** $y_{\hat{\jmath}}^{(k)} \neq \hat{b}$. In this case, $\mathcal{S}$ aborts.
  Case B.3: $y_f^{(k)} \neq \star$ **and** $y_{\hat{\jmath}}^{(k)} = \hat{b}$. Let $S = S_{\boldsymbol{y}} \backslash \{\hat{\jmath}, h\}$, where h is an index such that $y_h^{(k)} \neq \star$. Such an index h always exists since we assumed that each query contains at least two non-$\star$ entries. Then, for each $i \in S$, $\mathcal{S}$ chooses random $W_i \in \mathbb{G}_{p_4}$ and random $a_i', a_i'' \in \mathbb{Z}_N$ and sets $Y_i = g_1^{a_i'/r_{i,y_i^{(k)}}} \cdot g_2^{a_i''/v_{i,y_i^{(k)}}} \cdot W_i$. $\mathcal{S}$ then chooses random $a_{\hat{\jmath}}'' \in \mathbb{Z}_N$ and $W_{\hat{\jmath}}, W_h \in \mathbb{G}_{p_4}$ and sets $Y_{\hat{\jmath}} = T \cdot g_2^{a_{\hat{\jmath}}''/v_{\hat{\jmath},\hat{b}}} \cdot W_{\hat{\jmath}}$ and $Y_h = (A_1^{\alpha} B_4)^{-1/r_{h,y_h^{(k)}}} \cdot g_1^{-s'/r_{h,y_h^{(k)}}} \cdot g_2^{-(s''+a_{\hat{\jmath}}'')/v_{h,y_h^{(k)}}} \cdot W_h$, where $s' = \sum_{i \in S} a_i'$ and $s'' = \sum_{i \in S} a_i''$.
  This terminates the description of how $\mathcal{S}$ handles the k-th key query.
- **Remaining $q-k$ queries.** $\mathcal{S}$ handles the remaining $q-k$ queries as in Case A.2, independently from whether $y_f = \star$ or $y_f \neq \star$.
  More precisely, if $y_{\hat{\jmath}} = \hat{c}$ then $\mathcal{S}$ runs KeyGen on input $\boldsymbol{y}$ and Msk and all the needed r_{i,y_i}'s and v_{i,y_i}'s are found in Msk. On the other hand, if $y_{\hat{\jmath}} = \hat{b}$, $\mathcal{S}$ can use $A_1^{\beta} \cdot C_4$ from D.

Answering Challenge Query for $(\boldsymbol{x}_0, \boldsymbol{x}_1)$. $\mathcal{S}$ picks random $\eta \in \{0,1\}$ and sets $\boldsymbol{x} = \boldsymbol{x}_\eta$. Then $\mathcal{S}$ tries to construct the challenge ciphertext by running algorithm $\mathsf{Encrypt}$ on input the challenge vector $\boldsymbol{x}$, public parameters Pk' and by randomizing the $\mathbb{G}_{p_1}$ part of all components X_i for $i < f$ such that $x_{0,i} \neq x_{1,i}$. However, Pk' is incomplete since it is missing $T'_{\hat{\jmath},\hat{b}}$ and thus $\mathcal{S}$ might have to abort. More precisely, If $x_{\hat{\jmath}} = \hat{b}$, $\mathcal{S}$ aborts. Else (that is, if $x_{\hat{\jmath}} = \hat{c}$) $\mathcal{S}$ chooses random $s \in \mathbb{Z}_N$. For each $i \in [f-1]$ such that $x_{0,i} \neq x_{1,i}$, $\mathcal{S}$ sets r_i equal to a random

element in $\mathbb{Z}_N$ and $r_i = 1$ for all remaining i's. Then, for each $i \in [\ell]$, $\mathcal{S}$ picks random $Z_i \in \mathbb{G}_{p_3}$ and sets $X_i = T'^{sr_i}_{i,x_i} \cdot Z_i$, and returns the tuple $(X_i)_{i\in[\ell]}$.

This ends the description of $\mathcal{S}$.

The simulator $\mathcal{S}$ described will be used to prove properties of games GBadQ. We can modify the simulator $\mathcal{S}$ so that, on input f and k, the challenge ciphertext is constructed by randomizing the $\mathbb{G}_{p_1}$ part also of the f-th component. The so modified simulator, that we call $\mathcal{S}_2$, closely simulates the work of games $\mathsf{GBadQ2}$ and will be used to prove properties of these games.

A Sanity Check. We verify that $\mathcal{S}$ cannot test the nature of T and thus break Assumption 2. Indeed to do so, $\mathcal{S}$ should use T to generate a key for $\boldsymbol{y}$ and ciphertext for $\boldsymbol{x}$ such that $\mathsf{Match}(\boldsymbol{x}, \boldsymbol{y}) = 1$. Then, if $T = T_1$ the Test procedure will have success; otherwise, it will fail. In constructing the key, $\mathcal{S}$ would use T to construct the $\hat{\jmath}$-th component (which forces $y_{\hat{\jmath}} = \hat{b}$) and then it would need $r_{\hat{\jmath},\hat{b}} = 1/\beta$ to construct the matching ciphertext. However, $\mathcal{S}$ does not have access to this value as part of the challenge. If we modify Assumption 2 to include such a value as part of the challenge then the resulting assumption doesn't hold.

Why We Need Aborts. The aim of $\mathcal{S}$ is to use the value T from the challenge to simulate either $\mathsf{GBadQ}(f, k-1)$ or $\mathsf{GBadQ}(f, k)$ and it does so by embedding T in the $\hat{\jmath}$-th component of the reply to k-th key query. Suppose $y_f \neq \star$ and thus the two games are supposed to differ in the reply to the k-th key query. If $y^{(k)}_{\hat{\jmath}} = \star$ then the $\hat{\jmath}$-th component is empty. Moreover, if $y^{(k)}_{\hat{\jmath}} = \hat{c}$ then the $\hat{\jmath}$-th component can be computed using $1/r_{\hat{\jmath},\hat{c}}$ which is known to $\mathcal{S}$ but independent from T. Thus in both case $\mathcal{S}$'s plan fails and consequently $\mathcal{S}$ aborts (see Case B.2). Moreover, if $x_{\hat{\jmath}} = \hat{b}$ then $\mathcal{S}$ should use $T'_{\hat{\jmath},\hat{b}}$ which is missing from Pk'. Thus in this case $\mathcal{S}$ aborts too.

Notation. We use $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ to denote the event that $\mathcal{S}$ does not abort while computing the answer to the k-th query in an interaction with $\mathcal{A}$ on input f and k. This is equivalent to the event that the k-th key query $y^{(k)}$ of adversary $\mathcal{A}$ is such that $y^{(k)}_f = \star$ or $y^{(k)}_{\hat{\jmath}} = \hat{b}$. In addition, we use $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{S}}(f,k)$ to denote the event that $\mathcal{S}$ does not abort while computing the challenge ciphertext in an interaction with $\mathcal{A}$ on input f and k. This is equivalent to the event that adversary $\mathcal{A}$ outputs challenge vectors $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ such that $x_{\eta,\hat{\jmath}} = \hat{c}$.

For a game G between the challenger $\mathcal{C}$ and the adversary, we modify $\mathcal{C}$ so that $\mathcal{C}$ picks $\hat{\jmath}$ and $\hat{b}$ just like $\mathcal{S}$ does. This modification makes the definitions of events $\mathsf{NotAbort}^{\mathcal{A}}_{1,G}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,G}$ meaningful. Notice however that, unlike the simulator $\mathcal{S}$, $\mathcal{C}$ never aborts its interaction with $\mathcal{A}$ and that this modification does not affect $\mathcal{A}$'s view. We write $\mathsf{NotAbort}^{\mathcal{A}}_2$ as a shorthand for $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{GPK}}$.

Lemma 2. *For all f, k and $\mathcal{A}$,* $\mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)] \geq \frac{1}{\ell}$.

Proof. The probability of $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ is at least the probability that $y^{(k)}_{\hat{\jmath}} = \hat{b}$. Moreover, the view of $\mathcal{A}$ up to the k-th key query is independent from $\hat{b}$ and $\hat{\jmath}$. Now observe that $\boldsymbol{y}^{(k)}$ has at least two non-star entry and, provided that

$\hat{\jmath}$ is one of these (which happens with probability at least $2/\ell$), the probability that $y_{\hat{\jmath}}^{(k)} = \hat{b}$ is $1/2$. □

Lemma 3. *For all f, k and $\mathcal{A}$,* $\text{Prob}[\mathsf{NotAbort}_{2,G}^{\mathcal{A}}(f,k)] \geq \frac{1}{2\ell}$ *for* $G = \mathsf{GBadQ}(f,k)$.

Proof. $\mathsf{NotAbort}_{2,G}^{\mathcal{A}}(f,k)$ is the event that $y_{\hat{\jmath}}^{(k)} \neq x_{\eta,\hat{\jmath}}$ in the game G played by the challenger $\mathcal{C}$ with $\mathcal{A}$. It is easy to see that the probability that $\mathcal{C}$ correctly guesses $\hat{\jmath}$ and $\hat{b}$ such that $x_{\eta,\hat{\jmath}} = \hat{c} = 1 - \hat{b}$ is at least $1/(2\ell)$, independently from the view of $\mathcal{A}$. □

Lemma 4. *Suppose event* $\mathsf{NotAbort}_{1,\mathcal{S}}^{\mathcal{A}}(f,k)$ *occurs. If $T = T_1$ then $\mathcal{A}$'s view up to the Challenge Query in the interaction with $\mathcal{S}$ running on input (f,k) is the same as in* $\mathsf{GBadQ}(f,k-1)$. *If instead $T = T_2$ then $\mathcal{A}$'s view up to the Challenge Query in the interaction with $\mathcal{S}$ running on input (f,k) is the same as in* $\mathsf{GBadQ}(f,k)$.

Moreover, suppose events $\mathsf{NotAbort}_{1,\mathcal{S}}^{\mathcal{A}}(f,k)$ *and* $\mathsf{NotAbort}_{2,\mathcal{S}}^{\mathcal{A}}(f,k)$ *occur. If $T = T_1$ then $\mathcal{A}$'s total view in the interaction with $\mathcal{S}$ running on input (f,k) is the same as in* $\mathsf{GBadQ}(f,k-1)$. *If instead $T = T_2$ then $\mathcal{A}$'s total view in the interaction with $\mathcal{S}$ running on input (f,k) is the same as in* $\mathsf{GBadQ}(f,k)$.

Proof. First observe that Pk has the same distribution as the public parameters seen by $\mathcal{A}$ in both games. The same holds for the answers to the first $(k-1)$ Key Queries and to the last $(q-k)$ Key Queries. Let us now focus on the answer to the k-th Key Query. We have two cases:

Case 1: $y_f^{(k)} = \star$**.** Then the view of $\mathcal{A}$ in the interaction with $\mathcal{S}$ is independent from T (see Case B.1) and, on the other hand, by definition, the two games coincide. Therefore the lemma holds in this case.

Case 2: $y_f^{(k)} \neq \star$**.** Suppose $T = T_1 = A_1^{\alpha\beta} \cdot D_4$ and that $\mathsf{NotAbort}_{1,\mathcal{S}}^{\mathcal{A}}(f,k)$ occurs. Therefore, $y_{\hat{\jmath}}^{(k)} = \hat{b}$ and $\mathcal{S}$'s answer to the k-th key query has the same distributions as in $\mathsf{GBadQ}(f,k-1)$. Indeed, we have that $Y_{\hat{\jmath}} = g_1^{a'_{\hat{\jmath}}/r_{\hat{\jmath},\hat{b}}} \cdot g_2^{a''_{\hat{\jmath}}/v_{\hat{\jmath},\hat{b}}} \cdot D_4 \cdot W_{\hat{\jmath}}$ with $a'_{\hat{\jmath}} = \alpha$ and $r_{\hat{\jmath},\hat{b}} = 1/\beta$ and $Y_h = g_1^{-(a'_{\hat{\jmath}}+s')/r_{h,y_h^{(k)}}} \cdot g_2^{-(a''_{\hat{\jmath}}+s'')/v_{h,y_h^{(k)}}} \cdot (B_4^{-1/r_{h,y_h^{(k)}}} \cdot W_h)$ and thus the a'_is and a''_is are random and sum up to 0.

On the other hand if T is random in $\mathbb{G}_{p_1p_4}$ and $\mathsf{NotAbort}_{1,\mathcal{S}}^{\mathcal{A}}(f,k)$ occurs, the $\mathbb{G}_{p_1}$ parts of the Y_i's are random and thus the answer to the k-th query of $\mathcal{A}$ is distributed as in $\mathsf{GBadQ}(f,k)$.

For the second part of the lemma, we observe that the challenge ciphertext has the same distribution in both games and that, if $\mathsf{NotAbort}_{2,\mathcal{S}}^{\mathcal{A}}(f,k)$ occurs, $\mathcal{S}$ properly constructs the challenge ciphertext. □

Next we define event $E_f^{\mathcal{A}}$ as the event that in game GPK, the adversary $\mathcal{A}$ declares two challenge vectors that differ in the f-th component. When the adversary $\mathcal{A}$ is clear from the context we will simply write E_f.

Next we define event $E_{f,G}^{\mathcal{A}}$ as the event that in game G the adversary $\mathcal{A}$ declares two challenge vectors that differ in the f-th component. When the

adversary $\mathcal{A}$ is clear from the context we will simply write $E_{f,G}$. We extend the definition of $E_{f,G}$ to include the game played by $\mathcal{A}$ against the simulator $\mathcal{S}$. Thus we denote by $E^{\mathcal{A}}_{f,\mathcal{S}}(f',k)$ the event that in the interaction between $\mathcal{A}$ and $\mathcal{S}$ on input f' and k, $\mathcal{S}$ does not abort and $\mathcal{A}$ declares two challenge vectors that differ in the f-th component. If $\mathcal{A}, f'$ and k are clear from the context, we will simply write $E_{f,\mathcal{S}}$.

Lemma 5. *If Assumption 2 holds, then for $k = 1,\ldots,q$ and $f = 1,\ldots,\ell+1$, and for all PPT adversaries $\mathcal{A}$, $\left|\text{Prob}[E^{\mathcal{A}}_{f,G}] - \text{Prob}[E^{\mathcal{A}}_{f,H}]\right|$ and $\left|\text{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,G}] - \text{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,H}]\right|$ are negligible functions of λ, for games $G = \mathsf{GBadQ}(f, k-1)$ and $H = \mathsf{GBadQ}(f,k)$.*

Proof. We prove the lemma for $E_{f,G}$ and $E_{f,H}$. A similar reasoning holds for $\mathsf{NotAbort}^{\mathcal{A}}_{2,G}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,H}$. For the sake of contradiction, suppose that $\text{Prob}[E^{\mathcal{A}}_{f,G}] \geq \text{Prob}[E^{\mathcal{A}}_{f,H}] + \epsilon$ for some non-negligible ϵ. Then we can modify simulator $\mathcal{S}$ into algorithm $\mathcal{B}$ with a non-negligible advantage in breaking Assumption 2. Algorithm $\mathcal{B}$ simply execute $\mathcal{S}$'s code. By Lemma 2 event $\mathsf{NotAbort}_{1,\mathcal{S}}$ occurs with probability at least $1/\ell$ and in this case $\mathcal{B}$ can continue the execution of $\mathcal{S}$'s code and receive the challenge vectors from $\mathcal{A}$. At this point, $\mathcal{B}$ checks whether they differ in the f-th component. If they do, $\mathcal{B}$ outputs 1; else $\mathcal{B}$ outputs 0. It is easy to see that, by Lemma 4, the above algorithm has a non-negligible advantage in breaking Assumption 2. □

The proof of the following corollary is straightforward from Lemma 5 and Observations 1-3.

Corollary 1. *For all $f = 1,\ldots,\ell+1$ and $k = 0,\ldots,q$, and all PPT adversaries $\mathcal{A}$, we have that, for $H = \mathsf{GBadQ}(f,k)$ $\left|\text{Prob}[E^{\mathcal{A}}_{f,H}] - \text{Prob}[E^{\mathcal{A}}_{f}]\right|$ and $\left|\text{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,H}] - \text{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2}]\right|$ are negligible.*

We define event $\mathsf{Succ}^{\mathcal{A}}(f,k)$ as

$$\mathsf{Succ}^{\mathcal{A}}(f,k) := \mathsf{NotAbort}^{\mathcal{A}}_{1,\mathcal{S}}(f,k) \wedge \ \mathsf{NotAbort}^{\mathcal{A}}_{2,\mathcal{S}}(f,k) \wedge E^{\mathcal{A}}_{f,\mathcal{S}}(f,k). \qquad (1)$$

We are now ready to prove Lemma 6.

Lemma 6. *Suppose there exists an adversary $\mathcal{A}$ and integers $1 \leq f \leq \ell+1$ and $1 \leq k \leq q$ such that $\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \geq \epsilon$, where $G = \mathsf{GBadQ}(f,k-1)$, $H = \mathsf{GBadQ}(f,k)$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathsf{Adv}^{\mathcal{B}}_2 \geq \text{Prob}[E_f] \cdot \epsilon/(2\cdot\ell^2) - \nu(\lambda)$, for a negligible function ν.*

Proof. Assume without loss of generality that $\mathsf{Adv}^{\mathcal{A}}[G] \geq \mathsf{Adv}^{\mathcal{A}}[H] + \epsilon$ and consider the following algorithm $\mathcal{B}$. $\mathcal{B}$ uses simulator $\mathcal{S}$ as a subroutine and interacts with $\mathcal{A}$ on input integers f and k for which the above inequality holds, and an instance (D,T) of Assumption 2. If event $\mathsf{Succ}^{\mathcal{A}}(f,k)$ does not occur, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ receives $\mathcal{A}$'s output η' and checks if $\eta = \eta'$ (recall that

η is the random bit chosen by $\mathcal{S}$ in preparing the challenge ciphertext). If $\eta = \eta'$ then $\mathcal{B}$ outputs 1; otherwise $\mathcal{B}$ outputs 0. Therefore we have

$$\text{Prob}[\mathcal{B} \text{ outputs } 1|T = T_1] = \text{Prob}[\mathcal{B} \text{ outputs } 1|T = T_1 \wedge \mathsf{Succ}^{\mathcal{A}}(f,k)] \cdot \text{Prob}[\mathsf{Succ}_{\mathcal{A}}(f,k)|T = T_1] \quad (2)$$

By definition of $\mathsf{Succ}^{\mathcal{A}}(f,k)$ we have $\text{Prob}[\mathsf{Succ}_{\mathcal{A}}(f,k)|T = T_1] = \text{Prob}[E_{f,\mathcal{S}} \wedge \mathsf{NotAbort}_{1,\mathcal{S}} \wedge \mathsf{NotAbort}_{2,\mathcal{S}}|T = T_1] = \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}|T = T_1] \cdot \text{Prob}[E_{f,\mathcal{S}} \wedge \mathsf{NotAbort}_{2,\mathcal{S}}|\mathsf{NotAbort}_{1,\mathcal{S}} \wedge T = T_1]$. Now observe that event $\mathsf{NotAbort}_{1,\mathcal{S}}$ is determined before $\mathcal{S}$ uses T and thus $\text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}|T = T_1] = \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}]$. Moreover, by Lemma 4, if event $\mathsf{NotAbort}_{1,\mathcal{S}}$ occurs and $T = T_1$, the view of $\mathcal{A}$ up to Challenge Query is equal to the view of $\mathcal{A}$ in game G and thus $\text{Prob}[E_{f,\mathcal{S}} \wedge \mathsf{NotAbort}_{2,\mathcal{S}}|\mathsf{NotAbort}_{1,\mathcal{S}} \wedge T = T_1] = \text{Prob}[E_{f,G} \wedge \mathsf{NotAbort}_{2,G}]$ whence $\text{Prob}[\mathsf{Succ}^{\mathcal{A}}(f,k)|T = T_1] = \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,G} \wedge E_{f,G}] = \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,G}] \cdot \text{Prob}[E_{f,G}]$, where $\mathsf{NotAbort}_{2,G}$ and $E_{f,G}$ are independent. Finally, if $T = T_1$ and $\mathsf{Succ}^{\mathcal{A}}(f,k)$ occures, then, by Lemma 4, $\mathcal{A}$'s view is exactly as in game G, and thus the probability that $\mathcal{B}$ outputs 1 is equal to the probability that $\mathcal{A}$ wins in game G. We can thus rewrite Eq. 2 as $\text{Prob}[\mathcal{B} \text{ outputs } 1|T = T_1] = \text{Prob}[\mathcal{A} \text{ wins in } G] \cdot \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,G}] \cdot \text{Prob}[E_{f,G}]$ A similar reasoning yields $\text{Prob}[\mathcal{B} \text{ outputs } 1|T = T_2] = \text{Prob}[\mathcal{A} \text{ wins in } H] \cdot \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,H}] \cdot \text{Prob}[E_{f,H}]$ By using Corollary 1, Lemma 2 and Lemma 3, we can conclude that there exists a negligible function ν such that we have $\mathsf{Adv}_2^{\mathcal{B}} = \text{Prob}[\mathsf{NotAbort}_{1,\mathcal{S}}] \cdot \text{Prob}[\mathsf{NotAbort}_2] \cdot \text{Prob}[E_f] \cdot \Big(\text{Prob}[\mathcal{A} \text{ wins in } G] - \text{Prob}[\mathcal{A} \text{ wins in } H]\Big) - \nu(\lambda) \geq \frac{\epsilon}{2\ell^2} \cdot \text{Prob}[E_f] - \nu(\lambda)$. □

The following Lemma can be proved by referring to simulator $\mathcal{S}_2$. We omit further details since the proof is essentially the same as the one of Lemma 6.

Lemma 7. *Suppose there exists an adversary $\mathcal{A}$ and integers $1 \leq f \leq \ell+1$ and $1 \leq k \leq q$ such that $\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \geq \epsilon$, where $G = \mathsf{GBadQ2}(f, k-1)$, $H = \mathsf{GBadQ2}(f,k)$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathsf{Adv}_2^{\mathcal{B}} \geq \text{Prob}[E_f] \cdot \epsilon/(2 \cdot \ell^2) - \nu(\lambda)$, for a negligible function ν.*

The Advantage of $\mathcal{A}$ in GPK. We prove that, under Assumption 2, every PPT adversary $\mathcal{A}$ has a negligible advantage in $\mathsf{GPK} = \mathsf{GBadCh}(1)$ by proving that it is computationally indistinguishable from $\mathsf{GBadCh}(\ell+1)$ that, by Observation 5, gives no advantage to any adversary. Proof. Let $E^{\mathcal{A}}_{f,f'}$ denote the event that during the execution of $\mathsf{GBadCh}(f')$ adversary $\mathcal{A}$ outputs two challenge vectors that differ in the f-th component. For an event E, we define the *advantage* $\mathsf{Adv}^{\mathcal{A}}[G|E]$ *of* $\mathcal{A}$ *in* G *conditioned on event* E as $\mathsf{Adv}^{\mathcal{A}}[G|E] = \text{Prob}[\mathcal{A} \text{ wins in game } G|E] - \frac{1}{2}$.

Observation 6. *For all PPT adversaries $\mathcal{A}$ and all $1 \leq f \leq \ell$, we have that $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|\neg E_{f,f}] = \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|\neg E_{f,f+1}]$.*

PROOF. By definition of GBadCh, if the challenge vectors coincide in the f-th component, then $\mathcal{A}$'s view in $\mathsf{GBadCh}(f)$ and $\mathsf{GBadCh}(f+1)$ are the same. □

Observation 7. *For all PPT adversaries $\mathcal{A}$ and all $1 \leq f \leq \ell$, we have that* $\mathrm{Prob}[E^{\mathcal{A}}_{f,f}] = \mathrm{Prob}[E^{\mathcal{A}}_{f,f+1}]$.

PROOF. The view of $\mathcal{A}$ in $\mathsf{GBadCh}(f)$ up to the Challenge Query is independent from f. □

Therefore we can set $\mathrm{Prob}[E^{\mathcal{A}}_{f,f}] = \mathrm{Prob}[E^{\mathcal{A}}_{f,1}] = \mathrm{Prob}[E^{\mathcal{A}}_{f}]$. □

Lemma 8. *If Assumption 2 holds, then, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}]$ is negligible. Specifically, if there is an adversary $\mathcal{A}$ with $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}] = \epsilon$ then there exists an adversary $\mathcal{B}$ against Assumption 2 such that $\mathsf{Adv}^{\mathcal{B}}_2 \geq \frac{\epsilon^2}{2q\ell^4} - \nu(\lambda)$, for some negligible function ν.*

Let $\mathcal{A}$ be a PPT adversary such that $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}] \geq \epsilon$. Since $\mathsf{GPK} = \mathsf{GBadCh}(1)$ and $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(\ell+1)] = 0$ there must exist $f \in [\ell]$ such that

$$\left|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)]\right| \geq \epsilon' = \epsilon/\ell. \tag{3}$$

Now recall that $\mathsf{GBadCh}(f) = \mathsf{GBadQ}(f,0)$ and $\mathsf{GBadCh}(f+1) = \mathsf{GBadQ2}(f,0)$. Thus, there exists k, $1 \leq k \leq q$ such that $\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \geq \epsilon'/(2q)$ where $G = \mathsf{GBadQ}(f,k)$ and $H = \mathsf{GBadQ}(f,k-1)$ or where $G = \mathsf{GBadQ2}(f,k)$ and $H = \mathsf{GBadQ2}(f,k-1)$. Then by Lemma 6, in the former case, and by Lemma 7 in the latter, we can construct an adversary $\mathcal{B}$ against Assumption 2, such that $\mathsf{Adv}^{\mathcal{B}}_2 \geq \frac{\epsilon}{4q\ell^3} \cdot \mathrm{Prob}[E_f] - \nu(\lambda)$ Now it remains to estimate $\mathrm{Prob}[E_f]$. Notice that we can write $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)] = \mathrm{Prob}[E_{f,f}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|E_{f,f}] + \mathrm{Prob}[\neg E_{f,f}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|\neg E_{f,f}]$. and $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)] = \mathrm{Prob}[E_{f,f+1}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|E_{f,f+1}] + \mathrm{Prob}[\neg E_{f,f+1}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)]|\neg E_{f,f+1}]$. and, by combining Equation 3 and Observations 6 and 7, we obtain $\mathrm{Prob}[E_f] \cdot \left|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|E_{f,f}] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|E_{f,f+1}]\right| \geq \epsilon'$. Since no advantage is greater than $1/2$, we can conclude that $\mathrm{Prob}[E_f] \geq 2 \cdot \epsilon'$ and thus $\mathcal{B}$ as advantage $\mathsf{Adv}^{\mathcal{B}}_2 \geq \frac{\epsilon^2}{2q\ell^4} - \nu(\lambda)$

5.3 Wrapping Up

By combining Lemma 1 and Lemma 8 we obtain our main result.

Theorem 8. *If Assumption 1 and 2 hold, then the HVE scheme described in Section 4 is secure (in the sense of Definition 1).*

6 Reductions

In this section we show how to construct an encryption scheme for the class of *Boolean predicates* that can be expressed as a k-CNF or k-DNF formula and disjunctions over *binary variables* from an HVE scheme.

Reducing k-CNF to HVE. We consider formulae Φ in k-CNF, for constant k, over n variables in which each clause $C \in \Phi$ contains exactly k distinct variables. We call such a clause *admissible* and denote by $\mathbb{C}_n$ the set of all admissible clauses over the n variables $x_1, \ldots, x_n$ and set $M_n = |\mathbb{C}|$. Notice that $M_n = \Theta(n^k)$. We also fix a canonical ordering $C_1, \ldots, C_{M_n}$ of the clauses in $\mathbb{C}_n$. Let $\mathcal{H} = (\mathsf{Setup}_{\mathcal{H}}, \mathsf{KeyGen}_{\mathcal{H}}, \mathsf{Encrypt}_{\mathcal{H}}, \mathsf{Test}_{\mathcal{H}})$ be an HVE scheme and construct a k-CNF scheme $\mathsf{kCNF} = (\mathsf{Setup}_{\mathsf{kCNF}}, \mathsf{KeyGen}_{\mathsf{kCNF}}, \mathsf{Encrypt}_{\mathsf{kCNF}}, \mathsf{Test}_{\mathsf{kCNF}})$ as follows:

$\mathsf{Setup}_{\mathsf{kCNF}}(1^\lambda, 1^n)$: The algorithm returns the output of $\mathsf{Setup}_{\mathcal{H}}(1^\lambda, 1^{M_n})$.

$\mathsf{KeyGen}_{\mathsf{kCNF}}(\mathsf{Msk}, \Phi)$: For a k-CNF formula Φ, the key generation algorithm constructs vector $\boldsymbol{y} \in \{0, 1, \star\}^{M_n}$ by setting, for each $i \in \{1, \ldots, M_n\}$, $y_i = 1$ if $C_i \in \Phi$; $y_i = \star$ otherwise. We denote this transformation by $y = \mathsf{FEncode}(\Phi)$. Then the key generation algorithm returns $\mathsf{Sk}_\Phi = \mathsf{KeyGen}_{\mathcal{H}}(\mathsf{Msk}, \boldsymbol{y})$.

$\mathsf{Encrypt}_{\mathsf{kCNF}}(\mathsf{Pk}, \boldsymbol{z})$: The algorithm constructs vector $\boldsymbol{x} \in \{0, 1\}^{M_n}$ in the following way: For each $i \in \{1, \ldots, M_n\}$ the algorithms sets $x_i = 1$ if C_i is satisfied by $\boldsymbol{z}$; $x_i = 0$ if C_i is not satisfied by $\boldsymbol{z}$. We denote this transformation by $\boldsymbol{x} = \mathsf{AEncode}(\boldsymbol{z})$. Then the encryption algorithm returns $\mathsf{Ct} = \mathsf{Encrypt}_{\mathcal{H}}(\mathsf{Pk}, \boldsymbol{x})$.

$\mathsf{Test}_{\mathsf{kCNF}}(\mathsf{Sk}_\Phi, \mathsf{Ct})$: The algorithm returns the output of $\mathsf{Test}_{\mathcal{H}}(\mathsf{Sk}_\Phi, \mathsf{Ct})$.

Correctness. It follows from the observation that for tuple $(\Phi, \boldsymbol{z})$, we have that $\mathsf{Match}(\mathsf{AEncode}(\boldsymbol{z}), \mathsf{FEncode}(\Phi)) = 1$ if and only if $\mathsf{Satisfy}(\Phi, \boldsymbol{z}) = 1$.

Security. Let $\mathcal{A}$ be an adversary for kCNF that tries to break the scheme for n variables and consider the following adversary $\mathcal{B}$ for $\mathcal{H}$ that uses $\mathcal{A}$ as a subroutine and tries to break a $\mathcal{H}$ with $\ell = M_n$ by interacting with challenger $\mathcal{C}$. $\mathcal{B}$ receives a Pk for $\mathcal{H}$ and passes it to $\mathcal{A}$ Whenever $\mathcal{A}$ asks for the key for formula Φ, $\mathcal{B}$ constructs $\boldsymbol{y} = \mathsf{FEncode}(\Phi)$ and asks $\mathcal{C}$ for a key $\mathsf{Sk}_{\boldsymbol{y}}$ for $\boldsymbol{y}$ and returns it to $\mathcal{A}$. When $\mathcal{A}$ asks for a challenge by providing truth assignments $\boldsymbol{z}_0$ and $\boldsymbol{z}_1$, $\mathcal{B}$ simply computes $\boldsymbol{x}_0 = \mathsf{AEncode}(\boldsymbol{z}_0)$ and $\boldsymbol{x}_1 = \mathsf{AEncode}(\boldsymbol{z}_1)$ and gives the pair $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ to $\mathcal{C}$. $\mathcal{B}$ then returns the challenge ciphertext obtained from $\mathcal{C}$ to $\mathcal{A}$. Finally, $\mathcal{B}$ outputs $\mathcal{A}$'s guess.

First, $\mathcal{B}$'s simulation is perfect. Indeed, we have that if for all $\mathcal{A}$'s queries Φ we have that $\mathsf{Satisfy}(\Phi, \boldsymbol{z}_0) = \mathsf{Satisfy}(\Phi, \boldsymbol{z}_1)$, then all $\mathcal{B}$'s queries $\boldsymbol{y}$ to $\mathcal{C}$ also have the property $\mathsf{Match}(\boldsymbol{y}, \boldsymbol{x}_0) = \mathsf{Match}(\boldsymbol{y}, \boldsymbol{x}_1)$. Thus $\mathcal{B}$'s advantage is the same as $\mathcal{A}$'s. By combining the above reduction with our constructions for HVE:

Theorem 9. *For any constant $k > 0$, if Assumption 1 and 2 hold for generator $\mathcal{G}$ then there exists a secure encryption scheme for the class of predicates that can be represented by k-CNF formulae.*

Reducing Disjunctions to HVE. In this section we consider the class of Boolean predicates that can be expressed as a single disjunction. We assume without loss of generality that a disjunction does not contain a variable and its negated. Let $\mathcal{H} = (\mathsf{Setup}_{\mathcal{H}}, \mathsf{KeyGen}_{\mathcal{H}}, \mathsf{Encrypt}_{\mathcal{H}}, \mathsf{Test}_{\mathcal{H}})$ be an HVE scheme and construct the predicate-only scheme $\vee = (\mathsf{Setup}_{\vee}, \mathsf{KeyGen}_{\vee}, \mathsf{Encrypt}_{\vee}, \mathsf{Test}_{\vee})$ for disjunctions in the following way:

$\mathsf{Setup}_{\vee}(1^\lambda, 1^n)$: the algorithm returns the output of $\mathsf{Setup}_{\mathcal{H}}(1^\lambda, 1^n)$.

$\mathsf{KeyGen}_{\vee}(\mathsf{Msk}, C)$: For a clause C, the key generation algorithm constructs vector $\boldsymbol{y} \in \{0, 1, \star\}^n$ in the following way. Let $\boldsymbol{w}$ be a truth assignment to the n variables that does not satisfy clause C. For each $i \in \{1, \ldots, n\}$, the algorithms

sets $y_i = w_i$ if the i-th variable appears in C; $y_i = \star$ otherwise. We denote this transformation by $\boldsymbol{y} = \mathsf{CEncode}(C)$. The output is $\mathsf{Sk}_C = \mathsf{KeyGen}_{\mathcal{H}}(\mathsf{Msk}, \boldsymbol{y})$.

$\mathsf{Encrypt}_{\vee}(\mathsf{Pk}, \boldsymbol{z})$: The encryption algorithm returns $\mathsf{Ct} = \mathsf{Encrypt}_{\mathcal{H}}(\mathsf{Pk}, \boldsymbol{z})$.

$\mathsf{Test}_{\vee}(\mathsf{Sk}_C, \mathsf{Ct})$: The algorithm returns $1 - \mathsf{Test}_{\mathcal{H}}(\mathsf{Sk}_C, \mathsf{Ct})$.

Correctness. It follows from the observation that for a clause C and assignment $\boldsymbol{z}$, $\mathsf{Satisfy}(C, \boldsymbol{z}) = 1$ if and only if $\mathsf{Match}(\mathsf{CEncode}(C), \boldsymbol{z}) = 0$.

Security. If $\mathcal{H}$ is secure then $\vee$ is secure. In particular, notice that if for $\mathcal{A}$'s query C we have that $\mathsf{Satisfy}(C, \boldsymbol{z}_0) = \mathsf{Satisfy}(C, \boldsymbol{z}_1) = \xi \in \{0, 1\}$, then for $\mathcal{B}$'s query $\boldsymbol{y} = \mathsf{CEncode}(C)$ to $\mathcal{C}$ we have that $\mathsf{Match}(\boldsymbol{y}, \boldsymbol{z}_0) = \mathsf{Match}(\boldsymbol{y}, \boldsymbol{z}_1) = 1 - \xi$.

Theorem 10. *If Assumption* 1 *and* 2 *hold for generator $\mathcal{G}$ then there exists a secure encryption scheme for disjunctions.*

Reducing k-DNF to k-CNF. We observe that if Φ is a predicate represented by a k-DNF formula over *binary variables* then its negation $\bar{\Phi}$ can be represented by a k-CNF formula. Therefore let $\mathsf{kCNF} = (\mathsf{Setup}_{\mathsf{kCNF}}, \mathsf{KeyGen}_{\mathsf{kCNF}}, \mathsf{Encrypt}_{\mathsf{kCNF}}, \mathsf{Test}_{\mathsf{kCNF}})$ and consider the following scheme $\mathsf{kDNF} = (\mathsf{Setup}_{\mathsf{kDNF}}, \mathsf{KeyGen}_{\mathsf{kDNF}}, \mathsf{Encrypt}_{\mathsf{kDNF}}, \mathsf{Test}_{\mathsf{kDNF}})$. The setup algorithm $\mathsf{Setup}_{\mathsf{kDNF}}$ is the same as $\mathsf{Setup}_{\mathsf{kCNF}}$. The key generation algorithm $\mathsf{Setup}_{\mathsf{kDNF}}$ for predicate Φ represented by a k-DNF simply invokes the key generation algorithm $\mathsf{Setup}_{\mathsf{kCNF}}$ for $\bar{\Phi}$ that can be represented by a k-CNF formula. The encryption algorithm $\mathsf{Encrypt}_{\mathsf{kDNF}}$ is the same as $\mathsf{Encrypt}_{\mathsf{kCNF}}$. The test algorithm $\mathsf{Test}_{\mathsf{kDNF}}$ on input ciphertext Ct and key for k-DNF formula Φ (that is actually a key for k-CNF formula $\bar{\Phi}$) thus $\mathsf{Test}_{\mathsf{kCNF}}$ on Ct and the key and complements the result. Correctness and security can be easily argued as for Disjunctions. By combining the above reduction with the construction given by Theorem 9.

Theorem 11. *If Assumption* 1 *and* 2 *hold for generator $\mathcal{G}$ then there exists a secure encryption scheme for k-DNF formulae.*

Acknowledgements. We would like to thank the anonymous referees for their helpful comments and suggestions. The work of the authors has been supported in part by the European Commission through the EU ICT program under Contract ICT-2007-216676 ECRYPT II, and by the italian Ministry of University and Research Project PRIN 2008 *PEPPER: Privacy and Protection of Personal Data* (prot. 2008SY2PH4)."

References

1. Boneh, D.: Bilinear Groups of Composite Order. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 39–56. Springer, Heidelberg (2007)
2. Boneh, D., Boyen, X.: Secure Identity Based Encryption Without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)

3. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
4. 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)
5. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
6. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
7. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
8. Gentry, C., Halevi, S.: Hierarchical Identity Based Encryption with Polynomially Many Levels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 437–456. Springer, Heidelberg (2009)
9. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-Based Encryption for Fine-Grained Access Control for Encrypted Data. In: ACM CCS 2006, Alexandria, VA, USA, October 30-November 3, pp. 89–98 (2006)
10. Iovino, V., Persiano, G.: Hidden-Vector Encryption with Groups of Prime Order. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 75–88. Springer, Heidelberg (2008)
11. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
12. Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
13. Okamoto, T., Takashima, K.: Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012)
14. O'Neill, A.: Definitional issues in functional encryption. Technical Report 2010-556, Cryptology ePrint Archives (2010), http://eprint.iacr.org/2010/556/
15. Sedghi, S., van Liesdonk, P., Nikova, S., Hartel, P., Jonker, W.: Searching Keywords with Wildcards on Encrypted Data. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 138–153. Springer, Heidelberg (2010)
16. Shi, E., Waters, B.: Delegating Capabilities in Predicate Encryption Systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)
17. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
18. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)

Shorter IBE and Signatures via Asymmetric Pairings

Jie Chen[1], Hoon Wei Lim[1], San Ling[1], Huaxiong Wang[1], and Hoeteck Wee[2,1]

[1] Division of Mathematical Sciences
School of Physical & Mathematical Sciences
Nanyang Technological University, Singapore
[2] Department of Computer Science
George Washington University, USA
s080001@e.ntu.edu.sg,
{hoonwei,lingsan,hxwang}@ntu.edu.sg,
hoeteck@gwu.edu

Abstract. We present efficient Identity-Based Encryption (IBE) and signature schemes under the Symmetric External Diffie-Hellman (SXDH) assumption in bilinear groups. In both the IBE and the signature schemes, all parameters have constant numbers of group elements, and are shorter than those of previous constructions based on Decisional Linear (DLIN) assumption. Our constructions use both dual system encryption (Waters, Crypto '09) and dual pairing vector spaces (Okamoto and Takashima, Pairing '08, Asiacrypt '09). Specifically, we show how to adapt the recent DLIN-based instantiations of Lewko (Eurocrypt '12) to the SXDH assumption. To our knowledge, this is the first work to instantiate either dual system encryption or dual pairing vector spaces under the SXDH assumption.

1 Introduction

Identity-Based Encryption. The idea of using a user's identity as her public encryption key, and thus eliminating the need for a public key certificate, was conceived by Shamir [34]. Such a primitive is known as Identity-Based Encryption (IBE), which has been extensively studied particularly over the last decade. We now have constructions of IBE schemes from a large class of assumptions, namely pairings, quadratic residuosity and lattices, starting with the early constructions in the random oracle model [9, 16, 22], to more recent constructions in the standard model [14, 7, 8, 15, 2].

Short IBE. It is desirable that an IBE scheme be as efficient as possible, if it were to have any impact on practical applications. Ideally, we would like to have constant-size public parameters, secret keys, and ciphertexts. Moreover, the scheme should ideally achieve full security, namely to be resilient even against an adversary that adaptively selects an identity to attack based on previous secret keys. The first fully secure efficient IBE with constant-size public parameters and ciphertexts under standard assumptions was obtained by Waters [37]

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 122–140, 2013.

in 2009; this scheme relied on the Decisional Bilinear Diffie-Hellman (DBDH) and Decisional Linear (DLIN) assumptions. Since then, Lewko and Waters [26] and Lewko [25] gave additional fully secure efficient IBE schemes that achieve incomparable guarantees. Prior to these works, all known IBEs (in the standard model) were either selectively secure [14, 7, 15, 2], or require long parameters [8, 36, 15, 2], or were based on less standard assumptions that depended on the query complexity of the adversary [21]. From a practical stand-point, Waters' fully secure IBE [37] is still not very efficient as it has relatively large ciphertexts and secret keys, i.e., eleven and nine group elements,[1] respectively. Lewko's scheme [25] improved on both of these parameters at the cost of larger public parameters and master key.

Shorter IBE? In his work, Waters also suggested obtaining even more efficient IBE schemes by turning to asymmetric bilinear groups:

> Using the SXDH assumption we might hope to shave off three group elements from both ciphertexts and private keys.

In fact, improving the efficiency of a scheme using asymmetric pairings was first observed by Boneh, Boyen and Shacham [10]. At a fixed security level, group elements in the asymmetric setting are smaller and pairings can be computed more efficiently [19]. (Estimated bit sizes of group elements for bilinear group generators are given in Appendix A.) Informally, the SXDH assumption states that there are prime-order groups (G_1, G_2, G_T) that admits a bilinear map $e : G_1 \times G_2 \to G_T$ such that the Decisional Diffie-Hellman (DDH) assumption holds in both G_1 and G_2. The SXDH assumption was formally defined by Ballard et al. [4] in their construction of a searchable encryption scheme, and has since been used in a number of different contexts, including secret-handshake schemes [3], anonymous IBE [17], continual leakage-resilience [12], and most notably, Groth-Sahai proofs [24]. Evidence for the validity of this assumption were presented in the works of Verheul [35] and Galbraith and Rotger [20].

1.1 Our Contributions

In this work, we present a more efficient IBE scheme under the SXDH assumption; our scheme also achieves anonymity.[2] The ciphertexts and secret keys consist of only five and four group elements, respectively. That is, we shave off two group elements from both ciphertexts and private keys in Lewko's DLIN-based IBE [25]. See Table 1 for a summary of comparisons between existing and our IBE schemes, where λ is the security parameter. Applying Naor's transform [9, 11] to our scheme, we also obtain an efficient signature scheme.

[1] Here, we do not separately consider group elements from target groups of pairings, although a ciphertext typically has a group element that is from an associated target group.

[2] It follows from our analysis that Lewko's IBE [25] is also anonymous, although this was not pointed out in her paper.

Table 1. Comparison between existing and our IBE schemes

Source	# PP	# MK	# SK	# CT	# pairing	anonymity	assumptions
Waters [36]	$\mathcal{O}(\lambda)$	1	2	3	2	No	DBDH
Waters [37]	13	5	9	11	9	No	DLIN,DBDH
Lewko [25]	25	30	6	7	6	Yes	DLIN
RCS [33]	9	7	7	9	7	No	XDH, DLIN, DBDH
Ours	9	9	4	5	4	Yes	SXDH

Our Approach. As with all known fully secure efficient IBEs, our construction relies on Waters' dual system encryption framework [37]. Following Lewko's DLIN-based IBE [25], we instantiate dual system encryption under the SXDH assumption via dual pairing vector spaces [29, 30], which is a technique to achieve orthogonality in prime-order groups. This is the first work to instantiate either dual system encryption or dual pairing vector spaces under the SXDH assumption. We proceed to highlight several salient features of our IBE scheme in relation to Lewko's IBE [25]:

- Our scheme has an extremely simple structure, similar to the selectively secure IBE of Boneh and Boyen [7], as well as the fully secure analogues given by Lewko and Waters [26] and Lewko [25].
- By shifting from the DLIN assumption to the simpler SXDH assumption, we obtain IBE schemes that are syntactically simpler and achieve shorter parameters. Specifically, Lewko's IBE scheme [25] relies on 6 basis vectors to simulate the subgroup structure in the Lewko-Waters IBE scheme [26], whereas our construction uses only 4 basis vectors. This means that we can use a 4-dimensional vector space instead of a 6-dimensional one. As a result, we save two group elements in both the secret key and the ciphertext, that is, by a factor of 1/3. The savings for the public parameters and master key is even more substantial, because we use only two basis vectors for the main scheme, as opposed to four basis vectors in Lewko's scheme. In both our scheme and in Lewko's, the remaining two basis vectors are used for the semi-functional components in the proof of security.
- The final step of the proof of security (after switching to semi-functional secret keys and ciphertexts) is different from that of Lewko's. We rely on an information theoretic argument similar to that in [32] instead of computational arguments.

Finally, we believe that our SXDH instantiations constitute a simpler demonstration of the power of dual pairing vector spaces.

Independent Work of Ramanna et al. An independent work of Ramanna, Chatterjee and Sarkar [33] also demonstrated how to obtain more efficient fully secure IBE via asymmetric pairings. Similar to our work, their constructions rely on dual system encryption; however, they do not make use of dual pairing vector spaces. Our constructions achieve shorter ciphertexts and secret keys than their

work, while relying on a single assumption (whereas their construction relies on a triplet of assumptions). Moreover, our scheme achieves anonymity; theirs does not. Finally, they obtain their schemes via careful optimizations, whereas our scheme is derived via a more general framework.

Outline. In Section 2, we present the preliminaries, including security definitions for IBE, our security assumptions, and an overview of dual pairing vector spaces. In Section 3, we present the subspace assumptions based on SXDH. Finally, we present our IBE and signature schemes in Sections 4 and 5.

2 Preliminaries

In this section, we first recall the definitions of security for IBE, and signatures. We then present a few backgrounds related to groups with efficiently computable bilinear maps and define the Symmetric External Diffie-Hellman assumption.

2.1 Identity-Based Encryption and Signatures

Identity-Based Encryption. An Identity-Based Encryption [9] scheme consists of four algorithms: $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{Setup}(\lambda) \to \mathsf{PP}, \mathsf{MK}$ The setup algorithm takes in the security parameter λ, and outputs the public parameters PP, and the master key MK.
- $\mathsf{KeyGen}(\mathsf{PP}, \mathsf{MK}, ID) \to \mathsf{SK}_{ID}$ The key generation algorithm takes in the master key MK, and an identity ID, and produces a secret key SK_{ID} for that identity.
- $\mathsf{Enc}(\mathsf{PP}, ID, M) \to \mathsf{CT}_{ID}$ The encryption algorithm takes in an identity ID, and a message M, and outputs a ciphertext CT_{ID} encrypted under that identity.
- $\mathsf{Dec}(\mathsf{PP}, \mathsf{SK}_{ID}, \mathsf{CT}_{ID}) \to M$ The decryption algorithm takes in a secret key SK_{ID}, and a ciphertext CT_{ID}, and outputs the message M when the CT_{ID} is encrypted under the same ID.

Anonymous IBE. The security notion of anonymous IBE was formalized by [1], which is defined by the following game, played by a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$.

- Setup The challenger $\mathcal{B}$ runs the setup algorithm to generate PP and MK. It gives PP to the adversary $\mathcal{A}$.
- $\mathsf{Phase\ 1}$ The challenger $\mathcal{A}$ adaptively requests keys for identities ID, and is provided with corresponding secret keys SK_{ID}, which the challenger $\mathcal{B}$ generates by running the key generation algorithm.
- $\mathsf{Challenge}$ The adversary $\mathcal{A}$ gives the challenger $\mathcal{B}$ two challenge pairs (M_0, ID_0^*) and (M_1, ID_1^*). The challenge identities must not have been queried in Phase 1. The challenger sets $\beta \in \{0, 1\}$ randomly, and encrypts M_β under ID_β^* by running the encryption algorithm. It sends the ciphertext to the adversary $\mathcal{A}$.

- Phase 2 This is the same as Phase 1, with the added restriction a secret key for ID_0^*, ID_1^* cannot be requested.
- Guess The adversary $\mathcal{A}$ must output a guess β' for β.

The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IBE}}(\lambda)$ of an adversary $\mathcal{A}$ is defined to be $\Pr[\beta' = \beta] - 1/2$.

Definition 1. *An Identity-Based Encryption scheme is secure and anonymous if all PPT adversaries achieve at most a negligible advantage in the above security game.*

Remark: The security notion of non-anonymous IBE is defined as above with restriction that $ID_0^* = ID_1^*$.

Signatures. A signature scheme is made up of three algorithms, $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ for generating keys, signing, and verifying signatures, respectively.

- $\mathsf{KeyGen}(1^\lambda) \rightarrow \mathsf{PK}, \mathsf{SK}$ The key generation algorithm takes in the security parameter λ, and outputs the public key PK, and the secret key SK.
- $\mathsf{Sign}(\mathsf{SK}, M) \rightarrow \sigma$ The signing algorithm takes in the secret key SK, and a message M, and produces a signature σ for that message.
- $\mathsf{Verify}(\mathsf{PK}, \sigma, M) \rightarrow \mathsf{CT}$ The verifying algorithm takes in the public key PK, and a signature pair (σ, M), and outputs `valid` or `invalid`.

The standard notion of security for a signature scheme is called existential unforgeability under a chosen message attack [23], which is defined using the following game between a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$.

- Setup The challenger $\mathcal{B}$ runs the setup algorithm to generate PK and SK. It gives PK to the adversary $\mathcal{A}$.
- Queries The adversary $\mathcal{A}$ adaptively requests for messages $M_1, \ldots, M_\nu \in \{0,1\}^*$, and is provided with corresponding signatures $\sigma_1, \ldots, \sigma_\nu$ by running the sign algorithm Sign.
- Output Eventually, the adversary $\mathcal{A}$ outputs a pair (M, σ).

The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Sig}}(\lambda)$ of an adversary $\mathcal{A}$ is defined to be the probability that $\mathcal{A}$ wins in the above game, namely

(1) M is not any of $M_1, \ldots, M_\nu$;
(2) $\mathsf{Verify}(\mathsf{PK}, \sigma, M)$ outputs `valid`.

Definition 2. *A signature scheme is existentially unforgeable under an adaptive chosen message attack if all PPT adversaries achieve at most a negligible advantage in the above security game.*

We assume that for any PPT algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the above game is negligible in the security parameter λ.

2.2 Dual Pairing Vector Spaces

Our constructions are based on dual pairing vector spaces proposed by Okamoto and Takashima [29, 30]. In this paper, we concentrate on the asymmetric version [32]. We only briefly describe how to generate random dual orthonormal bases. See [29, 30, 32] for a full definition of dual pairing vector spaces.

Definition 3. *"Asymmetric bilinear pairing groups"* $(q, G_1, G_2, G_T, g_1, g_2, e)$ *are a tuple of a prime* q*, cyclic (multiplicative) groups* G_1, G_2 *and* G_T *of order* q*,* $g_1 \neq 1 \in G_1$*,* $g_2 \neq 1 \in G_2$*, and a polynomial-time computable nondegenerate bilinear pairing* $e : G_1 \times G_2 \to G_T$ *i.e.,* $e(g_1^s, g_2^t) = e(g_1, g_2)^{st}$ *and* $e(g_1, g_2) \neq 1$*.*

In addition to referring to individual elements of G_1 or G_2, we will also consider "vectors" of group elements. For $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$ and $g_\beta \in G_\beta$, we write $g_\beta^{\boldsymbol{v}}$ to denote a n-tuple of elements of G_β for $\beta = 1, 2$:

$$g_\beta^{\boldsymbol{v}} := (g_\beta^{v_1}, \ldots, g_\beta^{v_n}).$$

For any $a \in \mathbb{Z}_q$ and $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{Z}_q^n$, we have:

$$g_\beta^{a\boldsymbol{v}} := (g_\beta^{av_1}, \ldots, g_\beta^{av_n}), \quad g_\beta^{\boldsymbol{v}+\boldsymbol{w}} := (g_\beta^{v_1+w_1}, \ldots, g_\beta^{v_n+w_n}).$$

Then we define

$$e(g_1^{\boldsymbol{v}}, g_2^{\boldsymbol{w}}) := \prod_{i=1}^{n} e(g_1^{v_i}, g_2^{w_i}) = e(g_1, g_2)^{\boldsymbol{v}\cdot\boldsymbol{w}}.$$

Here, the dot product is taken modulo q.

Dual Pairing Vector Spaces. For a fixed (constant) dimension n, we will choose two random bases $\mathbb{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ and $\mathbb{B}^* := (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*)$ of $\mathbb{Z}_q^n$, subject to the constraint that they are "dual orthonormal", meaning that

$$\boldsymbol{b}_i \cdot \boldsymbol{b}_j^* = 0 \,(\mathrm{mod}\ q)$$

whenever $i \neq j$, and

$$\boldsymbol{b}_i \cdot \boldsymbol{b}_i^* = \psi \,(\mathrm{mod}\ q)$$

for all i, where ψ is a random element of $\mathbb{Z}_q$. We denote such algorithm as $\mathsf{Dual}(\cdot)$.

Then for generators $g_1 \in G_1$ and $g_2 \in G_2$, we have

$$e(g_1^{\boldsymbol{b}_i}, g_2^{\boldsymbol{b}_j^*}) = 1$$

whenever $i \neq j$, where 1 here denotes the identity element in G_T.

2.3 SXDH Assumptions

Definition 4. *[DDH1: Decisional Diffie-Hellman Assumption in G_1] Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} := (q, G_1, G_2, G_T, g_1, g_2, e) \xleftarrow{R} \mathcal{G},$$
$$a, b, c \xleftarrow{R} \mathbb{Z}_q,$$
$$D := (\mathbb{G}; g_1, g_2, g_1^a, g_1^b).$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DDH1}}(\lambda) := \left|\Pr[\mathcal{A}(D, g_1^{ab}) - \Pr[\mathcal{A}(D, g_1^{ab+c})]\right|.$$

is negligible in the security parameter λ.

The dual of above assumption is Decisional Diffie-Hellman assumption in G_2 (denoted as DDH2), which is identical to Definitions 4 with the roles of G_1 and G_2 reversed. We say that:

Definition 5. *The Symmetric External Diffie-Hellman assumption holds if DDH problems are intractable in both G_1 and G_2.*

2.4 Statistical Indistinguishability Lemma

We require the following lemma from [27] in our security proof.

Lemma 1. *Let $C := \{(\boldsymbol{x}, \boldsymbol{v}) | \boldsymbol{x} \cdot \boldsymbol{v} \neq 0, \boldsymbol{x}, \boldsymbol{v} \in \mathbb{Z}_q^n\}$. For all $(\boldsymbol{x}, \boldsymbol{v}) \in C$, $(\boldsymbol{r}, \boldsymbol{w}) \in C$, $\rho, \tau \leftarrow \mathbb{Z}_q$, and $A \xleftarrow{R} \mathbb{Z}_q^{n \times n}$,*

$$\Pr[\boldsymbol{x}(\rho A^{-1}) = \boldsymbol{r} \wedge \boldsymbol{v}(\tau A^t) = \boldsymbol{w}] = \frac{1}{\#C},$$

where $\#C = (q^n - 1)(q^n - q^{n-1})$.

In other words, $(\rho \boldsymbol{x} A^{-1})$ and $(\tau \boldsymbol{v} A^t)$ are uniformly and independently distributed (i.e., equivalently distributed to $(\boldsymbol{r}, \boldsymbol{w}) \xleftarrow{R} \mathbb{Z}_q^n \times \mathbb{Z}_q^n$ where $\boldsymbol{r} \cdot \boldsymbol{w} \neq 0$, while $\Pr[\boldsymbol{r} \cdot \boldsymbol{w} = 0] = 1/q$) when $\boldsymbol{x} \cdot \boldsymbol{v} \neq 0$.

3 Subspace Assumptions via SXDH

In this section, we present Subspace assumptions derived from the SXDH assumption. We will rely on these assumptions later to instantiate our IBE scheme. These are analogues of the DLIN-based Subspace assumptions given in [25, 31].

Definition 6. *[DS1: Decisional Subspace Assumption in G_1] Given a group generator $\mathcal{G}(\cdot)$, define the following distribution:*

$$\begin{aligned}
&\mathbb{G} := (q, G_1, G_2, G_T, g_1, g_2, e) \xleftarrow{R} \mathcal{G}(1^\lambda),\\
&(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} \mathsf{Dual}(\mathbb{Z}_q^n),\\
&\tau_1, \tau_2, \mu_1, \mu_2 \xleftarrow{R} \mathbb{Z}_q,\\
&U_1 := g_2^{\mu_1 \boldsymbol{b}_1^* + \mu_2 \boldsymbol{b}_{k+1}^*}, U_2 := g_2^{\mu_1 \boldsymbol{b}_2^* + \mu_2 \boldsymbol{b}_{k+2}^*}, \ldots, \ldots, U_k := g_2^{\mu_1 \boldsymbol{b}_k^* + \mu_2 \boldsymbol{b}_{2k}^*},\\
&V_1 := g_1^{\tau_1 \boldsymbol{b}_1}, V_2 := g_1^{\tau_1 \boldsymbol{b}_2}, \ldots, V_k := g_1^{\tau_1 \boldsymbol{b}_k},\\
&W_1 := g_1^{\tau_1 \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_{k+1}}, W_2 := g_1^{\tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{b}_{k+2}}, \ldots, W_k := g_1^{\tau_1 \boldsymbol{b}_k + \tau_2 \boldsymbol{b}_{2k}},\\
D := (\mathbb{G}; &g_2^{\boldsymbol{b}_1^*}, g_2^{\boldsymbol{b}_2^*}, \ldots, g_2^{\boldsymbol{b}_k^*}, g_2^{\boldsymbol{b}_{2k+1}^*}, \ldots, g_2^{\boldsymbol{b}_n^*}, g_1^{\boldsymbol{b}_1}, \ldots, g_1^{\boldsymbol{b}_n}, U_1, U_2, \ldots, U_k, \mu_2),
\end{aligned}$$

where k, n are fixed positive integers that satisfy $2k \leq n$. We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DS1}}(\lambda) := |\Pr[\mathcal{A}(D, V_1, \ldots, V_k) = 1] - \Pr[\mathcal{A}(D, W_1, \ldots, W_k) = 1]|$$

is negligible in the security parameter λ.

For our construction, we only require the assumption for $n = 4, k = 2$. Furthermore, we do not need to provide μ_2 to the distinguisher. Informally, this means that, given:

$$\tau_1, \tau_2, \mu_1, \mu_2 \xleftarrow{R} \mathbb{Z}_q; \qquad \text{and} \qquad U_1 := g_2^{\mu_1 \boldsymbol{b}_1^* + \mu_2 \boldsymbol{b}_3^*}, U_2 := g_2^{\mu_1 \boldsymbol{b}_2^* + \mu_2 \boldsymbol{b}_4^*},$$

the distributions (V_1, V_2) and (W_1, W_2) are computationally indistinguishable, where:

$$\begin{aligned}
&V_1 := g_1^{\tau_1 \boldsymbol{b}_1}, V_2 := g_1^{\tau_1 \boldsymbol{b}_2},\\
&W_1 := g_1^{\tau_1 \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_3}, W_2 := g_1^{\tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{b}_4}.
\end{aligned}$$

Lemma 2. *If the DDH assumption in G_1 holds, then the Subspace assumption in G_1 stated in Definition 6 also holds. More precisely, for any adversary $\mathcal{A}$ against the Subspace assumption in G_1, there exist probabilistic algorithms $\mathcal{B}$ whose running times are essentially the same as that of $\mathcal{A}$, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DS1}}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH1}}(\lambda).$$

Proof. We assume there exists a PPT algorithm $\mathcal{A}$ breaking the Subspace assumption with non-negligible advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DS1}}(\lambda)$ (for some fixed positive integers k, n satisfying $n \geq 2k$). We create a PPT algorithm $\mathcal{B}$ which breaks the DDH assumption in G_1 with non-negligible advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DS1}}(\lambda)$. $\mathcal{B}$ is given $g_1, g_2, g_1^a, g_1^b, T$, where T is either g_1^{ab} or T is a uniformly random element of G_1.

$\mathcal{B}$ first samples random dual orthonormal bases, denoted by $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$ and $\boldsymbol{f}_1^*, \ldots, \boldsymbol{f}_n^*$. From the definition, $\mathcal{B}$ chooses vectors $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n, \boldsymbol{f}_1^*, \ldots, \boldsymbol{f}_n^*$ randomly, subject to the constraints that $\boldsymbol{f}_i \cdot \boldsymbol{f}_j^* \equiv 0 \,(\mathrm{mod}\, q)$ when $i \neq j$, and

$\boldsymbol{f}_i \cdot \boldsymbol{f}_i^* \equiv \psi \pmod q$ for all i from 1 to n, where ψ is a random element of $\mathbb{Z}_q$. Then, $\mathcal{B}$ implicitly sets:

$$\boldsymbol{b}_1 := \boldsymbol{f}_1 + a\boldsymbol{f}_{k+1}, \boldsymbol{b}_2 := \boldsymbol{f}_2 + a\boldsymbol{f}_{k+2}, \ldots, \boldsymbol{b}_k := \boldsymbol{f}_k + a\boldsymbol{f}_{2k},$$
$$\boldsymbol{b}_{k+1} := \boldsymbol{f}_{k+1}, \ldots, \boldsymbol{b}_n := \boldsymbol{f}_n.$$

$\mathcal{B}$ also sets the dual basis as:

$$\boldsymbol{b}_1^* := \boldsymbol{f}_1^*, \boldsymbol{b}_2^* := \boldsymbol{f}_2^*, \ldots, \boldsymbol{b}_k^* := \boldsymbol{f}_k^*,$$
$$\boldsymbol{b}_{k+1}^* := \boldsymbol{f}_{k+1}^* - a\boldsymbol{f}_1^*, \ldots, \boldsymbol{b}_{2k}^* := \boldsymbol{f}_{2k}^* - a\boldsymbol{f}_k^*,$$
$$\boldsymbol{b}_{2k+1}^* := \boldsymbol{f}_{2k+1}^*, \ldots, \boldsymbol{b}_n^* := \boldsymbol{f}_n^*.$$

We observe that under these definitions, $\boldsymbol{b}_i \cdot \boldsymbol{b}_j^* \equiv 0 \pmod q$ when $i \neq j$, and $\boldsymbol{b}_i \cdot \boldsymbol{b}_i^* \equiv \psi \pmod q$ for all i from 1 to n. We note that $\mathcal{B}$ can produce all of $g_1^{\boldsymbol{b}_1}, \ldots, g_1^{\boldsymbol{b}_n}$ (given g_1, g_1^a) as well as $g_2^{\boldsymbol{b}_1^*}, \ldots, g_2^{\boldsymbol{b}_k^*}$ and $g_2^{\boldsymbol{b}_{2k+1}^*}, \ldots, g_2^{\boldsymbol{b}_n^*}$ (given g_2). However, $\mathcal{B}$ cannot produce $g_2^{\boldsymbol{b}_{k+1}^*}, \ldots, g_2^{\boldsymbol{b}_{2k}^*}$ (these require knowledge of g_2^a). It is not difficult to check that $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ and $\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*$ are properly distributed.

Now $\mathcal{B}$ creates $U_1, \ldots, U_k$ by choosing random values $\mu_1', \mu_2' \in \mathbb{Z}_q$ and setting:

$$U_1 := g_2^{\mu_1' \boldsymbol{b}_1^* + \mu_2' \boldsymbol{f}_{k+1}^*} := g_2^{(\mu_1' + a\mu_2')\boldsymbol{b}_1^* + \mu_2' \boldsymbol{b}_{k+1}^*}.$$

In other words, $\mathcal{B}$ has implicitly set $\mu_1 := \mu_1' + a\mu_2'$ and $\mu_2 := \mu_2'$. We note that these values are uniformly random, and μ_2 is known to $\mathcal{B}$. $\mathcal{B}$ can then form $U_2, \ldots, U_k$ as:

$$U_2 := g_2^{\mu_1' \boldsymbol{b}_2^* + \mu_2' \boldsymbol{f}_{k+2}^*}, \ldots, U_k := g_2^{\mu_1' \boldsymbol{b}_k^* + \mu_2' \boldsymbol{f}_{2k}^*}.$$

$\mathcal{B}$ implicitly sets $\tau_1 := b, \tau_2 := c$ and computes:

$$T_1 := T^{\boldsymbol{f}_{k+1}} \cdot (g_1^b)^{\boldsymbol{f}_1}, \ldots, T_k := T^{\boldsymbol{f}_{2k}} \cdot (g_1^b)^{\boldsymbol{f}_k}.$$

If $T = g_1^{ab}$, then these are distributed as $V_1, \ldots, V_k$, since

$$T^{\boldsymbol{f}_{k+i}} \cdot (g_1^b)^{\boldsymbol{f}_i} = g_1^{\tau_1 \boldsymbol{b}_i}.$$

If $T = g_1^{ab+c}$, then these are distributed as $W_1, \ldots, W_k$, since

$$T^{\boldsymbol{f}_{k+i}} \cdot (g_1^b)^{\boldsymbol{f}_i} = g_1^{\tau_1 \boldsymbol{b}_i + \tau_2 \boldsymbol{b}_{k+i}}.$$

$\mathcal{B}$ then gives

$$D := (\mathbb{G}; g_2^{\boldsymbol{b}_1^*}, g_2^{\boldsymbol{b}_2^*}, \ldots, g_2^{\boldsymbol{b}_k^*}, g_2^{\boldsymbol{b}_{2k+1}^*}, \ldots, g_2^{\boldsymbol{b}_n^*}, g_1^{\boldsymbol{b}_1}, \ldots, g_1^{\boldsymbol{b}_n}, U_1, U_2, \ldots, U_k, \mu_2)$$

to $\mathcal{A}$, along with $T_1, \ldots, T_k$. $\mathcal{B}$ can then leverage $\mathcal{A}$'s advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DS1}}(\lambda)$ in distinguishing between the distributions $(V_1, \ldots, V_k)$ and $(W_1, \ldots, W_k)$ to achieve an advantage $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH1}}(\lambda)$ in distinguishing $T = g_1^{ab}$ from $T = g_1^{ab+c}$, hence violating the DDH assumption in G_1.

The dual of the Subspace assumption in G_1 is Subspace assumption in G_2 (denoted as DS2), which is identical to Definitions 6 with the roles of G_1 and G_2 reversed. Similarly, we can prove that the Subspace assumption holds in G_2 if the DDH assumption in G_2 holds.

4 Identity-Based Encryption

We now present our IBE construction along with our proof of its security under the SXDH assumption.

Construction. We begin with our IBE scheme:

- $\mathsf{Setup}(1^\lambda)$ This algorithm takes in the security parameter λ and generates a bilinear pairing $\mathbb{G} := (q, G_1, G_2, G_T, g_1, g_2, e)$ for sufficiently large prime order q. The algorithm samples random dual orthonormal bases, $(\mathbb{D}, \mathbb{D}^*) \xleftarrow{R} \mathsf{Dual}(\mathbb{Z}_q^4)$. Let $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_4$ denote the elements of $\mathbb{D}$ and $\boldsymbol{d}_1^*, \ldots, \boldsymbol{d}_4^*$ denote the elements of $\mathbb{D}^*$. It also picks $\alpha \xleftarrow{R} \mathbb{Z}_q$ and outputs the public parameters as
$$\mathsf{PP} := \{\mathbb{G}; e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*}, g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_2}\},$$
and the master key
$$\mathsf{MK} := \{\alpha, g_2^{\boldsymbol{d}_1^*}, g_2^{\boldsymbol{d}_2^*}\}.$$
- $\mathsf{KeyGen}(\mathsf{PP}, \mathsf{MK}, ID)$ This algorithm picks $r \xleftarrow{R} \mathbb{Z}_q$. The secret key is computed as
$$\mathsf{SK}_{ID} := g_2^{(\alpha + rID)\boldsymbol{d}_1^* - r\boldsymbol{d}_2^*}.$$
- $\mathsf{Enc}(\mathsf{PP}, ID, M)$ This algorithm picks $s \xleftarrow{R} \mathbb{Z}_q$ and forms the ciphertext as
$$\mathsf{CT}_{ID} := \left\{ C_0 := M \cdot (e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*})^s, \quad C_1 := g_1^{s\boldsymbol{d}_1 + sID\boldsymbol{d}_2} \right\}.$$
- $\mathsf{Dec}(\mathsf{PP}, \mathsf{SK}_{ID}, \mathsf{CT}_{ID})$ This algorithm computes the message as
$$M := C_0 / e(C_1, \mathsf{SK}_{ID}).$$

Correctness. Correctness is straight-forward:

$$\begin{aligned} C_0 / e(C_1, \mathsf{SK}_{ID}) &= M \cdot (e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*})^s / e(g_1^{s\boldsymbol{d}_1 + sID\boldsymbol{d}_2}, g_2^{(\alpha + rID)\boldsymbol{d}_1^* - r\boldsymbol{d}_2^*}) \\ &= M \cdot (e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*})^s / \big(e(g_1, g_2)^{\alpha s \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*} \\ &\qquad \cdot e(g_1, g_2)^{srID\boldsymbol{d}_1 \cdot \boldsymbol{d}_1^* - srID\boldsymbol{d}_2 \cdot \boldsymbol{d}_2^*} \big) \\ &= M \cdot (e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*})^s / e(g_1, g_2)^{\alpha s \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*} = M. \end{aligned}$$

Proof of Security. We prove the following theorem by showing a series of lemmas.

Theorem 1. *The IBE scheme is fully secure and anonymous under the Symmetric External Diffie-Hellman assumption. More precisely, for any adversary $\mathcal{A}$ against the IBE scheme, there exist probabilistic algorithms $\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_\nu$ whose running times are essentially the same as that of $\mathcal{A}$, such that*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IBE}}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{DDH1}}(\lambda) + \sum_{\kappa=1}^{\nu} \mathsf{Adv}_{\mathcal{B}_\kappa}^{\mathsf{DDH2}}(\lambda) + \frac{\nu}{q}$$

where ν is the maximum number of $\mathcal{A}$'s key queries.

We adopt the dual system encryption methodology by Waters [37] to prove the security of our IBE scheme. We use the concepts of *semi-functional ciphertexts* and *semi-functional keys* in our proof and provide algorithms that generate them. We note that these algorithms are only provided for definitional purposes, and are not part of the IBE system. In particular, they do not need to be efficiently computable from the public parameters and the master key.

KeyGenSF. The algorithm picks random values $r, t_3, t_4 \in \mathbb{Z}_q$ and forms a semi-functional secret key as

$$\mathsf{SK}_{ID}^{(SF)} := g_2^{(\alpha + rID)\boldsymbol{d}_1^* - r\boldsymbol{d}_2^* + t_3\boldsymbol{d}_3^* + t_4\boldsymbol{d}_4^*}.$$

EncryptSF. The algorithm picks random values random values $s, z_3, z_4 \in \mathbb{Z}_q$ and forms a semi-functional ciphertext as

$$\mathsf{CT}_{ID}^{(SF)} := \left\{ C_0 := M \cdot (e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*})^s, \quad C_1 := g_1^{s\boldsymbol{d}_1 + sID\boldsymbol{d}_2 + z_3\boldsymbol{d}_3 + z_4\boldsymbol{d}_4} \right\}.$$

We observe that if one applies the decryption procedure with a semi-functional key and a normal ciphertext, decryption will succeed because $\boldsymbol{d}_3^*, \boldsymbol{d}_4^*$ are orthogonal to all of the vectors in exponent of C_1, and hence have no effect on decryption. Similarly, decryption of a semi-functional ciphertext by a normal key will also succeed because $\boldsymbol{d}_3, \boldsymbol{d}_4$ are orthogonal to all of the vectors in the exponent of the key. When both the ciphertext and key are semi-functional, the result of $e(C_1, \mathsf{SK}_{ID})$ will have an additional term, namely

$$e(g_1, g_2)^{t_3 z_3 \boldsymbol{d}_3^* \cdot \boldsymbol{d}_3 + t_4 z_4 \boldsymbol{d}_4^* \cdot \boldsymbol{d}_4} = e(g_1, g_2)^{(t_3 z_3 + t_4 z_4)\psi}.$$

Decryption will then fail unless $t_3 z_3 + t_4 z_4 \equiv 0 \bmod q$. If this modular equation holds, we say that the key and ciphertext pair is *nominally semi-functional*.

For a probabilistic polynomial-time adversary $\mathcal{A}$ which makes ν key queries $ID_1, \ldots, ID_\nu$, our proof of security consists of the following sequence of games between $\mathcal{A}$ and a challenger $\mathcal{B}$.

- Game_{Real}: is the real security game.
- Game_0: is the same as Game_{Real} except that the challenge ciphertext is semi-functional.
- Game_κ: for κ from 1 to ν, Game_κ is the same as Game_0 except that the first κ keys are semi-functional and the remaining keys are normal.
- Game_{Final}: is the same as Game_ν, except that the challenge ciphertext is a semi-functional encryption of a random message in G_T and under a random identity in $\mathbb{Z}_q$. We denote the challenge ciphertext in Game_{Final} as $\mathsf{CT}_{ID_R}^{(R)}$.

We prove following lemmas to show the above games are indistinguishable by following an analogous strategy of [25]. Our main arguments are computational indistinguishability (guaranteed by the Subspace assumptions, which are implied by the SXDH assumption) and statistical indistinguishability. The advantage gap between Game_{Real} and Game_0 is bounded by the advantage of the Subspace assumption in G_1. Additionally, we require a statistical indistinguishability argument to show that the distribution of the challenge ciphertext remains the

same from the adversary's view. For κ from 1 to ν, the advantage gap between $\mathsf{Game}_{\kappa-1}$ and Game_{κ} is bounded by the advantage of Subspace assumption in G_2. Similarly, we require a statistical indistinguishability argument to show that the distribution of the the κ-th semi-functional key remains the same from the adversary's view. Finally, we statistically transform Game_{ν} to Game_{Final} in one step, i.e., we show the joint distributions of

$$(\mathsf{PP}, \mathsf{CT}^{(SF)}_{ID^*_\beta}, \{\mathsf{SK}^{(SF)}_{ID_\ell}\}_{\ell=1,\ldots,\nu}) \text{ and } (\mathsf{PP}, \mathsf{CT}^{(R)}_{ID_R}, \{\mathsf{SK}^{(SF)}_{ID_\ell}\}_{\ell=1,\ldots,\nu})$$

are equivalent for the adversary's view.

We let $\mathsf{Adv}^{\mathsf{Game}_{Real}}_{\mathcal{A}}$ denote an adversary $\mathcal{A}$'s advantage in the real game.

Lemma 3. *Suppose that there exists an adversary $\mathcal{A}$ where $|\mathsf{Adv}^{\mathsf{Game}_{Real}}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda)| = \epsilon$. Then there exists an algorithm $\mathcal{B}_0$ such that $\mathsf{Adv}^{\mathsf{DS1}}_{\mathcal{B}_0}(\lambda) = \epsilon$, with $k = 2$ and $n = 4$.*

Proof. $\mathcal{B}_0$ is given

$$D := (\mathbb{G}; g_2^{\boldsymbol{b}^*_1}, g_2^{\boldsymbol{b}^*_2}, g_1^{\boldsymbol{b}_1}, \ldots, g_1^{\boldsymbol{b}_4}, U_1, U_2, \mu_2).$$

along with T_1, T_2. We require that $\mathcal{B}_0$ decides whether T_1, T_2 are distributed as $g_1^{\tau_1 \boldsymbol{b}_1}, g_1^{\tau_1 \boldsymbol{b}_2}$ or $g_1^{\tau_1 \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_3}, g_1^{\tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{b}_4}$.

$\mathcal{B}_0$ simulates Game_{Real} or Game_0 with $\mathcal{A}$, depending on the distribution of T_1, T_2. To compute the public parameters and master secret key, $\mathcal{B}_0$ first chooses a random invertible matrix $A \in \mathbb{Z}_q^{2\times 2}$ (A is invertible with overwhelming probability if it is uniformly picked). We implicitly set dual orthonormal bases $\mathbb{D}, \mathbb{D}^*$ to:

$$\begin{aligned} \boldsymbol{d}_1 &:= \boldsymbol{b}_1, \quad \boldsymbol{d}_2 := \boldsymbol{b}_2, \quad (\boldsymbol{d}_3, \boldsymbol{d}_4) := (\boldsymbol{b}_3, \boldsymbol{b}_4)A, \\ \boldsymbol{d}^*_1 &:= \boldsymbol{b}^*_1, \quad \boldsymbol{d}^*_2 := \boldsymbol{b}^*_2, \quad (\boldsymbol{d}^*_3, \boldsymbol{d}^*_4) := (\boldsymbol{b}^*_3, \boldsymbol{b}^*_4)(A^{-1})^t. \end{aligned}$$

We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed, and reveal no information about A. Moreover, $\mathcal{B}$ cannot generate $g_2^{\boldsymbol{d}^*_3}, g_2^{\boldsymbol{d}^*_4}$, but these will not be needed for creating normal keys. $\mathcal{B}_0$ chooses random value $\alpha \in \mathbb{Z}_q$ and computes $e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}^*_1}$. It then gives $\mathcal{A}$ the public parameters

$$\mathsf{PP} := \{\mathbb{G}; e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}^*_1}, g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_2}\}.$$

The master key

$$\mathsf{MK} := \{\alpha, g_2^{\boldsymbol{d}^*_1}, g_2^{\boldsymbol{d}^*_2}\}$$

is known to $\mathcal{B}_0$, which allows $\mathcal{B}_0$ to respond to all of $\mathcal{A}$'s key queries by calling the normal key generation algorithm.

$\mathcal{A}$ sends $\mathcal{B}_0$ two pairs (M_0, ID^*_0) and (M_1, ID^*_1). $\mathcal{B}_0$ chooses a random bit $\beta \in \{0, 1\}$ and encrypts M_β under ID^*_β as follows:

$$C_0 := M_\beta \Big(e(T_1, g_2^{\boldsymbol{b}^*_1}) \Big)^{\alpha} = M_\beta \Big(e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \boldsymbol{d}^*_1} \Big)^{s}, \quad C_1 := T_1 (T_2)^{ID^*_\beta},$$

where $\mathcal{B}_0$ has implicitly set $s := \tau_1$. It gives the ciphertext $\mathsf{CT}_{ID^*_\beta}$ to $\mathcal{A}$.

Now, if T_1, T_2 are equal to $g_1^{\tau_1 \boldsymbol{b}_1}, g_1^{\tau_1 \boldsymbol{b}_2}$, then this is a properly distributed normal encryption of M_β. In this case, $\mathcal{B}_0$ has properly simulated Game_{Real}. If T_1, T_2 are equal to $g_1^{\tau_1 \boldsymbol{b}_1 + \tau_2 \boldsymbol{b}_3}, g_1^{\tau_1 \boldsymbol{b}_2 + \tau_2 \boldsymbol{b}_4}$ instead, then the ciphertext element C_1 has an additional term of

$$\tau_2 \boldsymbol{b}_3 + ID^*_\beta \tau_2 \boldsymbol{b}_4$$

in its exponent. The coefficients here in the basis $\boldsymbol{b}_3, \boldsymbol{b}_4$ form the vector $\tau_2, ID^*_\beta \tau_2$. To compute the coefficients in the basis $\boldsymbol{d}_3, \boldsymbol{d}_4$, we multiply the matrix A^{-1} by the transpose of this vector, obtaining $\tau_2 A^{-1}(1, ID^*_\beta)^t$. Since A is random (everything else given to $\mathcal{A}$ has been distributed independently of A), these coefficients are uniformly random from Lemma 1. Therefore, in this case, $\mathcal{B}_0$ has properly simulated Game_0. This allows $\mathcal{B}_0$ to leverage $\mathcal{A}$'s advantage ϵ between Game_{Real} and Game_0 to achieve an advantage ϵ against the Subspace assumption in G_1, namely $\mathsf{Adv}^{\mathsf{DS1}}_{\mathcal{B}_0}(\lambda) = \epsilon$. □

Lemma 4. *Suppose that there exists an adversary $\mathcal{A}$ where $|\mathsf{Adv}^{\mathsf{Game}_{\kappa-1}}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Game}_\kappa}_{\mathcal{A}}(\lambda)| = \epsilon$. Then there exists an algorithm $\mathcal{B}_\kappa$ such that $\mathsf{Adv}^{\mathsf{DS2}}_{\mathcal{B}_\kappa}(\lambda) = \epsilon - 1/q$, with $k = 2$ and $n = 4$.*

Proof. $\mathcal{B}_\kappa$ is given

$$D := (\mathbb{G}; g_1^{\boldsymbol{b}_1}, g_1^{\boldsymbol{b}_2}, g_2^{\boldsymbol{b}^*_1}, \ldots, g_2^{\boldsymbol{b}^*_4}, U_1, U_2, \mu_2)$$

along with T_1, T_2. We require that $\mathcal{B}_\kappa$ decides whether T_1, T_2 are distributed as $g_2^{\tau_1 \boldsymbol{b}^*_1}, g_2^{\tau_1 \boldsymbol{b}^*_2}$ or $g_2^{\tau_1 \boldsymbol{b}^*_1 + \tau_2 \boldsymbol{b}^*_3}, g_2^{\tau_1 \boldsymbol{b}^*_2 + \tau_2 \boldsymbol{b}^*_4}$.

$\mathcal{B}_\kappa$ simulates Game_κ or $\mathsf{Game}_{\kappa-1}$ with $\mathcal{A}$, depending on the distribution of T_1, T_2. To compute the public parameters and master secret key, $\mathcal{B}_\kappa$ chooses a random matrix $A \in \mathbb{Z}_q^{2\times 2}$ (with all but negligible probability, A is invertible). We then implicitly set dual orthonormal bases $\mathbb{D}, \mathbb{D}^*$ to:

$$\begin{aligned} \boldsymbol{d}_1 &:= \boldsymbol{b}_1, \quad \boldsymbol{d}_2 := \boldsymbol{b}_2, \quad (\boldsymbol{d}_3, \boldsymbol{d}_4) := (\boldsymbol{b}_3, \boldsymbol{b}_4)A, \\ \boldsymbol{d}^*_1 &:= \boldsymbol{b}^*_1, \quad \boldsymbol{d}^*_2 := \boldsymbol{b}^*_2, \quad (\boldsymbol{d}^*_3, \boldsymbol{d}^*_4) := (\boldsymbol{b}^*_3, \boldsymbol{b}^*_4)(A^{-1})^t. \end{aligned}$$

We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed, and reveal no information about A. $\mathcal{B}_\kappa$ chooses random value $\alpha \in \mathbb{Z}_q$ and compute $e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}^*_1}$. $\mathcal{B}$ can gives $\mathcal{A}$ the public parameters

$$\mathsf{PP} := \{\mathbb{G}; e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}^*_1}, g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_2}\}.$$

The master key

$$\mathsf{MK} := \{\alpha, g_2^{\boldsymbol{d}^*_1}, g_2^{\boldsymbol{d}^*_2}\}$$

is known to $\mathcal{B}_\kappa$, which allows $\mathcal{B}_\kappa$ to respond to all of $\mathcal{A}$'s key queries by calling the normal key generation algorithm. Since $\mathcal{B}_\kappa$ also knows $g_2^{\boldsymbol{d}^*_3}$ and $g_2^{\boldsymbol{d}^*_4}$, it can easily produce semi-functional keys. To answer the first $\kappa - 1$ key queries that $\mathcal{A}$

makes, $\mathcal{B}_\kappa$ runs the semi-functional key generation algorithm to produce semi-functional keys and gives these to $\mathcal{A}$. To answer the κ-th key query for ID_κ, $\mathcal{B}_\kappa$ responds with:

$$\mathsf{SK}_{ID_\kappa} := (g_2^{\boldsymbol{b}_1^*})^\alpha T_1^{ID_\kappa} (T_2)^{-1}.$$

This implicitly sets $r := \tau_1$. If T_1, T_2 are equal to $g_2^{\tau_1 \boldsymbol{b}_1^*}, g_2^{\tau_1 \boldsymbol{b}_2^*}$, then this is a properly distributed normal key. If T_1, T_2 are equal to $g_2^{\tau_1 \boldsymbol{b}_1^* + \tau_2 \boldsymbol{b}_3^*}, g_2^{\tau_1 \boldsymbol{b}_2^* + \tau_2 \boldsymbol{b}_4^*}$, then this is a semi-functional key, whose exponent vector includes

$$ID_\kappa \tau_2 \boldsymbol{b}_3^* - \tau_2 \boldsymbol{b}_4^* \tag{1}$$

as its component in the span of $\boldsymbol{b}_3^*, \boldsymbol{b}_4^*$. To respond to the remaining key queries, $\mathcal{B}_\kappa$ simply runs the normal key generation algorithm.

At some point, $\mathcal{A}$ sends $\mathcal{B}_\kappa$ two pairs (M_0, ID_0^*) and (M_1, ID_1^*). $\mathcal{B}_\kappa$ chooses a random bit $\beta \in \{0,1\}$ and encrypts M_β under ID_β^* as follows:

$$C_0 := M_\beta \Big(e(U_1, g_2^{\boldsymbol{b}_1^*}) \Big)^\alpha = M_\beta \Big(e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \boldsymbol{d}_1^*} \Big)^s, \quad C_1 := U_1 (U_2)^{ID_\beta^*},$$

where $\mathcal{B}_\kappa$ has implicitly set $s := \mu_1$. The "semi-functional part" of the exponent vector here is:

$$\mu_2 \boldsymbol{b}_3 + ID^* \mu_2 \boldsymbol{b}_4. \tag{2}$$

We observe that if $ID_\beta^* = ID_\kappa$ (which is not allowed), then vectors 1 and 2 would be orthogonal, resulting in a nominally semi-functional ciphertext and key pair. It gives the ciphertext $\mathsf{CT}_{ID_\beta^*}$ to $\mathcal{A}$.

We now argue that since $ID_\beta^* \neq ID_\kappa$, in $\mathcal{A}$'s view the vectors 1 and 2 are distributed as random vectors in the spans of $\boldsymbol{d}_3^*, \boldsymbol{d}_4^*$ and $\boldsymbol{d}_3, \boldsymbol{d}_4$ respectively. To see this, we take the coefficients of vectors 1 and 2 in terms of the bases $\boldsymbol{b}_3^*, \boldsymbol{b}_4^*$ and $\boldsymbol{b}_3, \boldsymbol{b}_4$ respectively and translate them into coefficients in terms of the bases $\boldsymbol{d}_3^*, \boldsymbol{d}_4^*$ and $\boldsymbol{d}_3, \boldsymbol{d}_4$. Using the change of basis matrix A, we obtain the new coefficients (in vector form) as:

$$\tau_2 A^t (ID_\kappa, -1)^t,\ \mu_2 A^{-1} (1, ID_\beta^*).$$

Since the distribution of everything given to A except for the κ-th key and the challenge ciphertext is independent of the random matrix A and $ID_\beta^* \neq ID_\kappa$, we can conclude that these coefficients are uniformly random (except for $1/q$ probability) from Lemma 1. Thus, $\mathcal{B}_\kappa$ has properly simulated Game_κ in this case.

In summary, $\mathcal{B}_\kappa$ has properly simulated either $\mathsf{Game}_{\kappa-1}$ or Game_κ for $\mathcal{A}$, depending on the distribution of T_1, T_2. It can therefore leverage $\mathcal{A}$'s advantage ϵ between these games to obtain an advantage $\epsilon - 1/q$ against the Subspace assumption in G_2, namely $\mathsf{Adv}^{\mathsf{DS2}}_{\mathcal{B}_\kappa}(\lambda) = \epsilon - 1/q$. □

Lemma 5. *For any adversary* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{Game}_\nu}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Game}_{Final}}_{\mathcal{A}}(\lambda)$.

Proof. To prove this lemma, we show the joint distributions of

$$(\mathsf{PP}, \mathsf{CT}^{(SF)}_{ID^*_\beta}, \{\mathsf{SK}^{(SF)}_{ID_\ell}\}_{\ell=1,\ldots,\nu})$$

in Game_ν and that of

$$(\mathsf{PP}, \mathsf{CT}^{(R)}_{ID_R}, \{\mathsf{SK}^{(SF)}_{ID_\ell}\}_{\ell=1,\ldots,\nu})$$

in Game_{Final} are equivalent for the adversary's view, where $\mathsf{CT}^{(R)}_{ID_R}$ is a semi-functional encryption of a random message in G_T and under a random identity in $\mathbb{Z}_q$.

For this purpose, we pick $A := (\xi_{i,j}) \xleftarrow{R} \mathbb{Z}_q^{2\times 2}$ and define new dual orthonormal bases $\mathbb{F} := (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_4)$, and $\mathbb{F}^* := (\boldsymbol{f}^*_1, \ldots, \boldsymbol{f}^*_4)$ as follows:

$$\begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \boldsymbol{f}_3 \\ \boldsymbol{f}_4 \end{pmatrix} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \xi_{1,1} & \xi_{1,2} & 1 & 0 \\ \xi_{2,1} & \xi_{2,2} & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \\ \boldsymbol{d}_3 \\ \boldsymbol{d}_4 \end{pmatrix}, \quad \begin{pmatrix} \boldsymbol{f}^*_1 \\ \boldsymbol{f}^*_2 \\ \boldsymbol{f}^*_3 \\ \boldsymbol{f}^*_4 \end{pmatrix} := \begin{pmatrix} 1 & 0 & -\xi_{1,1} & -\xi_{2,1} \\ 0 & 1 & -\xi_{1,2} & -\xi_{2,2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}^*_1 \\ \boldsymbol{d}^*_2 \\ \boldsymbol{d}^*_3 \\ \boldsymbol{d}^*_4 \end{pmatrix}.$$

It is easy to verify that $\mathbb{F}$ and $\mathbb{F}^*$ are also dual orthonormal, and are distributed the same as $\mathbb{D}$ and $\mathbb{D}^*$.

Then the public parameters, challenge ciphertext, and queried secret keys $(\mathsf{PP}, \mathsf{CT}^{(SF)}_{ID^*_\beta}, \{\mathsf{SK}^{(SF)}_{ID_\ell}\}_{\ell=1,\ldots,\nu})$ in Game_ν are expressed over bases $\mathbb{D}$ and $\mathbb{D}^*$ as

$$\begin{aligned}
&\mathsf{PP} := \{\mathbb{G}; e(g_1,g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}^*_1}, g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_2}\}, \\
&\mathsf{CT}^{(SF)}_{ID^*_\beta} := \left\{ C_0 := M \cdot (e(g_1,g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}^*_1})^s, \quad C_1 := g_1^{s\boldsymbol{d}_1 + sID^*_\beta \boldsymbol{d}_2 + z_3 \boldsymbol{d}_3 + z_4 \boldsymbol{d}_4} \right\}, \\
&\left\{ \mathsf{SK}^{(SF)}_{ID_\ell} := g_2^{(\alpha + r_\ell ID_\ell)\boldsymbol{d}^*_1 - r_\ell \boldsymbol{d}^*_2 + t_{\ell,3}\boldsymbol{d}^*_3 + t_{\ell,4}\boldsymbol{d}^*_4} \right\}_{\ell=1,\ldots,\nu}.
\end{aligned}$$

Then we can express them over bases $\mathbb{F}$ and $\mathbb{F}^*$ as

$$\begin{aligned}
&\mathsf{PP} := \{\mathbb{G}; e(g_1,g_2)^{\alpha \boldsymbol{f}_1 \cdot \boldsymbol{f}^*_1}, g_1^{\boldsymbol{f}_1}, g_1^{\boldsymbol{f}_2}\}, \\
&\mathsf{CT}^{(SF)}_{ID^*_\beta} := \left\{ C_0 := M \cdot (e(g_1,g_2)^{\alpha \boldsymbol{f}_1 \cdot \boldsymbol{f}^*_1})^s, \quad C_1 := g_1^{s'\boldsymbol{f}_1 + s''\boldsymbol{f}_2 + z_3 \boldsymbol{f}_3 + z_4 \boldsymbol{f}_4} \right\}, \\
&\left\{ \mathsf{SK}^{(SF)}_{ID_\ell} := g_2^{(\alpha + r_\ell ID_\ell)\boldsymbol{f}^*_1 - r_\ell \boldsymbol{f}^*_2 + t'_{\ell,3}\boldsymbol{f}^*_3 + t'_{\ell,4}\boldsymbol{f}^*_4} \right\}_{\ell=1,\ldots,\nu},
\end{aligned}$$

where

$$\begin{aligned}
&s' := s - z_3\xi_{1,1} - z_4\xi_{2,1}, \quad s'' := sID^*_\beta - z_3\xi_{1,2} - z_4\xi_{2,2}, \\
&\left\{ \begin{array}{l} t'_{\ell,3} := t_{\ell,3} + \xi_{1,1}(\alpha + r_\ell ID_\ell) - r_\ell \xi_{1,2}, \\ t'_{\ell,4} := t_{\ell,4} + \xi_{2,1}(\alpha + r_\ell ID_\ell) - r_\ell \xi_{2,2} \end{array} \right\}_{\ell=1,\ldots,\nu},
\end{aligned}$$

which are all uniformly distributed since $\xi_{1,1}, \xi_{1,2}, \xi_{2,1}, \xi_{2,2}, t_{1,3}, t_{1,4}, \ldots, t_{\nu,3}, t_{\nu,4}$ are all uniformly picked from $\mathbb{Z}_q$.

In other words, the coefficients (s, sID^*_β) of $\boldsymbol{d}_1, \boldsymbol{d}_2$ in the C_1 term of the challenge ciphertext is changed to random coefficients $(s', s'') \in \mathbb{Z}_q \times \mathbb{Z}_q$ of $\boldsymbol{f}_1, \boldsymbol{f}_2$, thus the challenge ciphertext can be viewed as a semi-functional encryption of a random message in G_T and under a random identity in $\mathbb{Z}_q$. Moreover, all coefficients $\{(t'_{\ell,3}, t'_{\ell,4})\}_{\ell=1,\ldots,\nu}$ of $\boldsymbol{f}_1, \boldsymbol{f}_2$ in the $\{\mathsf{SK}^{(SF)}_{ID_\ell}\}_{\ell=1,\ldots,\nu}$ are all uniformly distributed since $\{(t_{\ell,3}, t_{\ell,4})\}_{\ell=1,\ldots,\nu}$ of $\boldsymbol{d}^*_3, \boldsymbol{d}^*_4$ are all independent random values. Thus

$$(\mathsf{PP}, \mathsf{CT}^{(SF)}_{ID^*_\beta}, \{\mathsf{SK}^{(SF)}_{ID_\ell}\}_{\ell=1,\ldots,\nu})$$

expressed over bases $\mathbb{F}$ and $\mathbb{F}^*$ is properly distributed as

$$(\mathsf{PP}, \mathsf{CT}^{(R)}_{ID_R}, \{\mathsf{SK}^{(SF)}_{ID_\ell}\}_{\ell=1,\ldots,\nu})$$

in Game_{Final}.

In the adversary's view, both $(\mathbb{D}, \mathbb{D}^*)$ and $(\mathbb{F}, \mathbb{F}^*)$ are consistent with the same public key. Therefore, the challenge ciphertext and queried secret keys above can be expressed as keys and ciphertext in two ways, in Game_ν over bases $(\mathbb{D}, \mathbb{D}^*)$ and in Game_{Final} over bases $(\mathbb{F}, \mathbb{F}^*)$. Thus, Game_ν and Game_{Final} are statistically indistinguishable. □

Lemma 6. *For any adversary* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{Game}_{Final}}_{\mathcal{A}}(\lambda) = 0$.

Proof. The value of β is independent from the adversary's view in Game_{Final}. Hence, $\mathsf{Adv}^{\mathsf{Game}_{Final}}_{\mathcal{A}}(\lambda) = 0$. □

In Game_{Final}, the challenge ciphertext is a semi-functional encryption of a random message in G_T and under a random identity in $\mathbb{Z}_q$, independent of the two messages and the challenge identities provided by $\mathcal{A}$. Thus, our IBE scheme is anonymous.

5 A Signature Scheme

In this section, we present the signature scheme derived from the preceding IBE scheme via Naor's transform. The security of the signature scheme follows from the full security of our IBE scheme.

– $\mathsf{KeyGen}(1^\lambda)$ This algorithm takes in the security parameter λ and generates a bilinear pairing $\mathbb{G} := (q, G_1, G_2, G_T, g_1, g_2, e)$ for sufficiently large prime order q. The algorithm samples random dual orthonormal bases, $(\mathbb{D}, \mathbb{D}^*) \xleftarrow{R} \mathsf{Dual}(\mathbb{Z}^4_q)$. Let $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_4$ denote the elements of $\mathbb{D}$ and $\boldsymbol{d}^*_1, \ldots, \boldsymbol{d}^*_4$ denote the elements of $\mathbb{D}^*$. It outputs the public key as

$$\mathsf{PK} = \{\mathbb{G}; e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}^*_1}, g_1^{\boldsymbol{d}_1}, g_1^{\boldsymbol{d}_2}\},$$

and the signing key

$$\mathsf{SK} = \{\alpha, g_2^{\boldsymbol{d}^*_1}, g_2^{\boldsymbol{d}^*_2}\}.$$

– $\mathsf{Sign}(\mathsf{PK}, \mathsf{SK}, M)$ This algorithm picks $r \xleftarrow{R} \mathbb{Z}_q$ and computes the signature as

$$\sigma = g_2^{(\alpha + rM)\boldsymbol{d}_1^* - r\boldsymbol{d}_2^*}.$$

– $\mathsf{Verify}(\mathsf{PK}, \sigma, M)$ This algorithm verifies a signature σ by testing whether $e(g_1^{\boldsymbol{d}_1 + M\boldsymbol{d}_2}, \sigma) = e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*}$.[3] If the equality holds the signature is declared `valid`; otherwise it is declared `invalid`.

Acknowledgments. We thank the referees for helpful feedback. Research of the authors is supported in part by the National Research Foundation of Singapore under Research Grant NRF-CRP2-2007-03. Hoeteck Wee's work is also supported by NSF CAREER Award CNS-1237429.

References

[1] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. J. Cryptology 21(3), 350–391 (2008)

[2] Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)

[3] Ateniese, G., Kirsch, J., Blanton, M.: Secret handshakes with dynamic and fuzzy matching. In: NDSS (2007)

[4] Ballard, L., Green, M., de Medeiros, B., Monrose, F.: Correlation-resistant storage via keyword-searchable encryption. IACR Cryptology ePrint Archive, Report 2005/417 (2005)

[5] Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management—part 1: General (revised). NIST Special Pub., 800-57 (2007)

[6] 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)

[7] Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[8] Boneh, D., Boyen, X.: Secure Identity Based Encryption Without Random Oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)

[9] Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)

[10] Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[11] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptology 17(4), 297–319 (2004)

[3] Directly applying Naor's transform yields a verification algorithm that works as follows: pick $s \leftarrow \mathbb{Z}_q$, and test whether $e(g_1^{(\boldsymbol{d}_1 + M\boldsymbol{d}_2)s}, \sigma) = (e(g_1, g_2)^{\alpha \boldsymbol{d}_1 \cdot \boldsymbol{d}_1^*})^s$. With overwhelming probability over s, this agrees with the verification algorithm as written.

[12] Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: FOCS, pp. 501–510 (2010)
[13] Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Des. Codes Cryptography 37(1), 133–141 (2005)
[14] Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
[15] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
[16] Cocks, C.: An identity based encryption scheme based on quadratic residues. In: IMA Int. Conf., pp. 360–363 (2001)
[17] Ducas, L.: Anonymity from Asymmetry: New Constructions for Anonymous HIBE. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 148–164. Springer, Heidelberg (2010)
[18] Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptology 23(2), 224–280 (2010)
[19] Freeman, D.M.: Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
[20] Galbraith, S.D., Rotger, V.: Easy decision Diffie-Hellman groups. IACR Cryptology ePrint Archive, Report 2004/070 (2004)
[21] Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
[22] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)
[23] Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
[24] 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)
[25] Lewko, A.: Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
[26] Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
[27] Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
[28] Miyaji, A., Nakabayashi, M., Takano, S.: Characterization of Elliptic Curve Traces under FR-Reduction. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 90–108. Springer, Heidelberg (2001)
[29] Okamoto, T., Takashima, K.: Homomorphic Encryption and Signatures from Vector Decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 57–74. Springer, Heidelberg (2008)

[30] Okamoto, T., Takashima, K.: Hierarchical Predicate Encryption for Inner-Products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
[31] Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
[32] Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. Cryptology ePrint Archive, Report 2010/563 (2010)
[33] Ramanna, S.C., Chatterjee, S., Sarkar, P.: Variants of waters' dual-system primitives using asymmetric pairings. IACR Cryptology ePrint Archive, Report 2012/024 (2012)
[34] 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)
[35] Verheul, E.R.: Evidence that XTR is more secure than Supersingular Elliptic Curve cryptosystems. Journal of Cryptology 17(4), 277–296 (2004)
[36] Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
[37] Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)

A Estimated Bit Sizes of Group Elements for Bilinear Group Generators

The ordinary elliptic curves that give the best performance while providing discrete log security comparable to three commonly proposed levels of AES security are as follows. The group sizes follow the 2007 NIST recommendations [5], descriptions of the elliptic curves are in [18].

80-bit security: A 170-bit MNT curve [28] with embedding degree $k = 6$.
128-bit security: A 256-bit Barreto-Naehrig curve [6] with $k = 12$.
256-bit security: A 640-bit Brezing-Weng curve [13] with $k = 24$.

Note that a symmetric pairing only exists on supersingular elliptic curves. The restriction to supersingular elliptic curves means that at high security levels the group G_1 will be much larger than the group G_1 on an equivalent ordinary curve.

Table 2. Estimated bit sizes of elements in bilinear groups

Pairings	80-bit AES			128-bit AES			256-bit AES		
	G_1	G_2	G_T	G_1	G_2	G_T	G_1	G_2	G_T
Asymmetric	170	340	1020	256	512	3072	640	2560	15360
Symmetric	176	176	1056	512	512	3072	2560	2560	15360

Core Based Architecture to Speed Up Optimal Ate Pairing on FPGA Platform

Santosh Ghosh[1,⋆], Ingrid Verbauwhede[1], and Dipanwita Roychowdhury[2]

[1] KU Leuven and IBBT
Dept. Electrical Engineering-ESAT/SCD/COSIC
Kasteelpark Arenberg 10, 3001 Heverlee-Leuven, Belgium
firstname.lastname@esat.kuleuven.be
[2] Indian Institute of Technology Kharagpur
Dept. Computer Science and Engineering
Kharagpur, WB, 721302, India
drc@cse.iitkgp.ernet.in

Abstract. This paper presents an efficient implementation of optimal-ate pairing over BN curves. It exploits the highly optimized IP cores available in modern FPGAs to speed up pairing computation. The pipelined datapaths for $\mathbb{F}_p$-operations and suitable memory cores help to reduce the overall clock cycle count more than 50%. The final design, on a Virtex-6 FPGA, computes an optimal-ate pairing having 126-bit security in 0.375 ms which is a 32% speedup from state of the art result.

Keywords: Pairing, BN curves, prime fields, FPGA, Karatsuba, Montgomery, Pipeline, IP core.

1 Introduction

The use of pairings in constructive cryptographic applications are running in their second decade. During this period it has gained a lot of importance because it enables practical realization of numerous protocols. At the same time it is also important to implement pairings for using those protocols in practice. Different alternatives have been derived from the original proposal of Tate pairing for its efficient computation. Optimal-ate pairing [24] is to date the most efficient one computed over elliptic curves (E) defined over a large prime field ($\mathbb{F}_p$). On the other hand, several algebraic curves have been discovered for providing better pairing computation technique as well as for achieving better security. We call them pairing-friendly curves. Barreto-Naehrig curve [3] is the most popular pairing-friendly curve in current days. It is well studied that the optimal-ate pairing on BN curves is one of the best choices of selecting pairings in practice [2].

This paper aims to design an efficient hardware architecture for computing optimal-ate pairing on BN curves. The architecture exploits highly optimized

⋆ The work started when the first author was in IIT Kharagpur, India. Now he is a Postdoctoral Fellow at KU Leuven funded by the IAP Programme P6/26 BCRYPT of Belgian Science Policy (Belspo).

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 141–159, 2013.

IP cores available for modern FPGAs. The in-built independencies of underlying operations of the pairing computation are fully utilized in order to run an optimized pipeline datapath with reduced number of stall cycles. The memory architecture based on IP cores are efficiently used for generating pipeline operands and storing intermediate results which reduces the use of registers in the design too. The pipelined datapath together with said memory architecture helps to reduce clock cycle count of the pairing computation. A dedicated inversion unit is also incorporated into the design for reducing further cycle count. In total, the final design achieves 32% speedup from the existing premier design [6] for computing optimal-ate pairing.

We start with a brief overview of optimal-ate pairing and its computation procedure over BN curves in § 2. The IP cores that are used in this design are introduced in § 3. The design of the most important underlying $\mathbb{F}_p$-arithmetic block is described in § 4 followed by the description of overall core-based architecture in § 5. The scheduling of operations in order to compute different steps of the pairing algorithm is given in § 6. In § 7, we provide the performance study of the new design with respect to existing results. The paper is concluded in § 8.

2 Optimal-Ate Pairing

Optimal-ate pairing [24] is a non-degenerative bilinear map from $\mathbb{G}_2 \times \mathbb{G}_1$ to $\mathbb{G}_T$ where $\mathbb{G}_2$ and $\mathbb{G}_1$ to be specific subgroups of $E(\mathbb{F}_{p^k})$, and $\mathbb{G}_T$ to be a subgroup of $\mathbb{F}^*_{p^k}$. Let n is a large odd prime dividing $\#E(\mathbb{F}_p)$, and k corresponds to the embedding degree that is the smallest positive integer such that $n|(p^k - 1)$. This paper focuses on the optimal-ate pairing on Barreto-Naehrig curve [3], which is well-suited for 128-bit security level and has degree six twist.

A BN curve is an elliptic curve defined over $\mathbb{F}_p$ by following equation.

$$E : y^2 = x^3 + b,$$

where $b \neq 0$ such that $\#E = n$, and $k = 12$. The BN parameters are defined by a suitable $z \in \mathbb{Z}$ such that $p = 36z^4 + 36z^3 + 24z^2 + 6z + 1$ and $n = 36z^4 + 36z^3 + 18z^2 + 6z + 1$ are prime. This paper focuss on optimal-ate pairing with $r = 6z + 2$ defined as [2]:

$$a_{opt} : E(\mathbb{F}_{p^{12}}) \cap Ker(\pi_p - p) \times E(\mathbb{F}_p[n]) \rightarrow \mathbb{F}^*_{p^{12}}/(\mathbb{F}^*_{p^{12}})^n$$
$$(Q, P) \mapsto \left(f_{(r,Q)}(P) \cdot g_{(rQ,\pi_p(Q))}(P) \cdot g_{(rQ+\pi_p(Q),-\pi^2_p(Q))}(P)\right)^{(p^{12}-1)/n}$$

where π_p is the Frobenius map on the curve ($\pi_p(x, y) = (x^p, y^p)$), and $g_{(Q_1,Q_2)}$ is the line through Q_1 and Q_2.

2.1 Computation Procedure

Algorithm 1 computes above optimal-ate pairing. We choose BN curve $E : y^2 = x^3 + 2$; $z = -(2^{62} + 2^{55} + 1) < 0$. The algorithm consists of two major parts

: namely, *Miller's loop* executed in line 2 to line 7, and *final exponentiation* executed in line 12 to line 13. In order to accommodate the negative r, line 8 computes a negation in $\mathbb{G}_2$ to make the final accumulator T the result of $[-|r|]Q$, and the value of $f_{(r,Q)}(P)$ is raised to the power p^6 which is equivalent to f^{-1} as shown in [2]. In line 10 to line 12, the algorithm computes $g_{(rQ,\pi_p(Q))}(P)$ and $g_{(rQ+\pi_p(Q),-\pi_p^2(Q))}(P)$, which are multiplied with f too. With above parameters the addition steps (line 5) invokes only four times throughout the Miller's loop which at the end helps to achieve higher speed of the pairing computation.

Algorithm 1. Optimal-ate pairing on BN curve ($t < 0$)

Input: $P = (x_P, y_P) \in E(\mathbb{F}_p[n])$, $Q = (x_Q\gamma^2, y_Q\gamma^3) \in E(\mathbb{F}_{p^{12}}) \cap Ker(\pi_p - p)$ with x_Q and $y_Q \in \mathbb{F}_{p^2}$, $r = |6t+2| = \Sigma_{i=0}^{s-1} r_i 2^i$.

Output: $a_{opt}(Q,P) \in \mathbb{F}_{p^{12}}$.

1. $T = (X_T\gamma^2, Y_T\gamma^3, Z_T) \leftarrow (x_Q\gamma^2, y_Q\gamma^3, 1)$, $f \leftarrow 1$;
2. **for** $i = s-2$ downto 0 **do**
3. $\quad g \leftarrow l_{(T,T)}(P)$, $T \leftarrow 2T$, $f \leftarrow f^2$, $f \leftarrow f \cdot g$;
4. $\quad$ **if** $r_i = 1$ **then**
5. $\quad\quad g \leftarrow l_{(T,Q)}(P)$, $T \leftarrow T + Q$, $f \leftarrow f \cdot g$;
6. $\quad$ **endif**
7. **endfor**
8. $T \leftarrow -T$, $f \leftarrow f^{p^6}$;
9. $Q_1 \leftarrow \pi_p(Q)$, $Q_2 \leftarrow -\pi_p^2(Q)$;
10. $g \leftarrow l_{(T,Q_1)}(P)$, $T \leftarrow T + Q_1$, $f \leftarrow f \cdot g$;
11. $g \leftarrow l_{(T,Q_2)}(P)$, $T \leftarrow T + Q_2$, $f \leftarrow f \cdot g$;
12. $f \leftarrow \left(f^{p^6-1}\right)^{p^2+1}$;
13. $f \leftarrow f^{(p^4-p^2+1)/n}$;
14. **return** f ;

Algorithm 1 employs arithmetic in $\mathbb{F}_{p^{12}}$. High-performance arithmetic over extension fields is achieved through a tower of extensions using irreducible binomials [18]. Accordingly, in our targeted setting we represent $\mathbb{F}_{p^{12}}$ using the towering scheme used in [2,22]:

$$\begin{aligned}
\mathbb{F}_{p^2} &= \mathbb{F}_p[i]/(i^2 - \beta), \text{ where } \beta = -1.\\
\mathbb{F}_{p^4} &= \mathbb{F}_{p^2}[s]/(s^2 - \xi), \text{ where } \xi = 1 + i.\\
\mathbb{F}_{p^{12}} &= \mathbb{F}_{p^4}[t]/(t^3 - s) \;=\; \mathbb{F}_{p^2}[\tau]/(\tau^6 - \xi).
\end{aligned}$$

Throughout the pairing computation we follow the towering $\mathbb{F}_{p^2} \rightarrow \mathbb{F}_{p^4} \rightarrow \mathbb{F}_{p^{12}}$ as it is shown in [7] that the arithmetic (mainly squaring) in this extension during final exponentiation is much cheaper than other towering extensions. The choice $p \equiv 3 \pmod 4$ accelerates arithmetic in $\mathbb{F}_{p^2}$, since multiplications by $\beta = -1$ can be computed as simple subtractions [22].

3 IP Cores on Xilinx FPGA

Various soft cores, specifically for memory and arithmetic functions, are provided by Xilinx which are easily configured into modern FPGAs like Virtex-6, Virtex-5, or Virtex-4 devices. The Xilinx LogiCORE IP block memory generator [25] core is an advanced memory constructor that generates area and performance-optimized memories using embedded block-RAM (BRAM) resources in Xilinx FPGAs. Available through the core generator software embedded with ISE tool, users can quickly create optimized memories to amend the performance of a design.

Two types of memory cores are used in the current design. Montgomery multiplication (Algorithm 2) uses $P^{(i)} = a^{(i)} \times b^{(i)}$ further for computing the final result ($c^{(i)}$). The value of $P^{(i)}$ is 512-bit long in the current cryptoprocessor. Therefore, we generate a memory core having 512-bit data width. This is a single port memory as its demand of read and write access are exclusive. The current multiplier performs at most 10 Montgomery multiplications in parallel. Thus we generate a memory core having nearest smallest size of 2^4 locations each of which are 512 bits long which is shown in Fig. 1. On the other hand, the top level design integrates two memory cores having 256-bit data width for accommodating one $\mathbb{F}_p$-element in a single memory location. In the current design, the datapath consists of several pipeline stages. Thus in order to avoid pipeline stalls, we generate a 256-bit wide true-dual-port memory core, where both ports are configured independently on the same shared memory space. The usage of this memory core in current design is described in § 5.3.

Similarly, Xilinx LogiCORE IP multiplier [25] implements high-performance, optimized multipliers. It allows the choice of LUTs or dedicated multiplier primitives to be selected for the core implementation. It further provides options for area or speed optimized design. The current design opts for the speed optimization on XtremeDSP slice that consists of dedicated multipliers. Thanks to LogiCORE for permitting a maximum of 64-bit unsigned operands which makes our design more simpler. The maximum speed of the 64-bit IP core is achieved through its 18 pipeline stages. However, the utilization of such pipeline depth is inconvenient for one pairing computation and need to allow several pipeline stalls. Its full utilization is only feasible through hyperthreading technology which in our design can be achieved by sharing the pipeline stages among several pairing computations. At the same time this advanced parallelism makes data-flow more complex and demands adequately large on-chip memory too. The current design tries to make a trade-off among speed, area, and design complexity. It finds that five stage pipeline of a 64-bit multiplier core provides the most suitable design with respect to computing one optimal-ate pairing at a time. With such design choices the IP core achieves a maximum operating clock frequency of 166 MHz on a Virtex-6 FPGA. Throughout the whole design process we preserve this operating frequency and always maintain the register to register combinatorial critical path having lesser delay than the period of above clock. The construction of such a critical-path constrained datapath makes rest of the design more challenging.

4 Base Field Multiplier

Multiplication in base field is the most important operation for computing a cryptographic pairing. In our case it is called $\mathbb{F}_p$-multiplication which can be executed by several techniques. This paper uses a straight forward Montgomery multiplication algorithm. The algorithm is executed by exploiting underlying Karatsuba multiplication for integers and by employing an efficient architecture. For executing extension field operations we always invoke our only multiplier for generating reduced result for each $\mathbb{F}_p$-multiplication. To speed up extension field arithmetic sometimes a lazy reduction technique is used [6]. However, instead of lazy reduction, our new multiplier executes multiple simultaneous $\mathbb{F}_p$-multiplications on pipelined datapath which ultimately speed up the overall pairing computation.

4.1 Montgomery and Karatsuba Combination

Montgomery multiplication algorithm avoids the division by p. The finite field multiplication is performed as modulo 2^n having $n = \lceil \log_2 p \rceil$ instead of modulo p. It is necessary to convert each operand from integer to its equivalent Montgomery form which costs another Montgomery multiplication. However, for repeated multiplications used in a pairing computation it is sufficient to convert the operands once at the beginning which is converted back at the end.

The Montgomery multiplication algorithm for large characteristic field is shown in Algorithm 2. The parenthesized indices represent the variables associated with that instruction. The indices are mainly used to identify an instruction and its associate variables inside our pipeline architecture. Algorithm 2 consists of three n bit integer multiplications, which determines the overall efficiency of the algorithm. Here we propose an efficient Montgomery multiplier architecture for modern FPGAs. Highly optimized IP cores available for FPGA devices together with our careful datapath design help to achieve an efficient pipelined architecture for Montgomery multiplication.

Algorithm 2. Montgomery multiplication

Input: $M = p$; $n = \lceil \log_2 M \rceil$; $R = 2^n$; $M' = -M^{-1} \bmod R$; $a^{(i)}, b^{(i)} \in \mathbb{Z}_M$.
Output: $a^{(i)} \cdot b^{(i)} \cdot R^{-1} \bmod M$.

1. $P^{(i)} \leftarrow a^{(i)} \cdot b^{(i)}$;
2. $U^{(i)} \leftarrow (P^{(i)} \bmod R) \cdot M' \bmod R$;
3. $c^{(i)} \leftarrow (P^{(i)} + U^{(i)} \cdot M)/R$;
4. **if** $c^{(i)} \geq M$ **then**
5. $\quad c^{(i)} \leftarrow c^{(i)} - M$;
6. **return** $c^{(i)}$;

Figure 1 depicts the proposed Montgomery multiplier architecture. It consists of a 256×256 bit Karatsuba multiplier, which is constructed by nine 64×64 bit multiplier cores. There is a small memory unit for holding intermediate result

of Montgomery multiplication which are used in later steps. Main novelty of the current design lies to efficient utilization of in-built multiplier and memory cores to achieve an optimized design on a modern FPGA platform. The top level of the architecture computes three integer multiplications in serial. The result of third multiplication is added with the result of the first one followed by a optional reduction (subtraction) to compute the result of a Montgomery multiplication. Although, the proposed design consists of a pipeline structure which is able to compute more than one multiplication in parallel. The detailed construction and its functionality is described in following sections.

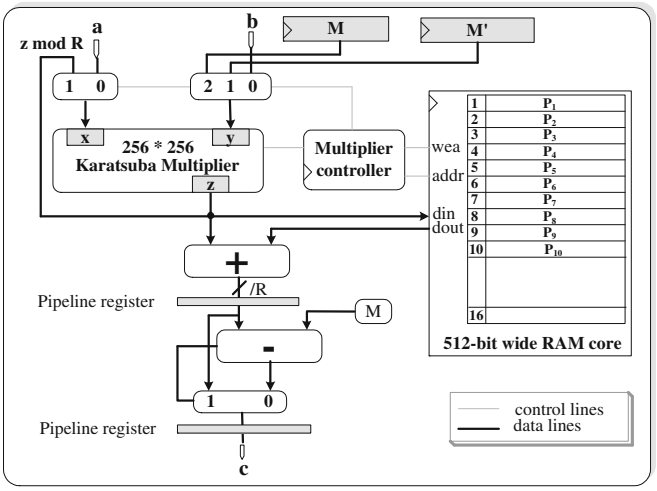


Fig. 1. The Montgomery multiplier

4.2 Delay Constrained Design

The proposed integer multiplier follows Karatsuba technique for performing 256-bit multiplications. Thus, three 128-bit multiplications, each of which is computed by three 64-bit multiplier cores, are performed in parallel. The post-multiplier operations are put into one additional pipeline stage for generating an 128×128 multiplication result. However, we find that the delay of datapath of post-multiplier operations is in between one and two clock periods for getting a 256×256 bit multiplication results. Thus, it is broken into two parts − adds two more pipeline stages. On the other hand, pre-multiplier datapath consists of an input multiplexer, an 128-bit adder and a 64-bit adder circuits, which forms two more pipeline stages. To sum up, the whole multiplier consists of 10 pipeline stages on which 10 independent multiplications can be executed in parallel.

The Montgomery multiplication algorithm (Algorithm 2) consists of three dependent integer multiplications. Therefore, we explore the parallelism at finite field level for which 10 independent $\mathbb{F}_p$-multiplications are fetched and issued

in parallel. The proposed design performs each Montgomery multiplication by executing operations divided in following five steps.

1. compute $P^{(i)} = a^{(i)} \times b^{(i)}$
2. store $P^{(i)}$ in RAM, and compute $U^{(i)} = (P^{(i)} \bmod R) \times M^{'}$
3. compute $V^{(i)} = (U^{(i)} \bmod R) \times M$
4. compute $c^{(i)} = (P^{(i)} + V^{(i)})/R$
5. compute $c^{(i)} = c^{(i)} - M$, if $c^{(i)} \geq M$.

We schedule the computation of $P^{(i)}$, $1 \leq i \leq 10$ first into the pipeline then all $U^{(i)}s$ followed by 10 $V^{(i)}s$. As soon as a $P^{(i)}$ gets out from the pipeline it is scheduled on-the-fly for computing $U^{(i)}$ as defined in step 2. The $P^{(i)}s$ are also stored into the 512-bit wide single-port-RAM (shown in Fig. 1) to use it further in step 4. Except $P^{(i)}$ it is not necessary to store other intermediate results $(U^{(i)}, V^{(i)})$. They are scheduled on-the-fly for further processing. The 31-st clock onwards from the beginning we start to receive $V^{(i)}$, which are then processed by two additional steps (step 4 and step 5) in two consecutive clock cycles. Therefore, to sum up, the cost of 10 Montgomery multiplications is 42 clock cycles in the current design. During these 42 clock cycles the multiplier communicates with external memory only at the first 10 clock cycles (to read $a^{(i)}$ and $b^{(i)}$) and the last 10 clock cycles (to write $c^{(i)}$). In between these two 10 clock cycles periods there are remaining 22 clock cycles when the external memory is free to access for other operations. These free cycles are utilized to accumulate $\mathbb{F}_p$ multiplication results to produce results in extension fields, to perform constant multiplications, and to perform other intermediate operations in pairing computation. This two levels of parallelism, namely, multiple $\mathbb{F}_p$-multiplications on a single unit and several $\mathbb{F}_p$-operations on different units, help to speed up pairing computation on the proposed design.

5 Architecture for Pairing

As shown in Algorithm 1, the pairing computation consists of following major operations.

1. *Doubling step:* An elliptic curve point doubling operation together with the computation of line function g.
2. *Addition step:* An elliptic curve point addition and the computation of g.
3. *Squaring:* Squaring of Miller variable f.
4. *Sparse multiplication:* A multiplication of Miller variable f with g having only half of the non-zero coefficients.
5. *Frobenius and Easy exponentiation:* Intermediate operations of Miller loop and hard exponentiation.
6. *Hard exponentiation:* Powering the intermediate result by $\phi_i(x)/n$ in cyclotomic subgroup.

In optimal-ate pairing on BN curve, first two steps are performed in $\mathbb{F}_{p^2}$, and most of the operations in other steps are performed in $\mathbb{F}_{p^{12}}$. Several advanced techniques can compute these extended field operations with much lower costs compared to their straight forward computation [8,16]. We choose the techniques having lower number of multiplications and squarings. The underlying operations in each techniques are computed in the base field. Therefore, we visualize the whole pairing computation as a sequence of $\mathbb{F}_p$ operations and try to execute them as fast as possible on a target platform.

5.1 Overview of the Architecture

The cost of a pairing computation is normally represented by the number of base field multiplications [17]. However, it is observed that apart from multiplications, a pairing computation consists of huge number of additions, subtractions and constant-multiplications. In current days the costs of a multiplication and an addition/subtraction with respect to time is almost same. Thus, the cost of a pairing computation equivalently depends on the efficiency of the "architecture" of all such operations. Moreover, this cost varies with the efficiency of the "scheduling" technique used on a specific implementation. Therefore, throughout the implementation we give equal attention to both architecture design and scheduling which in together maximizes the utilization of individual components and finally speeds up the pairing computation with constrained resources.

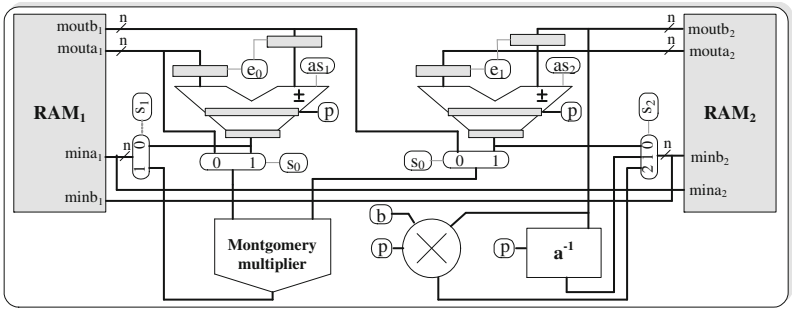


Fig. 2. The architecture for pairing computation

Figure 2 depicts the datapath of the architecture for pairing computation. It consists of a multiplier, two adder/subtractors, a constant-multiplier, and an inversion unit. All of them can independently perform respective operations in $\mathbb{F}_p$. In order to maximize their utilization we incorporate two true-dual-port RAM cores (call them RAM_1 and RAM_2) each of which contains identical data during a pairing computation. The operations in extension fields need to execute independent multiplications like $a \times b$ and $(a \pm b)(c \pm d)$. In order to support them without any pipeline stall, each of the multiplier inputs is multiplexed between an output port of RAM_1 and an output of adder/subtractor ($\pm$ block). The

architecture facilitates the port configuration in such a way that the output of each of functional units can be written in the same address of both RAMs in parallel. This helps to keep the identical data in both memory cores throughout the pairing computation which are exploited to improve the degree of parallelism.

5.2 Architecture Details

The architecture is developed with several pipeline stages in each of the functional units. Number of pipeline stages are identified to meet the maximum operating frequency provided by the 64×64 multiplier core as described in § 3.

Modular Adder Subtractor. The addition and subtraction in $\mathbb{F}_p$ can be realized by two consecutive n-bit adder circuits which produce final result in only one clock cycle. However, the latency of such a circuit in Virtex-6 FPGA is $11ns$, which is 1.8 times of our target critical path. We therefore divide this datapath into two pipeline stages which is illustrated in Fig. 3. The whole design now demands 130 Virtex-6 FPGA slices on which it achieves a maximum operating frequency of 183 MHz. Due to the pipeline structure its throughput is one $\mathbb{F}_p$ addition/subtraction per clock.

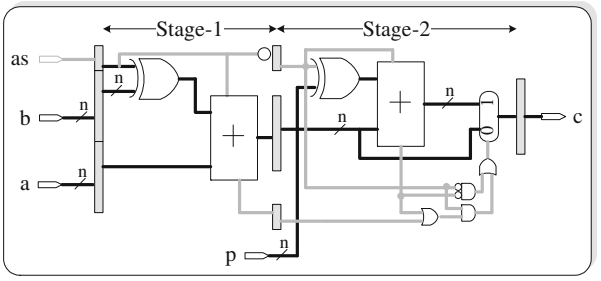


Fig. 3. Two stage pipeline for $\mathbb{F}_p$ addition and subtraction

Constant Multiplier. There are some operations in doubling and addition steps where a finite field element ($a \in \mathbb{F}_p$) is multiplied with small integers (≤ 6). We develop an adder based five stage pipeline structure for constant-multiplications which executes the target operations by following an addition chain. The first pipeline stage performs $2a \bmod p$, where doubling is simple rewiring followed by a conditional subtraction. Second and third stages is formed by following modular adder/subtractor (Fig. 3) unit. The only difference is that it performs both $3a = (2a + a) \bmod p$ and $4a = 2 \times 2a \bmod p$ in parallel. The second stage performs addition and doubling whereas we use the third stage for their reductions. The results of $3a$ and $4a$ are produced at the end of third stage. Similarly, fourth and fifth stages are formed to execute $5a = (2a+3a) \bmod p$ and $6a = 2 \times 3a \bmod p$. The pipeline registers are designed with optimum storage

space. For example, after second stage the value of a is no longer being used, so pipeline does not carry it beyond this point. Similarly, $4a$ is never used in further pipeline stages and the values of $2a$ and $3a$ are last used in the fourth stage. Through such observations, the pipelined constant-multiplier is designed, which optimizes overall area as well as time. Respective life-time diagram is shown in Fig. 4.

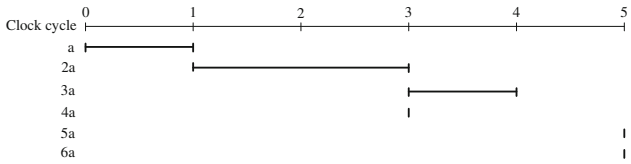


Fig. 4. Life time diagram of constant multiplication

Inversion Block. This block is developed as a dedicated unit for performing inversion in $\mathbb{F}_p$. It is based on the Extended Euclidean Algorithm. This unit is rarely (only once) used for computing a pairing. The functionality of this is described in § 6.4.

5.3 True Dual Port RAM

At this design stage we have already customized the datapath for pairing computation. So further speedup could be gained through the maximization of datapath utilization. A basic requirement for computing any two-input, one-output operation is to have two operands in parallel at the input ports of respective unit and an output destination available. This motivates the use of true-dual-port RAM for current design. It is called *true-dual-port* because both ports can independently perform read/write operations on the same shared memory space.

This RAM core is generated through Xilinx LogiCORE IP block generator tool (introduced in § 3). In the current design it is configure in write first mode having register at output port of the memory. Due to which we allow one clock cycle delay between address generation and availability of data at respective output port. At the same time, this register separates the datapath through multiplexer inside the memory block from the datapath between memory ports and the beginning of first pipeline stage of a functional unit. Otherwise, this combined datapath becomes longer than our target critical path. Each of this two memory cores contains 2^9 locations having 256-bit width that is sufficient to hold local and global variables during a pairing computation.

5.4 Working Principle of the Architecture

The overall architecture is constructed by observing that the pairing computation has several sets of independent base field operations. We perform an in-depth analysis on optimal-ate pairing algorithm to optimize such instruction sets

in order to maximize the utilization of the customized datapath with minimum storage space for temporary results. The analysis suggests that the formation of such instruction sets each of which containing at most 10 base field multiplications can utilize the current multiplier with minimum stall cycles for computing a single optimal-ate pairing. We call them *opt_set*. It is already described in § 5.1 that our architecture generates the result of $(a \pm b)$ and $(c \pm d)$ on-the-fly for performing $(a \pm b)(c \pm d)$. Thus multiplications in this form are also counting as a simple $\mathbb{F}_p$-multiplication during formation of the *opt_set*.

The execution of such an i-th *opt_set* on our architecture is as follows:

- It first schedules 10 multiplications on the pipelined multiplier in 10 consecutive clock cycles.
- From 11-th clock cycle, it schedules the additions, subtractions, and constant-multiplications on two adder/subtractor units and the constant-multiplier such that their results are written back to the memory within 31-st clock cycle. We perform those operations in this phase such that all multiplication results of $(i-1)$-th set are properly utilized and they are no longer used in future. The operations to prepare operands for multiplications of $(i+1)$-th *opt_set* are computed too in this phase.
- The results of the multiplications are available at multipliers output port from 32-nd clocks. These results are written back to the specific 10 conjugative locations in both RAMs from which the multiplications results of $(i-1)$-th *opt_set* are already utilized.

The execution of such a set takes 42 clock cycles, after which a new set is normally scheduled immediately from the next clock. Remember that all memory write operations in this implementation are performed in both RAMs (shown in Fig. 2 by $mina_1/mina_2$ and $minb_1/minb_2$) in same address for achieving higher degree of parallelism.

6 Scheduling and Pairing Computation

The execution control and the scheduling of operations on different functional units are performed by a state machine and few small counter logics. Here we present the instruction set formations for executing every step of the pairing computation. In the current design, the addition costs are hidden to multiplier cycles and therefore we use the techniques for internal operations especially, for extension-field arithmetic, having lower multiplications and squarings.

6.1 Execution of Doubling Step and f^2

There is no dependency between *doubling step* and f^2 computation which are therefore scheduled together. The step-by-step computation of the doubling step and f^2 is provided in Algorithm 5 of A.2. The formula of doubling step is followed from the state of the art existing pairing implementations [2,6,7]. We made rearrangements of the computations for making it suitable for our design. On

the other hand, it is shown in [16] that the representation of f in tower extension $\mathbb{F}_{((p^2)^2)^3}$ helps to reduce the operation count of f^2 computation in final exponentiation. Though this towering does not help to reduce the computation costs of f^2 within the Miller loop, but for simplicity, throughout the implementation we use the same towering to represent f.

The operations in this steps as well as other parts of the pairing computation described in this paper are performed either in $\mathbb{F}_{q^2}$ or in $\mathbb{F}_{q^3}$. Various technique for computing multiplication and squaring in such quadratic and cubic extension fields are explained in [8]. In this paper, we follow Karatsuba technique for computing both multiplication and squaring in $\mathbb{F}_{q^3}$, whereas, in case of $\mathbb{F}_{q^2}$ we use Karatsuba technique for multiplication and complex method for squaring. Formula for all such used techniques are provided in A.1. We represent the Miller variable f as :

$$\begin{aligned} f &= f_0 + f_1\tau + f_2\tau^2 + f_3\tau^3 + f_4\tau^4 + f_5\tau^5 \\ &= (f_0 + f_3 s) + (f_1 + f_4 s)t + (f_2 + f_5 s)t^2, \end{aligned}$$

which is considered as: $a_0 + a_1 t + a_2 t^2$ with $a_j \in \mathbb{F}_q, q = p^4$, $0 \leq j \leq 2$. Computation of f^2 in this towering extension consists of 36 multiplications in $\mathbb{F}_p$, which all are independent − though some of them need a few prior additions. On the other hand the computation of doubling step in Projective coordinate requires 27 multiplications in $\mathbb{F}_p$, which are not free to schedule at any point of time as they have several data dependencies. Thanks to our pipeline and memory architecture that we can manage all operations of this phase in 7 *opt_sets*. Among them first *opt_sets* containing 10 multiplications, second one consists of 8 multiplications and each of the remaining five consist of 9 multiplications. After receiving the multiplication results of final *opt_set* a few additions are performed for final accumulation.

6.2 The Addition Step

The addition step consists of 41 $\mathbb{F}_p$-multiplications which have several data dependencies. We compute them by forming five *opt_sets* with few intermediate additions during which the multiplier pipeline runs with bubbles. That is, we do not start $(i+1)$-th *opt_set* immediately after completing the execution of i-th *opt_set*. However, due to the dual adder/subtractor units these stall cycles are small compared to overall execution cycles. The formula for computing this step in Projective coordinates is provided in [2,6,7]. Algorithm 6 in A.2 provides the same with little rearrangements of operations to fit our current scheduling.

6.3 Computation of $f \cdot g$

The Karatsuba technique costs 54 $\mathbb{F}_p$-multiplications for computing a multiplication in $\mathbb{F}_{((p^2)^2)^3}$. However, in the $f \cdot g$ computations of steps 3, 5, 10, and 11 of Algorithm 1 only half of the coefficients of g are non-zero. Due to the sparse

value[1] of g, $f \cdot g$ consists of 39 $\mathbb{F}_p$-multiplications. The respective technique is provided in Algorithm 3 of A.2. We accommodate them in four *opt_sets*, where except the last one each *opt_set* contains 10 base field multiplications. A few additions, which depend on the multiplication results of final *opt_set*, are performed and update the value of f at respective locations at the end.

6.4 Inversion in $\mathbb{F}_p$

For powering f by $p^6 - 1$ in step 12 of Algorithm 1 it is essential to compute an inversion in $\mathbb{F}_{p^{12}}$, which is easily deduced as a single inversion in $\mathbb{F}_p$ along with several multiplications. The inverse of $a \in \mathbb{F}_p$ is in general computed by two methods – *Fermat's Little Theorem* or *Extended Euclidean Algorithm* (EEA). The first one computes inversion through exponentiation $a^{-1} \equiv a^{p-2} \bmod p$. On the other hand an efficient variant of EEA for $\mathbb{F}_p$-inversion is known as *Binary Inversion Algorithm*, which is primarily based on *gcd* computation. The exponentiation is efficiently implemented through an iterated square-and-multiply procedure for which an efficient implementation of the field multiplier is sufficient. However in our pipelined multiplier, execution of a single exponentiation is too costly as its i-th iteration cannot be started before completing $(i-1)$-th iteration. Thus, it will costs $33\lceil \log_2 p \rceil$ clock cycles with right-to-left execution.

On the other hand, an efficient implementation of binary inversion algorithm, as shown in [13], takes $2\lceil \log_2 p \rceil$ clock cycles. The stand alone implementation of this inversion unit requires 1350 Virtex-6 slices. On the other hand without this unit the current design takes $33\lceil \log_2 p \rceil$ number of clock cycles, which is 16.5 times more than the time taken by dedicated inversion unit. Thus we incorporate it into our design especially for computing a single inversion in final exponentiation. With our parameter settings without this unit the current design requires $7,874$ additional clock cycles for computing an inversion.

6.5 Exponentiation by $|z|$

After executing step 12 of Algorithm 1, the value of f becomes an element of the cyclotomic subgroup ($\mathbb{G}_{\phi_{12}(p)}$) in $\mathbb{F}_{p^{12}}$. An efficient technique used in this design for computing step 13 of Algorithm 1 (hard part of final exponentiation) is given in [23]. There are three exponentiations in $\mathbb{G}_{\phi_{12}(p)}$ by $|z|$ which are the most costly operations in this step. With our towering representation this squaring (Algorithm 4 in A.2) is much cheaper than a squaring computed in Miller's loop [16]. This squaring is executed by two *opt_sets* and few final additions and constant multiplications by our design. The whole exponentiation is performed by standard left-to-right square-and-multiply algorithm. Therefore, the multiplication is performed only if the respective exponent bit is one. This multiplication is a full multiplication (having no sparse operands) in $\mathbb{F}_{p^{12}}$, which consists of 54

[1] An operand in $\mathbb{F}_{p^{12}}$ is sparse when some of its coefficients are trivial (i.e., either zero or one).

independent $\mathbb{F}_p$-multiplications. We schedule them on the pipelined multiplier by forming six *opt_sets*.

In contrasts to pipelined design of [6] the current design uses MSB first method. Due to the low Hamming weight of $|z|$ the multiplications cost is vary low compared to the costs of squarings, and the current pipeline is suitable to execute one individual non-linear operation in $\mathbb{F}_{p^{12}}$. On the other hand, [2] shows a compressed technique for exponentiation by $|z|$ using Montgomery's simultaneous inversion trick [20]. However, this technique does not help to speed up pairing computation in our design as an inversion is 127 times slower than a multiplication in the current design.

7 Results

The whole design has been done in Verilog (HDL). Implementation has been performed on Xilinx ISE Design Suit 12.4. Table 1 shows the implementation results. On a Virtex-6 xc6vlx240t-3ff1759 FPGA the proposed design runs at a maximum frequency of $166MHz$. In total, with dedicated inversion unit, this design uses 5163 logic slices, 144 DSP slices and 21 BRAMS. It finishes computation of one 126-bit secure optimal-ate pairing in $375\mu s$. Table 2 gives the clock cycle counts required by the proposed design to computing different steps of an optimal-ate pairing on 126-bit secure BN curve.

Table 1. Area utilization on Virtex-6 FPGA

Current design	**Frequency** $[MHz]$	**Multipliers**	**Logic Elements**	**Memory**
with inversion	166	144 DSP48E1s	5163 slice‡	21 RAMB36E1
without inversion			3813 slice‡	
‡ : One Virtex-6 slice consists of four LUTs and eight flip-flops.				

Table 2. Cycle count for different steps of optimal-ate pairing on BN_{126} curve

Current design	$2T$, $g_{(T,T)}(P)$, and f^2	$T+Q$ and $g_{(T,Q)}(P)$	$f.g$	**Miller's Loop**	a^{-1} in $\mathbb{F}_p$	m^t in $\mathbb{G}_{\phi_k(p)}$	**Post M. Loop**	**Total**
with inv.	314	235	192	34,092	508	7,018	28,074	62,166
without inv.	314	235	192	34,092	8,448	7,018	36,014	70,106

7.1 Comparison with Recent Designs

Table 3 shows the comparative analysis of recent hardware and software results of pairing. With respect to latency of a pairing computation over BN curves with similar security level the present design achieves 32% speedup from the existing premier design proposed in [6]. Its slice counts is also relatively less with cost of more parallelism on higher number of DSP blocks. The clock cycle

count of the current design is reduced drastically due to higher parallelism on the pipelined datapath. In contrary the implementation of pairings over general elliptic curves having 128-bit security is still slower than that over a supersingular curves proposed in [14]. This may be due to the easier binary field arithmetic.

Table 3. Performance of hardware and software results of pairings

Designs	**Curve**	**FPGA**	**Area**	**Freq.** $[MHz]$	**Cycle** $[\times 10^3]$	**Delay** $[\mu s]$
This work (inv)	BN_{126}	xc6vlx240t-3	5163 Slices, 144 DSP	166	62	375
(without inv)	BN_{126}	xc6vlx240t-3	3813 Slices, 144 DSP	166	70	422
Cheung *et al.* [6]	BN_{126}	xc6vlx240t-2	7032 Slices, 32 DSP	250	143	573
	BN_{192}	Stratix-III	9910 A, 171 DSP	131	790	6030
Fan et al. [12]	BN_{128}	xc6vlx240t-3	4014 Slices, 42 DSP	210	245	1170
Ghosh *et al.* [15]	BN_{128}	xc4vlx200-12	52000 Slices	50	821	16400
Kammler *et al.* [19]	BN_{128}	130nm CMOS	97000 Gates	338	5,340*	15800
Ghosh *et al.* [14]	$E/\mathbb{F}_{2^{1223}}$	xc6vlx130t-3	15167 Slices	250	76†	190
Estibals [10]	$E/\mathbb{F}_{3^{5\cdot 97}}$	xc4vlx200-11	4755 Slices	192	429	2227
Aranha *et al.* [1]	$Co/\mathbb{F}_{2^{367}}$	xc4vlx25-11	4518 Slices	220	774*	3518
Naehrig *et al.* [21]	BN_{128}	core2 Q6600	–	2394	4,470	1860
Beuchat *et al.* [4]	BN_{126}	core i7 2.8GHz	–	2800	2,330	830
Aranha *et al.* [2]	BN_{126}	Phenom II	–	3000	1,562	520
Aranha *et al.* [1]	genus-2	Core i5	–	2530	2,440	960

† Estimated by the authors. * Estimation provided in [6].

Till 2010, the software for pairing outperforms the hardware and it was a bit uncomfortable to the hardware world. It was due to several unexplored in-built features available in the hardware platforms, especially FPGA platforms for pairing computation. However, at the end of last year it becomes true by the design shown in [6,14] for pairing too that customized hardware always outperforms a pure software. The current design in that respect not only gains the speedup from existing design but also it shows a direction for further improvement of pairing computations through exploitation of several highly optimized IP cores in different platforms.

8 Conclusion

In this paper we have proposed a core based architecture for pairing computation on general elliptic curves defined over large prime fields. Due to intelligent pipeline the proposed design has achieved a 32% speedup over existing designs. Moreover, a dedicated field inversion unit has reduced the clock cycle count of final exponentiation as well as a full pairing computation. The application of IP cores with more pipeline-depth may be targeted in future for executing multiple pairing computations at a time in order to handle several parallel client requests.

Acknowledgements. This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

References

1. Aranha, D.F., Beuchat, J.L., Detrey, J., Estibals, N.: Optimal Eta pairing on supersingular genus-2 binary hyperelliptic curves. Cryptology ePrint Archive, Report 2010/559, http://eprint.iacr.org/
2. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster Explicit Formulas for Computing Pairings over Ordinary Curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (2011)
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. Beuchat, J.-L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
5. Beuchat, J.L., Detrey, J., Estibals, N., Okamoto, E., Rodríguez-Henríquez, F.: Fast architectures for the η_T pairing over small-characteristic supersingular elliptic curves. IEEE Transactions on Computers 60(2) (2011)
6. Cheung, R.C.C., Duquesne, S., Fan, J., Guillermin, N., Verbauwhede, I., Yao, G.X.: FPGA Implementation of Pairings Using Residue Number System and Lazy Reduction. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 421–441. Springer, Heidelberg (2011)
7. Costello, C., Lange, T., Naehrig, M.: Faster Pairing Computations on Curves with High-Degree Twists. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 224–242. Springer, Heidelberg (2010)
8. Devegili, A., ÓhÉigeartaigh, C., Scott, M., Dahab, R.: Multiplication and squaring on pairing-friendly fields. Cryptology ePrint, Report 2006/471 (2006)
9. Duquesne, S., Guillermin, N.: A FPGA pairing implementation using the residue number system. Cryptology ePrint Archive, Report 2011/176 (2011), http://eprint.iacr.org/
10. Estibals, N.: Compact Hardware for Computing the Tate Pairing over 128-Bit-Security Supersingular Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 397–416. Springer, Heidelberg (2010)
11. Fan, J., Vercauteren, F., Verbauwhede, I.: Faster $\mathbb{F}_p$-Arithmetic for Cryptographic Pairings on Barreto-Naehrig Curves. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 240–253. Springer, Heidelberg (2009)
12. Fan, J., Vercauteren, F., Verbauwhede, I.: Efficient Hardware Implementation of $\mathbb{F}_p$-arithmetic for Pairing-Friendly Curves. IEEE Trasaction on Computers (2011), http://dx.doi.org/10.1109/TC.2011.78
13. Ghosh, S., Mukhopadhyay, D., Roychowdhury, D.: Petrel: power and timing attack resistant elliptic curve scalar multiplier based on programmable arithmetic unit. IEEE Trans. on Circuits and Systems I 58(11), 1798–1812 (2011)

14. Ghosh, S., Roychowdhury, D., Das, A.: High Speed Cryptoprocessor for η_T Pairing on 128-bit Secure Supersingular Elliptic Curves over Characteristic Two Fields. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 442–458. Springer, Heidelberg (2011)
15. Ghosh, S., Mukhopadhyay, D., Roychowdhury, D.: High Speed Flexible Pairing Cryptoprocessor on FPGA Platform. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 450–466. Springer, Heidelberg (2010)
16. Granger, R., Scott, M.: Faster Squaring in the Cyclotomic Subgroup of Sixth Degree Extensions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 209–223. Springer, Heidelberg (2010)
17. Hankerson, D., Menezes, A., Scott, M.: Software implementation of pairings. Cryptology and Info. Security Series, ch. 12, pp. 188–206. IOS Press (2009)
18. IEEE: P1363.3: Standard for Identity-Based Cryptographic Techniques using Pairings (2006), http://grouper.ieee.org/groups/1363/IBC/submissions/
19. Kammler, D., Zhang, D., Schwabe, P., Scharwaechter, H., Langenberg, M., Auras, D., Ascheid, G., Mathar, R.: Designing an ASIP for Cryptographic Pairings over Barreto-Naehrig Curves. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 254–271. Springer, Heidelberg (2009)
20. Montgomery, P.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Mathematics of Computation 48, 243–264 (1987)
21. Naehrig, M., Niederhagen, R., Schwabe, P.: New Software Speed Records for Cryptographic Pairings. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 109–123. Springer, Heidelberg (2010)
22. Pereira, G.C.C.F., Simplício Jr., M.A., Naehrig, M., Barreto, P.S.L.M.: A Family of Implementation-Friendly BN Elliptic Curves. Journal of Systems and Software 84(8), 1319–1326 (2011)
23. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 78–88. Springer, Heidelberg (2009)
24. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory 56(1), 455–461 (2010)
25. Xilinx. LogiCORE IP Block Generator (2010), http://www.xilinx.com

A Appendix

A.1 Multiplication and Squaring in $\mathbb{F}_{q^2}$ and $\mathbb{F}_{q^3}$

Let an element $\alpha \in \mathbb{F}_{q^2}$ be represented as $\alpha_0 + \alpha_1 X$, where $\alpha_0, \alpha_1 \in \mathbb{F}_q$ and X is an indeterminate. The formula of Karatsuba multiplication $c = ab$ on $\mathbb{F}_{q^2}$ is : $v_0 = a_0 b_0,\ v_1 = a_1 b_1,\ c_0 = v_0 + \zeta v_1,\ c_1 = (a_0 + a_1)(b_0 + b_1) - v_0 - v_1$, where $v_0, v_1, c_0, c_1, a_0, a_1, b_0, b_1 \in \mathbb{F}_q$. Here ζ is a quadratic non-residue in $\mathbb{F}_q$. The cost of such a multiplication is $(3m + 5a + 1\zeta_m)$ in $\mathbb{F}_q$. Similarly, the squaring $c = a^2$ on $\mathbb{F}_{q^2}$ using Complex method is computed by : $v_0 = a_0 a_1,\ c_0 = (a_0 + a_1)(a_0 + \zeta a_1) - v_0 - \zeta v_0,\ c_1 = 2v_0$. The cost of such a squaring is $(2m + 5a + 2\zeta_m)$ in $\mathbb{F}_q$. The equation of c_0 is easily deduced to $a_0^2 + \zeta a_1^2$, which eliminates additions but needs two squaring instead of one multiplication. In the current design squaring and multiplication is performed by same unit with same cost. On the other hand

the addition costs are hidden to multiplication costs, thus we use above formula to compute squaring in $\mathbb{F}_{q^2}$.

Similarly, let an element $\alpha \in \mathbb{F}_{q^3}$ be represented as $\alpha_0 + \alpha_1 X + \alpha_1 X^2$, where $\alpha_i \in \mathbb{F}_q$ and X is an indeterminate. The formula of Karatsuba multiplication $c = ab$ on $\mathbb{F}_{q^3}$ is : $v_0 = a_0 b_0$, $v_1 = a_1 b_1$, $v_2 = a_2 b_2$, $c_0 = v_0 + \vartheta((a_1 + a_2)(b_1 + b_2) - v_1 - v_2)$, $c_1 = (a_0 + a_1)(b_0 + b_1) - v_0 - v_1 + \vartheta v_2$, $c_2 = (a_0 + a_2)(b_0 + b_2) - v_0 + v_1 - v_2$, where $v_i, c_i, a_i, b_i, \in \mathbb{F}_q$. Here ϑ is a cubic non-residue in $\mathbb{F}_q$. The cost of such a multiplication is $(6m + 15a + 2\vartheta_m)$ in $\mathbb{F}_q$. This multiplication formula is also used for squaring $c = a^2$ on $\mathbb{F}_{q^3}$ replacing b by a. Thus the cost estimation for squaring replaces 6 multiplications by six squaring in $\mathbb{F}_q$.

A.2 Sub-Routines for Optimal-Ate Pairing

Algorithm 3. Computation of $f \cdot g$

Input: $f = (f_0 + f_3 s) + (f_1 + f_4 s)t + (f_2 + f_5 s)t^2$ and $g = (g_0 + g_3 s) + g_1 t \in \mathbb{F}_{((p^2)^2)^3}$ with $f_j, g_0, g_1, g_3 \in \mathbb{F}_{p^2}$, $0 \leq j \leq 5$.

Output: $f \cdot g$.

1. $v_0 \leftarrow (g_0 + g_3 s)(f_0 + f_3 s)$, $v_1 \leftarrow g_1 \cdot (f_1 + f_4 s)$,
2. $u_0 \leftarrow g_1 \cdot ((f_1 + f_2) + (f_4 + f_5 s)$,
   $u_1 \leftarrow ((g_0 + g_1) + g_3 s)((f_0 + f_1) + (f_3 + f_4 s))$,
   $u_2 \leftarrow (g_0 + g_3 s)((f_0 + f_2) + (f_3 + f_5 s))$;
3. $c_0 \leftarrow v_0 + \xi(u_0 - v_1)$, $c_1 \leftarrow u_1 - v_0 - v_1$, $c_2 \leftarrow u_2 - v_0 + v_1$;
4. **return** $c_0 + c_1 t + c_2 t^2$;

Algorithm 4. Squaring of f in $\mathbb{G}_{\phi_{12}(p)}$ [16]

Input: $f = (f_0 + f_3 s) + (f_1 + f_4 s)t + (f_2 + f_5 s)t^2 \in \mathbb{F}_{((p^2)^2)^3}$ with $f_j \in \mathbb{F}_{p^2}$, $0 \leq j \leq 5$.

Output: f^2.

1. $v_0 \leftarrow f_0 f_3$, $v_1 \leftarrow f_1 f_4$, $v_2 \leftarrow f_2 f_5$, $A_0 \leftarrow f_0 + f_3$, $A_1 \leftarrow f_0 + \xi f_3$, $B_0 \leftarrow f_1 + f_4$, $B_1 \leftarrow f_1 + \xi f_4$, $C_0 \leftarrow f_2 + f_5$, $C_1 \leftarrow f_2 + \xi f_5$;
2. $u_0 \leftarrow A_0 A_1$, $u_1 \leftarrow B_0 B_1$, $u_2 \leftarrow C_0 C_1$,
   $A_0 \leftarrow v_0 + \xi v_0$, $B_0 \leftarrow v_1 + \xi v_1$, $C_0 \leftarrow v_2 + \xi v_2$;
3. $c_0 \leftarrow 3(u_0 - A_0) - 2f_0$, $c_1 \leftarrow 6v_2 + 2f_1$, $c_2 \leftarrow 3(u_1 - B_0) - 2f_2$, $c_3 \leftarrow 6v_0 + 2f_3$, $c_4 \leftarrow 3(u_2 - C_0) - 2f_4$, $c_5 \leftarrow 6v_1 + 2f_5$;
4. **return** $(c_0 + c_3 s) + (c_1 + c_4 s)t + (c_2 + c_5 s)t^2$;

Algorithm 5. Doubling step and f^2

Input: $f = a_0 + a_1 t + a_2 t^2$ with a_0, a_1, and $a_2 \in \mathbb{F}_{p^4}$; $P = (x_P, y_P) \in E(\mathbb{F}_p)$;
$T = (X_T\tau^2, Y_T\tau^3, Z_T) \in E(\mathbb{F}_{p^{12}})$ with X_T, Y_T, and $Z_T \in \mathbb{F}_{p^2}$.

Output: $2T$, $l_{(T,T)}(P)$ and f^2.

1. $B \leftarrow Y_T^2$, $E \leftarrow 2Y_TZ_T$, $C \leftarrow 3Z_T^2$, $D \leftarrow 2X_TY_T$;
   $T_0 \leftarrow a_{0,0} + a_{0,1}$, $T_1 \leftarrow a_{0,0} + \xi a_{0,1}$;
2. $A \leftarrow X_T^2$, $U_0 \leftarrow a_{0,0}a_{0,1}$, $U_1 \leftarrow T_0T_1$;
   $g_3 \leftarrow B + iC$, $H \leftarrow 3C$, $F \leftarrow B + iH$, $G \leftarrow B - iH$;
   $T_0 \leftarrow a_{1,0} + a_{1,1}$, $T_1 \leftarrow a_{1,0} + \xi a_{1,1}$;
3. $g_0 \leftarrow Ey_P$, $J \leftarrow 4HC$, $g_1 \leftarrow -3Ax_P$, $I \leftarrow G^2$;
   $V_{0,0} \leftarrow U_1 - U_0 - \xi U_0$, $V_{0,1} \leftarrow 2U_0$;
4. $Z_{2T} \leftarrow 4BE$, $X_{2T} \leftarrow DF$, $U_0 \leftarrow a_{1,0}a_{1,1}$;
   $Y_{2T} \leftarrow I + J$, $A_0 \leftarrow a_{1,0} + a_{2,0}$, $A_1 \leftarrow a_{1,1} + a_{2,1}$;
5. $U_1 \leftarrow T_0T_1$, $W_0 \leftarrow a_{2,0}a_{2,1}$, $Z_0 \leftarrow A_0A_1$;
   $T_0 \leftarrow a_{2,0} + a_{2,1}$, $T_1 \leftarrow a_{2,0} + \xi a_{2,1}$, $X_0 \leftarrow A_0 + A_1$, $X_1 \leftarrow A_0 + \xi A_1$;
   $A_0 \leftarrow a_{0,0} + a_{1,0}$, $A_1 \leftarrow a_{0,1} + a_{1,1}$;
6. $W_1 \leftarrow T_0T_1$, $Z_1 \leftarrow X_0X_1$, $Y_0 \leftarrow A_0A_1$;
   $V_{1,0} \leftarrow U_1 - U_0 - \xi U_0$, $V_{1,1} \leftarrow 2U_0$, $X_0 \leftarrow A_0 + A_1$, $X_1 \leftarrow A_0 + \xi A_1$;
   $A_0 \leftarrow a_{0,0} + a_{2,0}$, $A_1 \leftarrow a_{0,1} + a_{2,1}$, $T_0 \leftarrow A_0 + A_1$, $T_1 \leftarrow A_0 + \xi A_1$;
7. $Y_1 \leftarrow X_0X_1$, $W_0 \leftarrow A_0A_1$, $W_1 \leftarrow T_0T_1$;
   $V_{2,0} \leftarrow W_1 - W_0 - \xi W_0$, $V_{2,1} \leftarrow 2W_0$, $V_{3,0} \leftarrow Z_1 - Z_0 - \xi Z_0$, $V_{3,1} \leftarrow 2Z_0$;
   $V_{3,0} \leftarrow V_{3,0} - V_{1,0} - V_{2,0}$, $V_{3,1} \leftarrow V_{3,1} - V_{1,1} - V_{2,1}$;
8. $c_{0,0} \leftarrow V_{0,0} + \xi V_{3,1}$, $c_{0,1} \leftarrow V_{0,1} + V_{3,0}$, $V_{3,0} \leftarrow Y_1 - Y_0 - \xi Y_0$, $V_{3,1} \leftarrow 2Y_0$;
   $c_{1,0} \leftarrow V_{3,0} - V_{0,0} - V_{1,0} + \xi V_{2,1}$, $c_{1,1} \leftarrow V_{3,1} - V_{0,1} - V_{1,1} + V_{2,0}$;
   $T_0 \leftarrow W_1 - W_0 - \xi W_0$, $T_1 \leftarrow 2W_0$;
   $c_{2,0} \leftarrow T_0 - V_{0,0} + V_{1,0} - V_{2,0}$, $c_{2,1} \leftarrow T_1 - V_{0,1} + V_{1,1} - V_{2,1}$;
9. **return** $(X_{2T}\tau^2, Y_{2T}\tau^3, Z_{2T})$, $g_0 + g_1\tau + g_3\tau^3$, $c_0 + c_1 t + c_2 t^2$;

Algorithm 6. Addition step

Input: $P = (x_P, y_P) \in E(\mathbb{F}_p)$, $Q = (x_Q\tau^2, y_Q\tau^3) \in E(\mathbb{F}_{p^{12}})$ and
$T = (X_T\tau^2, Y_T\tau^3, Z_T) \in E(\mathbb{F}_{p^{12}})$ with x_Q, y_Q, X_T, Y_T, and $Z_T \in \mathbb{F}_{p^2}$.

Output: $T + Q$ and $l_{(T,Q)}(P)$.

1. $E \leftarrow x_QZ_T - X_T$, $F \leftarrow y_QZ_T - Y_T$;
2. $E_2 \leftarrow E^2$, $F_2 \leftarrow F^2$, $g_3 \leftarrow x_QF - y_QE$;
3. $B \leftarrow X_TE_2$, $E_3 \leftarrow EE_2$, $A \leftarrow Z_TF_2 - 2B - E_3$;
4. $X_{T+Q} \leftarrow AE$, $Z_{T+Q} \leftarrow Z_TE_3$, $g_0 \leftarrow Ey_P$, $g_1 \leftarrow -Fx_P$;
5. $Y_{T+Q} \leftarrow F(B - A) - y_QE_3$;
6. **return** $(X_{T+Q}\tau^2, Y_{T+Q}\tau^3, Z_{T+Q})$, $g_0 + g_1\tau + g_3\tau^3$;

Faster Pairing Coprocessor Architecture

Gavin Xiaoxu Yao[1], Junfeng Fan[2],
Ray C.C. Cheung[1], and Ingrid Verbauwhede[2]

[1] Department of Electronic Engineering
City University of Hong Kong, Hong Kong SAR
Gavin.Yao@student.cityu.edu.hk, R.Cheung@cityu.edu.hk
[2] KU Leuven, ESAT/SCD-COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{Junfeng.Fan,Ingrid.Verbauwhede}@esat.kuleuven.be

Abstract. In this paper, we present a high-speed pairing coprocessor using Residue Number System (RNS) which is intrinsically suitable for parallel computation. This work improves the design of Cheung *et al.* [11] using a carefully selected RNS base and an optimized pipeline design of the modular multiplier. As a result, the cycle count for a modular reduction has been halved. When combining with the lazy reduction, Karatsuba-like formulas and optimal pipeline scheduling, a 128-bit optimal ate pairing computation can be completed in less than 100,000 cycles. We prototype the design on a Xilinx Virtex-6 FPGA using 5237 slices and 64 DSPs; a 128-bit pairing is computed in 0.358 ms running at 230MHz. To the best of our knowledge, this implementation outperforms all reported hardware and software designs.

Keywords: Optimal pairing, Residue Number System (RNS), Field Programmable Gate Array (FPGA).

1 Introduction

Pairing-Based Cryptography (PBC) has been applied to provide efficient solutions to several long-standing problems in cryptography, such as three-way key exchanges [22], identity-based encryptions [9], identity-based signatures [10], and non-interactive zero-knowledge proof systems [19]. As cryptographic schemes based on pairings are introduced and investigated, the performance of pairing computations also receives increasing interest [1,2,7,11,13,14,16,18,20,23,30].

The pairing computation is relatively complex and slow compared with other popular public-key primitives such as Rivest-Shamir-Adleman (RSA) [34] or Elliptic Curve Cryptography (ECC) [25,27]. For pairings over ordinary curves defined over prime fields $\mathbb{F}_p$, the computation can be broken down into modular multiplications and additions in the underlying fields. For example, an optimal ate pairing with 128-bit security consists of around ten thousand modular multiplications [2]. Thus, a faster pairing coprocessor is essential, and an efficient modular multiplier is the key component to make this happen.

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 160–176, 2013.

Due to the suitability for parallel implementation and the low cost for multiplications [31], Residue Number Systems (RNSs) have been introduced and studied for long integer modular multiplications [4,24,33,36]. With area complexity of $O(n)$, the time complexity is $O(1)$ for a multiplication and $O(n)$ for a modular reduction, where n is the number of machine-words to represent the modulus p. Recently, [11] has proposed a novel parameter selection method to ensure further complexity decrease for RNS modular reduction.

Moreover, lazy reduction and Karatsuba-like formulas are introduced to the computation. These techniques were first deployed for pairing by Scott [35] and then generalised by Aranha *et al.* [2]. In short, lazy reduction performs one reduction for multiple multiplications, which is possible for expressions like $\sum AB$ in $\mathbb{F}_p$; Karatsuba-like formulas save multiplications in extension fields. As such, the number of modular reductions and multiplications decreases.

In this paper, we improve the design of Cheung *et al.* [11] with a higher throughput. We show that reducing the number of moduli in the RNS basis leads to a faster RNS reduction. Although the size of multipliers in each channel is much larger than that of [11], the maximum frequency is only 8% lower due to the increase of pipeline stages. We maximally explore parallelisms in RNS arithmetic and the pairing algorithm to remove pipeline bubbles. As a result, our implementation of 126-bit optimal ate pairing uses only 78×10^3 cycles, 45% less than that of the design in [11]. Our pairing processor, implemented on a Xilinx Virtex-6 FPGA, requires 0.338 ms to finish one 126-bit optimal ate pairing and 0.358 ms for a 128-bit one. To the best of our knowledge, this implementation outperforms all previous hardware and software designs.

The rest of the paper is organized as follows: Section 2 provides a recap on mathematical background. Section 3 emphasizes on the optimal parameter selection. We illustrate the architectural design and the scheduling on the proposed architecture in detail in Section 4 and 5, respectively. Section 6 gives the FPGA implementation results of the proposed architecture, and compares them with recent results from the literatures. Finally, Section 7 concludes this paper.

2 Background

2.1 Bilinear Pairing

A bilinear pairing is a non-degenerate map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive groups and $\mathbb{G}_T$ is a subgroup of a multiplicative group. The core property of map e is linearity in both components, which enables the construction of novel cryptographic protocols. Popular pairings such as Tate pairing [5], ate pairing [21], R-ate pairing [26], optimal pairing [37] choose $\mathbb{G}_1$ and $\mathbb{G}_2$ to be specific cyclic subgroups of $E(\mathbb{F}_{p^k})$, and $\mathbb{G}_T$ to be a subgroup of $\mathbb{F}^*_{p^k}$.

The selection of parameters has an essential impact on the security and the performance of a pairing computation, and not all elliptic curves are suitable. We refer to Freeman *et al.* [15] for a summary of known pairing-friendly curves. Among them, Barreto and Naehrig (BN) described a parameterized family of

elliptic curves [6], and it is well-suited for computing asymmetric pairings. BN-curves are defined with $E : y^2 = x^3 + b, b \neq 0$ over $\mathbb{F}_p$, where $p = 36u^4 + 36u^3 + 24u^2 + 6u + 1$ and g, the order of E, is $36u^4 + 36u^3 + 18u^2 + 6u + 1$. Note that any $u \in \mathbb{Z}$ that generates prime p and g will suffice. BN-curves have embedding degree $k = 12$. Because of the limited space, we mainly focus on the discussion of the optimal ate pairing on BN-curves.

Let $E' : y^2 = x^3 + b/\zeta$ be a sextic twist of E with ζ not a cube nor a square in $\mathbb{F}_{p^2}$, and $E[g]$ be the subgroup of g-torsion points of E, then the optimal ate pairing is defined as [2,30]:

$$a_{opt} : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_T$$

$$(Q, P) \to (f_{r,Q}(P) \cdot l_{[r]Q,\pi_p(Q)}(P) \cdot l_{[r]Q+\pi_p(Q),-\pi_p^2(Q)}(P))^{\frac{p^{12}-1}{g}}$$

where $r = 6u + 2$. The group $\mathbb{G}_1 = E[g] \bigcap \mathrm{Ker}(\pi_p - [1]) = E(\mathbb{F}_p)[g]$ and $\mathbb{G}_2$ is the preimage $E'(\mathbb{F}_{p^2})[g]$ of $E[g] \bigcap \mathrm{Ker}(\pi_p - [p]) \subseteq E(\mathbb{F}_{p^{12}})[g]$ under the twisting isomorphism $\psi : E' \to E$. The group $\mathbb{G}_T$ is the subgroup of g-th roots of unity $\mu_g \subset \mathbb{F}^*_{p^{12}}$. The map $\pi_p : E \to E$ is the Frobenius endomorphism $\pi_p(x, y) = (x^p, y^p)$, and $f_{r,Q}(P)$ is a normalized function with divisor $(f_{r,Q}) = r(Q) - ([r]Q) - (r-1)(\mathcal{O})$. The line function, $l_{Q_1,Q_2}(P)$, is the line arising in the addition of Q_1 and Q_2 evaluated at point P.

Miller [28] proposed an algorithm that constructs $f_{r,Q}$ in stages by using double-and-add method. When u is selected as a negative integer, the corresponding Miller algorithm is shown in Algorithm 1 [2].

Algorithm 1. Optimal ate pairing on BN curves for $u < 0$ [2]

Require: $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, r = |6u + 2| = \sum_{i=0}^{\lfloor \log_2 r \rfloor} r_i 2^i$, where $u < 0$
Ensure: $a_{opt}(Q, P)$
1: $T \leftarrow Q, f \leftarrow 1$
2: **for** $i = \lfloor \log_2 r \rfloor - 1$ downto 0 **do**
3: $\quad f \leftarrow f^2 \cdot l_{T,T}(P), T \leftarrow 2T$
4: $\quad$ **if** $r_i = 1$ **then**
5: $\quad\quad f \leftarrow f \cdot l_{T,Q}(P), T \leftarrow T + Q$
6: $\quad$ **end if**
7: **end for**
8: $Q_1 \leftarrow \pi_p(Q), Q_2 \leftarrow \pi_p^2(Q)$
9: $T \leftarrow -T, f \leftarrow f^{p^6}$
10: $f \leftarrow f \cdot l_{T,Q_1}(P), T \leftarrow T + Q_1$
11: $f \leftarrow f \cdot l_{T,-Q_2}(P), T \leftarrow T - Q_2$
12: $f \leftarrow f^{(p^6-1)(p^2+1)(p^4-p^2+1)/g}$
13: **return** f

2.2 Residue Number System

A Residue Number System (RNS) uses a set of smaller integers to represent a large integer. An RNS is defined by a set of n coprime integer constants, $\mathfrak{B} = \{b_1, b_2, \ldots, b_n\}$. The set $\mathfrak{B}$ is also known as a *base*, and the element b_i, $1 \leqslant i \leqslant n$, is called *RNS modulus*, and each modulus forms an RNS *channel*. Let $M_{\mathfrak{B}} = \prod_{i=1}^{n} b_i$. Let $|a|_b$ be a modulo b, then any integer X, $0 \leqslant X < M_{\mathfrak{B}}$, can be uniquely represented as a set of smaller integers: $\{X\}_{\mathfrak{B}} = \{x_1, x_2, \ldots, x_n\}$, where $x_i = |X|_{b_i}$, $1 \leqslant i \leqslant n$. Similar to the radix-$2^w$ representation, we also call x_i a *digit* of X. The original value of X can be restored from $\{X\}_{\mathfrak{B}}$ using the Chinese Remainder Theorem (CRT):

$$X = \left| \sum_{i=1}^{n} \left| x_i \cdot B_i^{-1} \right|_{b_i} \cdot B_i \right|_{M_{\mathfrak{B}}}, \text{ where } B_i = \frac{M_{\mathfrak{B}}}{b_i} = \prod_{j=1, j \neq i}^{n} b_j, \ 1 \leqslant i \leqslant n. \quad (1)$$

Using RNS, arithmetic operations in $\mathbb{Z}/M_{\mathfrak{B}}\mathbb{Z}$ can be efficiently performed. Consider two integers X, Y and their RNS representations $\{X\}_{\mathfrak{B}} = \{x_1, x_2, \ldots, x_n\}$ and $\{Y\}_{\mathfrak{B}} = \{y_1, y_2, \ldots, y_n\}$, then

$$\{|X \odot Y|_{M_{\mathfrak{B}}}\}_{\mathfrak{B}} = \{|x_1 \odot y_1|_{b_1}, \ldots, |x_n \odot y_n|_{b_n}\}.$$

for $\odot \in \{+, -, \times, /\}$. The division is available only if Y is coprime with $M_{\mathfrak{B}}$, i.e. the multiplicative inverse of Y exists and is calculated in $\mathfrak{B}$.

Note that for all the basic operations $(+, -, \times, /)$, computations between x_i and y_i have no dependency on other digits, which largely simplifies the parallelization of the operations. Besides, the complexity of a multiplication in RNS is $O(n)$, while it is $O(n^2)$ using textbook arithmetics.

For every operation, there is an implicit channel reduction to bring the result in the range $[0, b_i)$. In order to accelerate the channel reduction, pseudo-Mersenne numbers of the form $b_i = 2^w - d_i$, where $d_i < 2^{\lfloor \frac{w}{2} \rfloor}$, are commonly chosen as moduli. To compute $x < 2^{2w}$ modulo b_i, one first performs the following step twice:

$$x \leftarrow (x \bmod 2^w) + d_i \cdot (x \operatorname{div} 2^w) \quad (2)$$

Then, x will be in the range of $[0, 2^{w+1})$, and after a conditional subtraction, one finishes the residue calculation. If the Hamming weight of d_i is small, multiplications by d_i can also be replaced by a few additions.

2.3 RNS Montgomery Algorithm and Faster Base Extension

Most cryptographic applications, such as pairings, require the operations modulo a prime, which prevents a direct utilization of RNS. This problem can be avoided by combining RNS representation and the Montgomery reduction algorithm [29].

Algorithm 2 shows the Montgomery modular multiplication algorithm without conditional subtraction in RNS context. A new base, $\mathfrak{C} = \{c_1, c_2, \ldots, c_n\}$, where $M_{\mathfrak{C}} = \prod_{i=1}^{n} c_i$ is coprime with $M_{\mathfrak{B}}$, is introduced to perform the division, and

Algorithm 2. RNS Montgomery Modular Multiplication [24]

Require: RNS bases $\mathfrak{B}$ and $\mathfrak{C}$ with $M_{\mathfrak{B}}, M_{\mathfrak{C}} > 2P$
Require: P, $M_{\mathfrak{B}}$, $M_{\mathfrak{C}}$ are pairwise coprime
Require: $\{X\}_{\mathfrak{B}}$, $\{X\}_{\mathfrak{C}}$, $\{Y\}_{\mathfrak{B}}$, $\{Y\}_{\mathfrak{C}}$ with $X, Y < 2P$
Precompute: $\{P'\}_{\mathfrak{B}} \leftarrow \{|-P^{-1}|_{M_{\mathfrak{B}}}\}_{\mathfrak{B}}$
Precompute: $\{M'\}_{\mathfrak{C}} \leftarrow \{|M_{\mathfrak{B}}^{-1}|_{M_{\mathfrak{C}}}\}_{\mathfrak{C}}$ and $\{P\}_{\mathfrak{C}}$
Ensure: $\{U\}_{\mathfrak{B}}$, $\{U\}_{\mathfrak{C}}$ s.t. $|U|_P = |XYM_{\mathfrak{B}}^{-1}|_P$, $U < 2P$

	in $\mathfrak{B}$	in $\mathfrak{C}$
1:	$\{T\}_{\mathfrak{B}} \leftarrow \{X\}_{\mathfrak{B}} \times \{Y\}_{\mathfrak{B}}$,	$\{T\}_{\mathfrak{C}} \leftarrow \{X\}_{\mathfrak{C}} \times \{Y\}_{\mathfrak{C}}$
2:	$\{Q\}_{\mathfrak{B}} \leftarrow \{T\}_{\mathfrak{B}} \times \{P'\}_{\mathfrak{B}}$	
3:	$\{Q\}_{\mathfrak{B}} \xrightarrow{Base\ Extension\ 1} \{Q\}_{\mathfrak{C}}$	
4:	$\{U\}_{\mathfrak{C}} \leftarrow (\{T\}_{\mathfrak{C}} + \{Q\}_{\mathfrak{C}} \times \{P\}_{\mathfrak{C}}) \times \{M'\}_{\mathfrak{C}}$	
5:	$\{U\}_{\mathfrak{B}} \xleftarrow{Base\ Extension\ 2} \{U\}_{\mathfrak{C}}$	
6:	**return** $\{U\}_{\mathfrak{B}}$ and $\{U\}_{\mathfrak{C}}$	

all the moduli from both $\mathfrak{B}$ and $\mathfrak{C}$ are pairwise coprime as $M_{\mathfrak{B}}$ and $M_{\mathfrak{C}}$ are coprime. The overhead is two Base Extensions (BEs) required in Algorithm 2.

BE is to compute $\{X\}_{\mathfrak{C}} = \{x'_1, x'_2, \ldots, x'_n\}$ given $\{X\}_{\mathfrak{B}} = \{x_1, x_2, \ldots, x_n\}$. We choose CRT method, specifically, the parallelizable Posch-Posch Method [24,32]. Given $\{X\}_{\mathfrak{B}}$, for (1), there must exist a certain integer $\lambda < n$ such that:

$$X = \left| \sum_{i=1}^{n} \left| x_i \cdot B_i^{-1} \right|_{b_i} \cdot B_i \right|_{M_{\mathfrak{B}}} = \left| \sum_{i=1}^{n} \xi_i \cdot B_i \right|_{M_{\mathfrak{B}}} = \sum_{i=1}^{n} \xi_i \cdot B_i - \lambda \cdot M_{\mathfrak{B}} \tag{3}$$

where $\xi_i = \left| x_i \cdot B_i^{-1} \right|_{b_i}$, $1 \leqslant i \leqslant n$, and λ can be calculated by:

$$\lambda = \left\lfloor \sum_{i=1}^{n} \frac{\xi_i \cdot B_i}{M_{\mathfrak{B}}} \right\rfloor = \left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{b_i} \right\rfloor \tag{4}$$

In [24], ξ_i/b_i is further approximated by $\xi_i/2^w$ as b_i is of the form $2^w - d_i, d_i > 0$. Once λ is obtained, $\{X\}_{\mathfrak{C}}$ can be computed by a matrix multiplication and channel reductions:

$$\left(x''_1, \ldots, x''_n\right)^T := \begin{pmatrix} |B_1|_{c_1} & \cdots & |B_n|_{c_1} \\ \vdots & \ddots & \vdots \\ |B_1|_{c_n} & \cdots & |B_n|_{c_n} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_n \end{pmatrix} - \lambda \begin{pmatrix} |M_{\mathfrak{B}}|_{c_1} \\ \vdots \\ |M_{\mathfrak{B}}|_{c_n} \end{pmatrix} \tag{5}$$

$$\{X\}_{\mathfrak{C}} \equiv \{x'_1, \ldots, x'_n\} := \{|x''_1|_{c_1}, \ldots, |x''_n|_{c_n}\} \tag{6}$$

Note that the elements in the matrix, $|B_i|_{c_j}$, $1 \leqslant i, j \leqslant n$, are constants and are determined as $|B_i|_{c_j} = \left| \prod_{k=1, k\neq i}^{n} b_k \right|_{c_j} = \left| \prod_{k=1, k\neq i}^{n} (b_k - c_j) \right|_{c_j}$. In [11], it shows

that the complexity of base extension can be reduced if the moduli in the two bases are close to each other. Define $\tilde{B}_{i,j} := \prod_{k=1,k\neq i}^{n}(b_k - c_j)$. $|B_i|_{c_j}$ or $\tilde{B}_{i,j}$ makes no difference to the results but $\tilde{B}_{i,j}$ can be much smaller when $b_k - c_j$ and n are small. We denote $v < w$ as the maximal bitlength of $\tilde{B}_{i,j}$. Now the $w \times w$ multiplications are substituted by the $v \times w$ multiplications.

Clearly, we need n $w\times w$-bit multiplications to calculate ξ_i and n^2 $v\times w$-bit multiplications for all $\sum_{i=1}^{n} \xi_i \cdot |B_i|_{c_j}, 1 \leqslant j \leqslant n$. In total, each reduction uses $4n$ digit multiplications and $2n^2$ $v\times w$-bit multiplications. Even though faster BE reduces the complexity, RNS reductions cost more than multiplications. Recall that a multiplication only takes $2n$ digit multiplications in Algorithm 2. Fortunately, lazy reduction is commonly used to reduce the number of the expensive modular reductions.

3 Parameter Selection

3.1 Pairing Parameter Selection

As stated in Section 2, we choose optimal ate pairing and BN-curve. Specifically, in order to achieve 128-bit security level, we choose $u = -(2^{63}+2^{22}+2^{18}+2^7+1)$ (p is 258-bit) as that in [11]. Also for comparison, we consider $u = -(2^{62}+2^{55}+1)$ (p is 254-bit) which is used in [2,11] and achieve 126-bit security level. We also deploy the same tower extension field as in [2]:

- $\mathbb{F}_{p^2} = \mathbb{F}_p[i]/(i^2 - \beta)$, where $\beta = -1$;
- $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^2}[W]/(W^6 - \zeta)$, where $\zeta = 1 + i$.

3.2 RNS Parameter Selection

As p is 258-bit, $n \cdot w$ should be greater than 258 to provide sufficient operating range. In [11], the authors have shown that it takes two cycles for a multiplication and $2 \cdot n/(\lceil w/v \rceil) + 4$ cycles for a modular reduction, where v is basically determined by n. The authors choose $n = 8$ in [11], so that $w = 33, v = 25$. As a result, a reduction takes 12 cycles, and 25×18-bit multipliers are in demand. In this section, we show that if $n = 4$, $w = 67$, v will be < 18, hence a reduction will only take six cycles; 18×18 multipliers are already competent.

The selected bases are chosen as follows: ($w = 67$)

$$\mathfrak{B} = \{2^w - 1, 2^w - 7, 2^w - 9, 2^w - 15\},$$

$$\mathfrak{C} = \{2^w - 0, 2^w - 3, 2^w - 5, 2^w - 31\}.$$

We use $\mathbb{L}_{\tilde{B}}$ and $\mathbb{L}_{\tilde{C}}$ to show the bit-length of all $\tilde{B}_{i,j}$ and $\tilde{C}_{i,j}$ in the matrix form, respectively.

$$\mathbb{L}_{\tilde{B}} = \begin{pmatrix} 10 & 8 & 7 & 6 \\ 9 & 8 & 7 & 6 \\ 7 & 8 & 7 & 6 \\ 14 & 14 & 14 & 14 \end{pmatrix}, \quad \mathbb{L}_{\tilde{C}} = \begin{pmatrix} 8 & 7 & 6 & 4 \\ 8 & 9 & 10 & 6 \\ 10 & 10 & 11 & 8 \\ 11 & 12 & 12 & 11 \end{pmatrix}.$$

The selected bases have the following features:

- All $\tilde{B}_{i,j}$ and $\tilde{C}_{i,j}$ are less than 18-bit. The largest element in $\tilde{B}_{i,j}$ and $\tilde{C}_{i,j}$ is 14-bit excluding the sign bit.
- The Hamming weights of all elements are less than 3 in non-adjacent form (NAF). Therefore, the channel reduction can be performed efficiently.
- b_i and c_i have the same NAF representation. For instance, b_3, $2^w - 15$, and c_3, $2^w - 31$ have the representation $2^w - 2^k + 1$, where $k = 4, 5$ respectively.
- All elements are equal to or less than 2^{67}, hence, after an addition or subtraction, the absolute value of the operand is less than 2^{68}, and can be represented by 69 bits including the sign bit.

The advantages of using the above features will be elaborated in the next section.

4 Architectural Design and Finite Field Arithmetics

4.1 The Controller and the Cox-Rower Architecture

The top level architecture is depicted by Fig. 1(a). The coprocessor can be divided into two major parts: the controller and the Arithmetic Logic Unit (ALU). We use an efficient and flexible micro-coded sequencer as the controller. By doing so, the controller maintains relatively small area, the ability to control accurately, and the flexibility for different curve and pairing operation. In this paper, we perform an optimal pairing computation on BN curves, however the proposed hardware architecture is capable of performing other pairings when the characteristic of the underlying field is less than 260 bits.

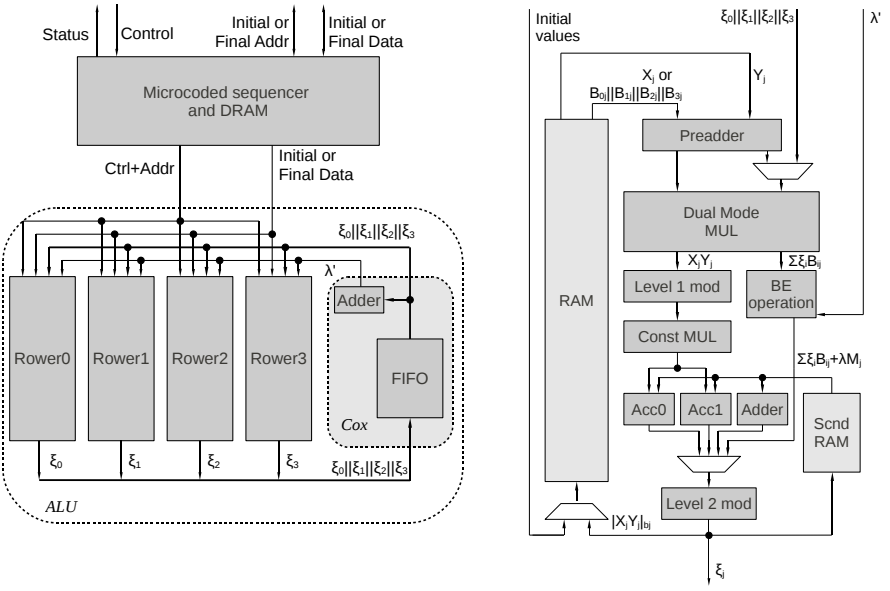
The ALU design is modified from that in [24]. We still call it the Cox-Rower architecture. Each rower performs operations in one channel of $\mathfrak{B}$ and one of $\mathfrak{C}$. Therefore, we have four rowers as $n = 4$. As described in Section 2, most operations are handled inside the channels independently, and the only step which requires communication between rowers is the ξ distribution in BE. We use a shared memory to redistribute the ξ values. Different from the original design [24], all ξ values for one BE are used at the same cycle, and hence, the ξ registers turn into a FIFO. The adder in the cox computes the value of λ for BE operation (4).

4.2 The Rower Design and Finite Field Arithmetics

We adjust the rower design to perform pairing computation more efficiently. The j-th rower architecture is shown in Fig. 1(b). For simplicity, we do not depict the control signal for multiplexers in the figures.

Dual Mode Multiplier

Fig. 2(a) depicts the dual mode multiplier in detail. It contains four 69×18-bit signed multipliers, and two different addition logics. One addition logic is for the 69×69-bit product, and the other is for the summation of four 69×18 products.



(a) Top level architecture with Cox-Rower ALU (b) The j-th rower design in the Cox-Rower architecture

Fig. 1. The Cox-Rower architecture

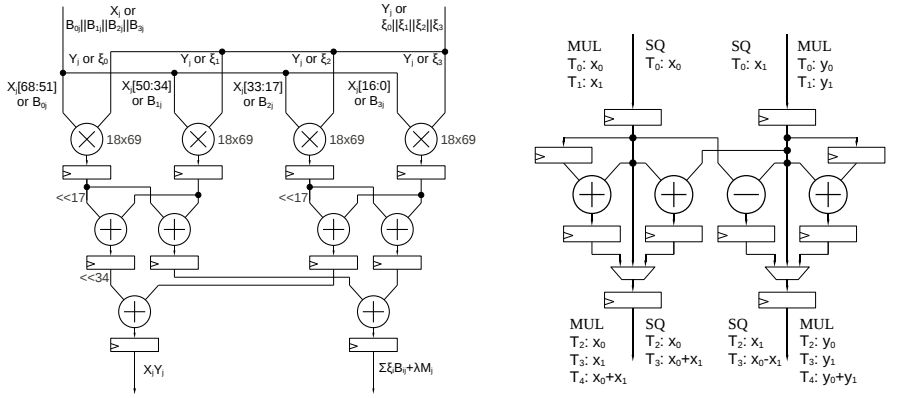
The sum of four 69×18 products is to perform one row of the matrix multiplication (5), since ξ_i is at most 69-bit and $\tilde{B}_{i,j}$ or $\tilde{C}_{i,j}$ is at most 18-bit. Therefore, a matrix multiplication only takes one cycle. Apart from the matrix multiplication, all the other multiplications are using full-length 69×69-bit signed representation. The four partial products are added up after the corresponding shift.

Preadders and Accumulators

Since the operation in $\mathbb{F}_{p^{12}}$ can be broken down to $\mathbb{F}_{p^2}$, the operation in $\mathbb{F}_{p^2}$ is the fundamental arithmetic for pairing. The preadders and the accumulators together provide fast $\mathbb{F}_{p^2}$ operation with Karatsuba-like formulas embedded. The squaring and multiplication in $\mathbb{F}_{p^2}$ can be written as follows:

$$\begin{aligned} z_0 + z_1 i \leftarrow & (x_0 + x_1 i)^2 \\ & = x_0^2 - x_1^2 + 2x_0 x_1 i \\ & = (x_0 + x_1)(x_0 - x_1) + 2x_0 x_1 i \end{aligned} \tag{7}$$

$$\begin{aligned} z_0 + z_1 i \leftarrow & (x_0 + x_1 i)(y_0 + y_1 i) \\ & = x_0 y_0 - x_1 y_1 + (x_0 y_1 + x_1 y_0) i \\ & = x_0 y_0 - x_1 y_1 + ((x_0 + x_1)(y_0 + y_1) - x_0 y_0 - x_1 y_1) i \end{aligned} \tag{8}$$



(a) Dual-mode multiplier design (b) Preadder design in each rower

Fig. 2. Architectural design of the components in a rower

When performing multiplication or squaring in $\mathbb{F}_{p^{12}}$, there are also operations as follows involved:

$$\begin{aligned} z_0 + z_1 i \leftarrow & (x_0 + x_1 i)^2 \zeta \\ = & \ x_0^2 - x_1^2 - 2x_0x_1 + (x_0^2 - x_1^2 + 2x_0x_1)i \\ = & \ (x_0 + x_1)(x_0 - x_1) - 2x_0x_1 + ((x_0 + x_1)(x_0 - x_1) + 2x_0x_1)i \quad (9) \end{aligned}$$

$$\begin{aligned} z_0 + z_1 i \leftarrow & (x_0 + x_1 i)(y_0 + y_1 i)\zeta \\ = & \ x_0y_0 - x_1y_1 - x_0y_1 - x_1y_0 + (x_0y_0 - x_1y_1 + x_0y_1 + x_1y_0)i \\ = & \ 2x_0y_0 - (x_0 + x_1)(y_0 + y_1) + ((x_0 + x_1)(y_0 + y_1) - 2x_1y_1)i \quad (10) \end{aligned}$$

The additions before multiplications are performed by the preadders and the ones after are done by accumulators. Fig. 2(b) shows the pipelined design of the preadders. There are only 2 patterns for $\mathbb{F}_{p^2}$ preaddition operations: squaring (7)/(9), and multiplication (8)/(10). The input and output sequences for squaring and multiplication are also provided in Fig. 2(b). We employ 2 accumulators in Fig. 1(b) and compute both z_0 and z_1 at the same time, because the same products are used for both z_0 and z_1 in (8), (9) and (10). As there are small constant multiplications in the algorithm, we also integrate a constant multiplier in the pipeline.

Channel Reducer

When performing the channel reduction, the multiplication by b_i or c_i in (2) is achieved by shifts and additions, as b_i and c_i are of small Hamming weight. Also as b_i and c_i are of the same NAF representations, the 2 channel reducers

can be integrated into one with a signal to control the shift. Using Rower 3 as an example, since $d_3 = 2^k - 1, k = 4, 5$, (2) is performed by one subtraction and one addition. To reduce modulo 2^w and $2^w - 1$, the arithmetic is even simpler and the design occupies less area.

After the first execution of (2), the bit-length of x will be less than or equal to $\lceil \log_2(d_i + 1) \rceil + w$, and it is only a few bits more than w as the selected d_i is very small. We choose to put the accumulators here, because compared to accumulating immediately after multiplication, the bit-length of the accumulators is shortened by almost a factor of 2. (Another choice is to put the accumulators after the second execution of (2), which needs another channel reduction for the accumulated results.) As long as the accumulated result x is smaller than $2^{2w-\lceil \log_2(d_i+1) \rceil}$ (which is always the case), the second execution of (2) can bring x less than 2^{w+1}. Therefore, the channel reduction is divided into Level 1, which performs the first execution, and Level 2, which performs the second reduction and result correction.

Other Components

Each rower also has a 3-port RAM, one write port and two read ports, so that the RAM can provide two operands at the same cycle. There is an adder and a secondary RAM (sncd RAM) involved. The secondary RAM stores the values for addition and the initial values for accumulation. For instance, the second RAM stores the value of $\{T\}_{\mathfrak{C}}$ generated in Step 1 of Algorithm 2, and this value is sent to accumulators as initial value in Step 4. Therefore, the addition and the multiplication are performed in parallel. The adder takes the operation which cannot be integrated in accumulation efficiently. The BE operation module computes $\lambda \cdot M_j$ and adds it to the sum generated by the dual mode multiplier.

4.3 Cycle Count of Finite Field Operations

Since all the other operations (namely, preaddition, accumulation, channel reduction) are hidden in the pipeline, the cycle count for each operation is the cycle used in the dual mode multiplier. For one modular reduction, there are one multiplication in $\mathfrak{B}$, three multiplications in $\mathfrak{C}$ and two matrix multiplications. As one matrix multiplication only takes 1 cycle, it takes 6 cycles to perform one reduction. Essentially, at least k reductions are required for a multiplication in $\mathbb{F}_{p^k}$, as the result has k coefficients. Excluding the reduction, one multiplication or squaring in $\mathbb{F}_p$ takes 2 cycles, one squaring and one multiplication in $\mathbb{F}_{p^2}$ take 4 and 6 cycles, respectively. One squaring in $\mathbb{F}_{p^{12}}$ is equivalent to 6 $\mathbb{F}_{p^2}$ squarings, 15 $\mathbb{F}_{p^2}$ multiplications and 12 reductions, while one normal multiplication in $\mathbb{F}_{p^{12}}$ is equivalent to 36 multiplications and 12 reductions. The cycle count provided in this section is under the condition that all the pipeline stages are filled. For the latency and the pipeline bubbles, readers can refer to Section 5 for more information.

5 Operation Scheduling

5.1 Optimal Pipeline Scheduling for RNS Montgomery Algorithm

While the pairing algorithm is more complex than ECC or RSA, it also contains more parallelisms to be exploited. Utilizing these parallelisms, pipeline structure is a popular technique to improve throughput with negligible latency overhead. Typically, the pipeline depth is equal to or less than the level of parallelism, otherwise, there will be pipeline bubbles introduced. On the other hand, adding more pipeline stages can achieve a higher frequency. Hence, the throughput of the implementation might still be higher even if new pipeline bubbles are introduced.

Before implementing Algorithm 2 using the proposed ALU, we first apply the following optimizations on the algorithm:

- The multiplication by $\{M'\}_{\mathfrak{C}}$ is distributed to $\{T\}_{\mathfrak{C}}$ and $\{QP\}_{\mathfrak{C}}$ in Step 4.
- We directly compute $\{\xi\}_{\mathfrak{C}}$ for BE2, and then compute $\{U\}_{\mathfrak{C}}$ by $\{\xi\}_{\mathfrak{C}} \times \{C\}_{\mathfrak{C}}$.
- We pre-compute the products of the constant multiplicands, i.e. $\{P'B^{-1}\}_{\mathfrak{B}}$, $\{M'C^{-1}\}_{\mathfrak{C}}$ and $\{PM'C^{-1}\}_{\mathfrak{C}}$, which reduces the number of operations.
- We use the computation of $\{T\}_{\mathfrak{C}}$, $\{R\}_{\mathfrak{C}} = \{T\}_{\mathfrak{C}} \times \{M'C^{-1}\}_{\mathfrak{C}}$, and $\{U\}_{\mathfrak{C}}$ to fill in the idle state.

Let the pipeline depth of the rower be τ, and assume that there are ρ modular multiplications which can be pipelined. Also, as the ξ delivery is a shared operation between rowers, let it take ϵ cycles. We generalize η, the minimal number of cycles to execute these ρ modular multiplications, as follows without detailed elaboration due to paper length:

$$\eta = \begin{cases} 8\rho, & \tau \leqslant \rho \\ 2\tau + 6\rho, & \rho < \tau \leqslant 2\rho - \epsilon \\ 4\tau + 2\rho + 2\epsilon, & 2\rho - \epsilon < \tau \leqslant 2\rho \\ 5\tau + 2\epsilon, & \tau > 2\rho \end{cases} \tag{11}$$

As each modular multiplication takes 8 cycles, the pipeline occupation rate is given by $\frac{8\rho}{\eta} \times 100\%$, and the number of pipeline bubbles is $\eta - 8\rho$. Typically, ϵ is a very small number. Therefore, the pipeline occupation rate will be over 80% if $\tau \leqslant 2\rho$. In other words, if there are ρ concurrent reductions can be performed, the number of pipeline stages can be up to 2ρ without introducing a lot of idle cycles. Furthermore, if there is no dependency, one can use the multiplications in the next pipeline round to fill in the idle states in the current round.

5.2 Scheduling of the Miller Loop

We examine the data dependency of Miller algorithm for the optimal ate pairing on BN curve. The explict formulas, the pipeline grouping, and the cycle count are provided by Table 1. The number of pipelined inputs, ρ, is the number of reductions at the same pipeline group. In fact, $\tau = 18$ for our FPGA prototype. Using the scheduling shown in Table 1 for a single pairing, it guarantees $\tau < 2\rho$,

Table 1. Pipeline scheduling and operation counts of Miller loop

Condition	Step	Pipeline group and Formulas	ρ for Single Pairing	Cycle Count	
				3 Pairings/3	Single Pairing
$r_i = 0$	1	$A = y_1^2$, $B = 3b'z_1^2$, $C = 2x_1y_1$ $D = 3x_1^2$, $E = 2y_1z_1$, $f_{0,1,2} = (f^2)_{0,1,2}$	16	180	180
	2	$x_3 = (A-3B)C$, $y_3 = A^2 + 6AB - 3B^2$, $z_3 = 4AE$ $l_3 = A - B$, $l_1 = \overline{x_P}D$, $l_0 = y_PE$, $f_{3,4,5} = (f^2)_{3,4,5}$	16	186	190
	3	$f = f \cdot l$ $x_1 = x_3, y_1 = y_3, z_1 = z_3$	12	180	180
$r_i = 1$	1	$A = y_1^2$, $B = 3b'z_1^2$, $C = 2x_1y_1$ $D = 3x_1^2$, $E = 2y_1z_1$, $f_2 = (f^2)_2$	12	116	116
	2	$x_3 = (A-3B)C$, $y_3 = A^2 + 6AB - 3B^2$, $z_3 = 4AE$ $l_3 = A - B$, $l_1 = \overline{x_P}D$, $l_0 = y_PE$, $f_{3,4,5} = (f^2)_{3,4,5}$ $x_1 = x_3, y_1 = y_3, z_1 = z_3$	16	186	190
	3	$A = y_1 - y_Qz_1$, $B = x_1 - x_Qz_1$ $f = f \cdot l$	16	224	228
	4	$C = A^2$, $D = B^2$ $l_3 = y_QB - x_QA$, $l_1 = x_PA$, $l_0 = \overline{y_P}B$	10	88	104
	5	$E = BD$, $C = BD + z_1C - 2x_1D$, $D = BD + z_1C - 2x_1D - x_1D$, $f = f \cdot l$	18	240	240
	6	$x_3 = BC$, $y_3 = -AD - y_1E$, $z_3 = z_1E$ $x_1 = x_3, y_1 = y_3, z_1 = z_3$, $f_{0,1} = (f^2)_{0,1}$	10	122	122

and hence the pipeline occupation rate is over 80%. Moreover, we use multiplications in the next group to fill in the idle state in the current, so that there is only 6 idle cycle when $r_i = 0$ (only 1% to the total cycles). For $r_i = 1$, there are 26 idle cycles. However, as the r is chosen with small Hamming weight, $r_i = 1$ rarely happens, and the pipeline occupation rate is over 98% for Miller loop.

To get rid of pipeline bubbles, one possible way is to perform multiple concurrent pairings, which happens to certain protocols [12] or the operation on a server side. We use Block RAM on FPGA to serve as the Rower memory, and even with the minimal configuration, it can store the intermediate values for 3 pairing computations with the same public parameter set. Therefore, our design is naturally suitable for 3 pairing computations at the same time.

5.3 Scheduling of the Final Steps

The final steps of an optimal pairing include 2 final additions (Step 10, 11 in Algorithm 1) and a final exponentiation (Step 12 in Algorithm 1). The operation count is given in Table 2. The formulas for final additions are the same with the point addition formulas in the Miller loop. The Frobenius endomorphism of Q_1, Q_2 computation is also included in the cycle count.

For the final exponentiation, the power $(p^{12}-1/g)$ is factored into three small exponents: (p^6-1), (p^2+1), and $(p^4-p^2+1)/g$. To compute (p^6-1), it requires an inversion. The formulas from [35] is utilized to transform the inversion in $\mathbb{F}_{p^{12}}$

Table 2. Cycle counts of final steps for the pairing computation

Step	3 pairings/3	Occu. Rate	Single Pair.	Occu. Rate	Step	3 pairings/3	Occu. Rate	Single Pair.	Occu. Rate
1st final add	502	99.2%	582	87.6%	p^2+1	356	100%	398	89.4%
2nd final add	284	98.6%	394	74.1%	mul in hardexp	288	100%	300	96.0%
p^6-1, before inv	237.6	86.9%	431	49.6%	sq in hardexp	138	100%	150	92.0%
single sq in inv	28.3	42.4%	79	15.2%	exp to p, $3p$	90	100%	106	84.9%
p^6-1, after inv	265.3	94.2%	374	69.0%	exp to $2p$	68	100%	98	69.4%

Table 3. Logic Utilizations of Virtex-6 XC6VLX240T-1

Logic Utilization	# DSPs	# LUTs	# Reg.	# Occupied Slices	# 18Kb BRAMs
Used	64	18,794	21,505	5,237	41
Available	768	150,720	301,440	37,680	832
Utilization	8%	12%	7%	13%	5%

to an inversion in $\mathbb{F}_p$. To perform this inversion in $\mathbb{F}_p$, we employ Fermat's little theorem, i.e. $d^{-1} = d^{p-2} \bmod p$ if p is prime, and exponentiation with LSB-first as [11]. The problem with this exponentiation in $\mathbb{F}_p$ is that the computation cannot be pipelined. Indeed, only one stage is taken out of all pipeline stages, which causes low pipeline occupation rate and a huge waste. We use the same formulas as [11] for the exponentiation to $(p^4-p^2+1)/g$, also known as *hard exponentiation*. We also use cyclotomic subgroup structure to accelerate squaring $\mathbb{F}_{p^{12}}$ [2,11]. Finally, in one hard exponentiation, 3 exponentiations to u take most of the time. Besides, there are 6 exponentiations to p or $3p$, 1 exponentiation to $2p$, 13 multiplications, and 4 squarings. We also consider that potentially 3 pairings are computed at the same time.

6 Implementation and Comparisons

6.1 Implementation

The pairing coprocessor is implemented on a Xilinx Virtex-6 XC6VLX240T-1 FPGA, which embeds 25×18 DSP slices (in fact, 18×18 multipliers are sufficient for our design). As there are 4 rowers and each rower contains 16 DSP slices, the total number of used DSP is 64. Data RAMs used in each Rower are implemented with 4 block RAMs. The block RAMs (BRAMs) also serve as the microcode sequencer. We have designed an 18-depth carefully tuned pipeline structure, which ensures the coprocessor to operate at 230MHz (with timing constrains of 210MHz, the coprocessor only occupies 3879 slices). Note that the logic utilization of DSPs and Slices is quite balanced (all around 10%), i.e. given a bounded area, we use all the available logic resources efficiently.

Table 4 gives number of cycles used in optimal ate pairing for our design. As expected, it provides better throughput if multiple pairings are computed at

Table 4. The number of cycles for one pairing

	Security [bit]	Miller loops	Final additions	Exp to p^6-1	Exp to p^2+1	Hard exp	Total # Cycles	Time [ns]
single pairing	126	37,000	976	20,792	398	34,934	94,100	409.1
3 pairings/3	126	36,664	786	7,663	356	32,300	77,769	338.1
single pairing	128	39,350	976	21,108	398	37,184	99,016	430.5
3 pairings/3	128	38,930	786	7,776	356	34,442	82,290	357.8

the same time. On the other hand, fine-pipelined architecture is affordable for pairing computation, as RNS and pairing algorithm provide enough parallelism. The increased frequency of the design could compensate the overhead induced by the pipeline bubbles.

6.2 Comparison

Table 5 provides the area and timing information of recent reported pairing implementations on both software and hardware platforms. The fastest FPGA implementation of pairing on BN curve in literature is Design II [11]. Compared

Table 5. Performance comparison of software and hardware implementations of pairings at around 128-bit security

Design	Pairing	Security [bit]	Platform	Algorithm	Area	Freq. [MHz]	Cycle [$\times 10^3$]	Delay [ms]
Ours	optimal ate	126	Xilinx FPGA (Virtex-6)	RNS (Parallel)	5237 slices 64 DSPs	210	78	0.338
		128					82	0.358
[11] Design II	optimal ate	126	Xilinx FPGA (Virtex-6)	RNS (Parallel)	7032 slices 32 DSPs	250	143	0.573
[11] Design I	optimal ate	126	Altera FPGA (Stratix III)	RNS (Parallel)	4233 ALMs 72 DSPs	165	176	1.07
[14]	ate	128	Xilinx FPGA (Virtex-6)	HMM (Parallel)	4014 slices 42 DSPs	210	336	1.60
	optimal ate						245	1.17
[16]	Tate	128	Xilinx FPGA (Virtex-4)	Blakley	52k Slices	50	1,730	34.6
	ate						1,207	24.2
	optimal ate						821	16.4
[23]	Tate	128	ASIC (130 nm)	Montgomery	97 kGates	338	11,627*	34.4
	ate						7,706*	22.8
	optimal ate						5,340*	15.8
[17]	η_T over $\mathbb{F}_{21223}$	128	Xilinx FPGA (Virtex-6)	-	15167 Slices	250	47.6	0.19
[13]	Tate over $\mathbb{F}_{3^{5\cdot 97}}$	128	Xilinx FPGA (Virtex-4)	-	4755 Slices 7 BRAMs	192	429	2.23
[1]	optimal Eta over $\mathbb{F}_{2^{367}}$	128	Xilinx Virtex-4	-	4518 Slices	220	774*	3.52
[20]	ate	128	64-bit Core2	Montgomery	-	2400	15,000	6.25
	optimal ate						10,000	4.17
[18]	ate	128	64-bit Core2	Montgomery		2400	14,429	6.01
[30]	optimal ate	128	Core2 Quad	Hybrid Mult.	-	2394	4,470	1.86
[7]	optimal ate	126	Core i7	Montgomery	-	2800	2,330	0.83
[2]	optimal ate	126	Phenom II	Montgomery	-	3000	1,562	0.52
[3]	η_T over $\mathbb{F}_{21223}$	128	Xeon (8 cores)	-	-	2000	3,020	1.51
[8]	η_T over $\mathbb{F}_{3509}$	128	Core i7 (8 cores)	-	-	2900	5,423	1.87
[1]	opt. Eta $\mathbb{F}_{2^{367}}$	128	Core i5	-	-	2530	2,440	0.96

* Estimated by the authors.

with this work, our prototype provides 41.0% speed-up, and with 25.5% less FPGA area. Also, the underlying multiplier used in our design is 18×18, which is popular among the off-the-shelf products, while Design II [11] requires 25×18 multipliers, which are only available on high-end FPGAs. [17] provides almost twice faster implementations on binary field, but with around 3 times of FPGA area as ours. Also, our work is the first hardware design which outperforms software implementations for pairing on BN curves.

7 Conclusions

In this paper, we present an efficient pairing coprocessor using RNS and lazy reduction techniques. We introduce a set of RNS moduli that are suitable for 258-bit modular multiplications in the pairing computation. The proposed architecture is prototyped on a Xilinx Virtex-6 FPGA, which utilizes 5237 slices and 64 DSPs, and can run at 230 MHz. The coprocessor computes one optimal pairing in 0.358 ms. To the best of our knowledge, this is a speed record for hardware implementations for pairings on BN-curve that achieves 128-bit security. For the future work, we plan to implement other curves and different types of pairings on this architecture. Furthermore, we will provide an optimal parameter set and pairing implementations for higher security level including 192-bit or 256-bit security.

Acknowledgements. We would like to thank Nicolas Guillermin for the helpful discussion on improving the scheduling of the RNS Montgomery algorithm. We also thank the anonymous referees for the helpful comments and improving the quality of this work. This work is partly supported by the City University of Hong Kong Start-up Grant 7200179, and partly by the Research Council KU Leuven: GOA TENSE (GOA/11/007) and by the European Commission under contract number ICT-2007-216676 ECRYPT-II NoE. In addition, this work is supported in part by the Flemish Government, FWO G.0550.12N and by the Hercules Foundation AKUL/11/19.

References

1. Aranha, D., Beuchat, J.L., Detrey, J., Estibals, N.: Optimal eta pairing on supersingular genus-2 binary hyperelliptic curves. Cryptology ePrint Archive, Report 2010/559 (2010), http://eprint.iacr.org/
2. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster Explicit Formulas for Computing Pairings over Ordinary Curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (2011)
3. Aranha, D.F., López, J., Hankerson, D.: High-Speed Parallel Software Implementation of the η_T Pairing. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 89–105. Springer, Heidelberg (2010)
4. Bajard, J.C., Kaihara, M., Plantard, T.: Selected RNS bases for modular multiplication. In: ARITH 2009: Proceedings of the 2009 19th IEEE Symposium on Computer Arithmetic, pp. 25–32. IEEE Computer Society, Washington, DC (2009)

5. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–369. Springer, Heidelberg (2002)
6. 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)
7. Beuchat, J.-L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
8. Beuchat, J.-L., López-Trejo, E., Martínez-Ramos, L., Mitsunari, S., Rodríguez-Henríquez, F.: Multi-core Implementation of the Tate Pairing over Supersingular Elliptic Curves. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 413–432. Springer, Heidelberg (2009)
9. 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)
10. Cha, J.C., Cheon, J.H.: An Identity-Based Signature from Gap Diffie-Hellman Groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2003)
11. Cheung, R.C.C., Duquesne, S., Fan, J., Guillermin, N., Verbauwhede, I., Yao, G.X.: FPGA Implementation of Pairings Using Residue Number System and Lazy Reduction. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 421–441. Springer, Heidelberg (2011)
12. Dutta, R., Barua, R., Sarkar, P.: Pairing-based cryptographic protocols: A survey. Cryptology ePrint Archive, Report 2004/064 (2004)
13. Estibals, N.: Compact Hardware for Computing the Tate Pairing over 128-Bit-Security Supersingular Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 397–416. Springer, Heidelberg (2010)
14. Fan, J., Vercauteren, F., Verbauwhede, I.: Efficient hardware implementation of $\mathbb{F}_p$-arithmetic for pairing-friendly curves. IEEE Transactions on Computers 61(5), 676–685 (2012)
15. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology 23, 224–280 (2010)
16. Ghosh, S., Mukhopadhyay, D., Roychowdhury, D.: High Speed Flexible Pairing Cryptoprocessor on FPGA Platform. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 450–466. Springer, Heidelberg (2010)
17. Ghosh, S., Roychowdhury, D., Das, A.: High Speed Cryptoprocessor for η_T Pairing on 128-bit Secure Supersingular Elliptic Curves over Characteristic Two Fields. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 442–458. Springer, Heidelberg (2011)
18. Grabher, P., Großschädl, J., Page, D.: On Software Parallel Implementation of Cryptographic Pairings. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 35–50. Springer, Heidelberg (2009)
19. 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)
20. Hankerson, D., Menezes, A., Scott, M.: Software Implementation of Pairings. Cryptology and Information Security Series, vol. 2, pp. 188–206. IOS Press (2009)
21. Hess, F., Smart, N., Vercauteren, F.: The Eta pairing revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)

22. Joux, A.: A one round protocol for tripartite Diffie-Hellman. Journal of Cryptology 17, 263–276 (2004)
23. Kammler, D., Zhang, D., Schwabe, P., Scharwaechter, H., Langenberg, M., Auras, D., Ascheid, G., Mathar, R.: Designing an ASIP for Cryptographic Pairings over Barreto-Naehrig Curves. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 254–271. Springer, Heidelberg (2009)
24. Kawamura, S.-I., Koike, M., Sano, F., Shimbo, A.: Cox-Rower Architecture for Fast Parallel Montgomery Multiplication. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 523–538. Springer, Heidelberg (2000)
25. Koblitz, N.: Elliptic Curve Cryptosystem. Mathematics of Computation 48, 203–209 (1987)
26. Lee, E., Lee, H.S., Park, C.M.: Efficient and generalized pairing computation on abelian varieties. IEEE Transactions on Information Theory 55(4), 1793–1803 (2009)
27. Miller, V.S.: Use of Elliptic Curves in Cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
28. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17, 235–261 (2004)
29. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44(170), 519–521 (1985)
30. Naehrig, M., Niederhagen, R., Schwabe, P.: New Software Speed Records for Cryptographic Pairings. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 109–123. Springer, Heidelberg (2010)
31. Parhami, B.: Computer Arithmetic: Algorithms and Hardware Designs. Oxford University Press (2000)
32. Posch, K., Posch, R.: Base extension using a convolution sum in residue number systems. Computing 50, 93–104 (1993)
33. Posch, K., Posch, R.: Modulo reduction in residue number systems. IEEE Transactions on Parallel and Distributed Systems 6(5), 449–454 (1995)
34. Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
35. Scott, M.: Implementing cryptographic pairings. In: Pairing-Based Cryptography - Pairing 2007. LNCS, vol. 4575, pp. 117–196. Springer (2007)
36. Szerwinski, R., Güneysu, T.: Exploiting the Power of GPUs for Asymmetric Cryptography. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 79–99. Springer, Heidelberg (2008)
37. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory 56(1), 455–461 (2010)

Implementing Pairings at the 192-Bit Security Level

Diego F. Aranha[1], Laura Fuentes-Castañeda[2], Edward Knapp[3], Alfred Menezes[3], and Francisco Rodríguez-Henríquez[2]

[1] Department of Computer Science, University of Brasília
dfaranha@unb.br
[2] CINVESTAV-IPN, Computer Science Department
{lfuentes,francisco}@cs.cinvestav.mx
[3] Department of Combinatorics & Optimization, University of Waterloo
edward.m.knapp@gmail.com, ajmeneze@uwaterloo.ca

Abstract. We implement asymmetric pairings derived from Kachisa-Schaefer-Scott (KSS), Barreto-Naehrig (BN), and Barreto-Lynn-Scott (BLS) elliptic curves at the 192-bit security level. Somewhat surprisingly, we find pairings derived from BLS curves with embedding degree 12 to be the fastest for our serial as well as our parallel implementations. Our serial implementations provide a factor-3 speedup over the previous state-of-the-art, demonstrating that pairing computation at the 192-bit security level is not as expensive as previously thought. We also present a general framework for deriving a Weil-type pairing that is well-suited for computing a single pairing on a multi-processor machine.

1 Introduction

Since the advent of pairing-based cryptography, researchers have been devising methods for constructing and efficiently implementing bilinear pairings. Initial work [5,12] was focused on implementing pairings at (roughly) the 80-bit security level. Koblitz and Menezes [19] highlighted the performance drawbacks of pairings at very high security levels. The subsequent discovery of Barreto-Naehrig (BN) elliptic curves [7], ideally suited for implementing pairings at the 128-bit security level, spurred a lot of research culminating in the implementation of Aranha et al. [2] that achieved speeds of under 2 million cycles for a 128-bit pairing computation on a single core of Phenom II, Core i5 and Opteron machines.

More recently, researchers have considered implementing pairings at even higher security levels. Costello, Lauter and Naehrig [9] argued that a certain family of embedding degree $k = 24$ Barreto-Lynn-Scott elliptic curves [6], henceforth called *BLS24* curves, are well-suited for implementing pairings at the 192, 224, 256, 288, and 320-bit security levels. Scott [28] implemented several pairing-based protocols using BN curves at the 128-bit security level, Kachisa-Schaefer-Scott (KSS) curves [17] with embedding degree $k = 18$ at the 192-bit security level, and BLS24 curves at the 256-bit security level. Scott concludes that the best

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 177–195, 2013.

choice of pairing to implement a particular protocol can depend on a variety of factors including the number and complexity of non-pairing operations in the protocol, the number of pairing computations that are required, and the applicability of several optimizations including fixed-argument pairings and products of pairings [27].

In this paper, we focus on fast implementations of a single pairing at the 192-bit security level. We chose the 192-bit level because it is the higher security level (the other is 128-bit) for public-key operations in the National Security Agency's Suite B Cryptography standard [23]. Moreover, as mentioned by Scott [28], the optimum choice of pairing-friendly curve for the 192-bit security level from the many available candidates [10] is not straightforward.

We examine a family of embedding degree $k = 12$ elliptic curves, henceforth called *BLS12 curves*, first proposed by Barreto, Lynn and Scott [6] (see also [8]). Unlike BN curves, the BLS12 curves are not ideal for the 128-bit security level since the group order $\#E(\mathbb{F}_p)$ is not prime. Nevertheless, our careful estimates and implementation results demonstrate that they outperform KSS, BN and BLS24 curves at the 192-bit security level. We also present a general framework for deriving analogues of the β Weil pairing, first presented in [3] for BN curves. This pairing is well-suited for computing a single pairing on a multi-processor machine since it avoids the relatively-costly final exponentiation that cannot be effectively parallelized and is present in all Tate-type pairings.

The remainder of the paper is organized as follows. The salient parameters of KSS, BN, BLS12 and BLS24 curves are presented in §2. In §3, we review Vercauteren's notion of an optimal pairing and present the β Weil pairing. The cost of the BLS12, KSS and BN pairings are estimated in §4, §5 and §6. Estimates for the BLS24 pairing are omitted due to a lack of space. Finally, §7 compares the estimated speeds of the four pairings and reports on our implementation. Our results show a significant performance improvement over the previous state-of-the-art for serial pairing implementation of the optimal ate pairing at the 192-bit security level, and an increased scalability of the β Weil pairing in relation to the optimal ate pairing.

2 Pairing-Friendly Elliptic Curves

Let p be a prime, and let E be an elliptic curve defined over the finite field $\mathbb{F}_p$. Let r be a prime with $r \mid \#E(\mathbb{F}_p)$ and $\gcd(r, p) = 1$. The *cofactor* is $\rho = \log p / \log r$. The *embedding degree* k is the smallest positive integer with $r \mid (p^k - 1)$. We will assume that k is even, whence $k > 1$ and $E[r] \subseteq E(\mathbb{F}_{p^k})$.

Let $\pi : (x, y) \mapsto (x^p, y^p)$ be the p-th power Frobenius endomorphism. The trace of the Frobenius is $t = p + 1 - \#E(\mathbb{F}_p)$. Let $\mathbb{G}_1 = \{P \in E[r] : \pi(P) = P\} = E(\mathbb{F}_p)[r]$; $\mathbb{G}_1$ is the 1-eigenspace of π acting on $E[r]$. Let d be the order of the automorphism group of E, and suppose that $d \mid k$. Let $e = k/d$ and $q = p^e$. Then there is a unique degree-d twist $\tilde{E}$ of E over $\mathbb{F}_q$ with $r \mid \#\tilde{E}(\mathbb{F}_q)$ [16]; let $\Psi : \tilde{E} \to E$ be the associated twisting isomorphism. Let $\tilde{Q} \in \tilde{E}(\mathbb{F}_q)$ be a point of order r; then $Q = \Psi(\tilde{Q}) \notin E(\mathbb{F}_p)$. The group $\mathbb{G}_2 = \langle Q \rangle$ is the p-eigenspace of

π acting on $E[r]$. Let $\mathbb{G}_T$ denote the order-r subgroup of $\mathbb{F}_{p^k}^*$. The pairings we study in this paper are non-degenerate bilinear maps from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ and are called Type 3 pairings in the literature [13].

Table 1 summarizes the salient parameters of the KSS [17], BN [7], BLS12 [6] and BLS24 [6] families of elliptic curves. All these curves are parameterized by a positive integer z, are defined by an equation of the form $Y^2 = X^3 + b$, and have a twist of order $d = 6$. Table 2 lists the important parameters of the particular KSS, BN, BLS12 and BLS24 curves that are suitable for implementing pairing-based protocols at the 192-bit security level. The requirements for this security level are that the bitlength of r be at least 384 (in order to resist Pollard's rho attack [25] on the discrete logarithm problem in $\mathbb{G}_1$), and that the bitlength of p^k should be at least 7680 (in order to resist the number field sieve attack [26] on the discrete logarithm problem in $\mathbb{F}_{p^k}^*$).

Table 1. Important parameters for the KSS, BN, BLS12 and BLS24 families

KSS curves: $k = 18$, $\rho \approx 4/3$ $p(z) = (z^8 + 5z^7 + 7z^6 + 37z^5 + 188z^4 + 259z^3 + 343z^2 + 1763z + 2401)/21$ $r(z) = (z^6 + 37z^3 + 343)/343$, $t(z) = (z^4 + 16z + 7)/7$
BN curves: $k = 12$, $\rho \approx 1$ $p(z) = 36z^4 + 36z^3 + 24z^2 + 6z + 1$ $r(z) = 36z^4 + 36z^3 + 18z^2 + 6z + 1$, $t(z) = 6z^2 + 1$
BLS12 curves: $k = 12$, $\rho \approx 1.5$ $p(z) = (z-1)^2(z^4 - z^2 + 1)/3 + z$, $r(z) = z^4 - z^2 + 1$, $t(z) = z + 1$
BLS24 curves: $k = 24$, $\rho \approx 1.25$ $p(z) = (z-1)^2(z^8 - z^4 + 1)/3 + z$, $r(z) = z^8 - z^4 + 1$, $t(z) = z + 1$

Table 2. Important parameters for the chosen KSS, BN, BLS12, BLS24 curves

Curve	b	k	z	$\lceil \log_2 p \rceil$	$\lceil \log_2 r \rceil$	ρ	$\lceil \log_2 q \rceil$	$\lceil \log_2 p^k \rceil$
KSS	2	18	$-2^{64} - 2^{51} + 2^{46} + 2^{12}$	508	376	1.35	1523	9137
BN	5	12	$2^{158} - 2^{128} - 2^{68} + 1$	638	638	1	1275	7647
BLS12	4	12	$-2^{107} + 2^{105} + 2^{93} + 2^5$	638	427	1.49	1276	7656
BLS24	4	24	$-2^{48} + 2^{45} + 2^{31} - 2^7$	477	383	1.25	1914	11482

3 Optimal Pairings

Let $R \in E(\mathbb{F}_{p^k})$ and let s be a non-negative integer. A *Miller function* $f_{s,R}$ [22] of *length* s is a function in $\mathbb{F}_{p^k}(E)$ with divisor $(f_{s,R}) = s(R) - (sR) - (s-1)(\infty)$. Note that $f_{s,R}$ is uniquely defined up to multiplication by nonzero constants in $\mathbb{F}_{p^k}$. The length s of a Miller function determines the number $\lfloor \log_2 s \rfloor$ of doubling steps, and the Hamming weight of s determines the number of addition steps in Miller's algorithm for computing $f_{s,R}$ [22]. We will always assume that Miller functions are minimally defined; that is, if $R \in E(\mathbb{F}_{p^\ell})$, then $f_{s,R}$ is selected from the function field $\mathbb{F}_{p^\ell}(E)$.

The Optimal Ate Pairing. Vercauteren's optimal pairing framework [30] allows one to compute a pairing using Miller functions each of length approximately $(1/\varphi(k)) \log r$.

For a point $R \in E[r]$ and polynomial $h = \sum h_i x^i \in \mathbb{Z}[x]$ such that $h(s) \equiv 0 \pmod{r}$, define the *extended Miller function* $f_{s,h,R}$ to be the normalized rational function with divisor

$$\sum_{i=0}^{\deg h} h_i[(s^i R) - (\infty)].$$

The *length* of the extended Miller function $f_{s,h,R}$ is the maximum of the absolute values of the h_i's. Observing that $f_{s,h_1,R} \cdot f_{s,h_2,R} = f_{s,h_1+h_2,R}$ and the polynomials $h(x) = r$, $h(x) = x^i - p^i$ satisfy the congruence condition with $s = p$, we desire elements in the following lattice which have small coefficients:

$$\left[\begin{array}{c|c} r(z) & \mathbf{0} \\ \hline \mathbf{v} & I_{\varphi(k)-1} \end{array}\right],$$

where $\mathbf{v}$ is the column vector with i-th entry $-p(z)^i$. This leads to the following result of Vercauteren's.

Theorem 1 ([30]). *There exists h such that $|h_i| \leq r^{1/\varphi(k)}$ and $(P, Q) \mapsto f_{p,h,Q}(P)^{(p^k-1)/r}$ is a pairing.*

For parameterized curves, the function $f_{p,h,Q}$ where $|h_i| \leq r^{1/\varphi(k)}$ can be computed as a product of Miller functions each having length approximately $(1/\varphi(k)) \log r$. Optimal ate pairings for KSS [30], BN [30], BLS12 [16] and BLS24 [16] curves are given in Table 3. In the table, $\ell_{S,T}$ denotes the line through points S and T.

Table 3. Optimal ate pairings

Curve	Optimal ate pairing: $(P, Q) \mapsto$	$h(x)$
KSS	$\left(f_{z,Q} \cdot f_{3,Q}^{p} \cdot \ell_{z[Q],[3p]Q}(P)\right)^{(p^{18}-1)/r}$	$z + 3x - x^4$
BN	$\left(f_{6z+2,Q} \cdot \ell_{[6z+2]Q,[p]Q} \cdot \ell_{[6z+2+p]Q,[-p^2]Q}(P)\right)^{(p^{12}-1)/r}$	$6z + 2 + x - x^2 + x^3$
BLS12	$(f_{z,Q}(P))^{(p^{12}-1)/r}$	$z - x$
BLS24	$(f_{z,Q}(P))^{(p^{24}-1)/r}$	$z - x$

The β Weil Pairing. Set $k = ed$, where d is the order of the automorphism group of E. Define w_s and $w_{s,h}$ as

$$w_s(P, Q) = \left(\frac{f_{s,Q}(P)}{f_{s,P}(Q)}\right)^{p^{k/2}-1} \quad \text{and} \quad w_{s,h}(P, Q) = \left(\frac{f_{s,h,Q}(P)}{f_{s,h,P}(Q)}\right)^{p^{k/2}-1}. \quad (1)$$

Hess [15] gave a framework for computing *optimal Weil* pairings, building on the methods of Vercauteren as expressed in Theorem 1.

Theorem 2 (Theorem 1 in [15]). *There exists h such that $|h_i| \leq r^{1/2}$ and $w_{p^e,h}$ is a pairing.*

The pairing $w_{p^e,h}$ with $|h_i| \leq r^{1/2}$ can be computed using two extended Miller functions of length approximately $\frac{1}{2}\log r$. We present a framework for constructing Weil-type pairings, called β *pairings*, which can be computed using $2e$ extended Miller functions each of length approximately $(1/\varphi(k))\log r$. In particular, we prove that for a polynomial h for which $h(p) \equiv 0 \pmod{r}$, the following is a pairing:

$$\beta\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T : (P,Q) \mapsto \prod_{i=0}^{e-1} w_{p,h}([p^i]P,Q)^{p^{e-1-i}}. \tag{2}$$

To establish that (2) is a pairing, we require a few technical lemmas, building on the work of Hess and Vercauteren. Lemma 1 gives a pairing which is the product of Weil pairings consisting of Miller functions having ate-like lengths.

Lemma 1. *For all positive integers s, the following map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ is a pairing:*

$$(P,Q) \mapsto \left(\prod_{i=0}^{e-1}\left(\frac{f_{p^s,[p^i]Q}(P)}{f_{p^s,[p^i]P}(Q)}\right)^{p^{e-1-i}}\right)^{p^{k/2}-1}.$$

Proof. It follows from Theorem 1 of [15] that the map

$$(P,Q) \mapsto \left(\frac{f_{p^e,Q}(P)}{f_{p^e,P}(Q)}\right)^{p^{k/2}-1}$$

is a pairing. Using Lemma 3(ii), one can see that

$$f_{p^e,P} = \prod_{i=0}^{e-1}(f_{p,[p^i]P})^{p^{e-1-i}}.$$

Hence, the result holds for $s = 1$.

Since

$$f_{p^s,P} = \prod_{j=0}^{s-1}(f_{p,[p^j]P})^{p^{s-1-j}},$$

we have that

$$\begin{aligned}\prod_{i=0}^{e-1}(f_{p^s,[p^i]P})^{p^{e-1-i}} &= \prod_{i=0}^{e-1}\left(\prod_{j=0}^{s-1}(f_{p,[p^i][p^j]P})^{p^{s-1-j}}\right)^{p^{e-1-i}} \\ &= \prod_{j=0}^{s-1}\left(\prod_{i=0}^{e-1}(f_{p,[p^i]([p^j]P)})^{p^{e-1-i}}\right)^{p^{s-1-j}} \\ &= \prod_{j=0}^{s-1}\left(f_{p^e,[p^j]P}\right)^{p^{s-1-j}}.\end{aligned}$$

From this, we can observe that

$$\left(\prod_{i=0}^{e-1}\left(\frac{f_{p^s,[p^i]Q}(P)}{f_{p^s,[p^i]P}(Q)}\right)^{p^{e-1-i}}\right)^{p^{k/2}-1} = \left(\prod_{i=0}^{s-1}\left(\frac{f_{p^e,[p^i]Q}(P)}{f_{p^e,[p^i]P}(Q)}\right)^{p^{s-1-i}}\right)^{p^{k/2}-1}. \quad (3)$$

By Lemma 6 of [14], the map $(P,Q) \mapsto f_{p^e,Q}(P)$ is a pairing. Thus, the right hand side of (3) is a product of pairings. □

The next lemma relates the previous pairing to the Weil pairing notation defined in (1).

Lemma 2. *The following identity holds for all positive integers s:*

$$\left(\prod_{i=0}^{e-1}\left(\frac{f_{p^s,[p^i]Q}(P)}{f_{p^s,[p^i]P}(Q)}\right)^{p^{e-1-i}}\right)^{p^{k/2}-1} = \prod_{i=0}^{e-1} w_{p^s}([p^i]P,Q)^{p^{e-1-i}}.$$

Proof. By Lemma 6 of [14], the map $(P,Q) \mapsto f_{p^s,Q}(P)$ is a pairing and so

$$\begin{aligned}\left(\prod_{i=0}^{e-1}\left(\frac{f_{p^s,[p^i]Q}(P)}{f_{p^s,[p^i]P}(Q)}\right)^{p^{e-1-i}}\right)^{p^{k/2}-1} &= \left(\prod_{i=0}^{e-1}\left(\frac{f_{p^s,Q}([p^i]P)}{f_{p^s,[p^i]P}(Q)}\right)^{p^{e-1-i}}\right)^{p^{k/2}-1} \\ &= \prod_{i=0}^{e-1} w_{p^s}([p^i]P,Q)^{p^{e-1-i}}.\end{aligned}$$

□

Finally, using the pairing relation from Lemma 2, we can obtain a pairing composed of Miller functions each with Vercauteren-style bound on the length.

Theorem 3. *There exists h such that* $|h_i| \leq r^{1/\varphi(k)}$ *and the following is a pairing:*

$$\beta\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T\ :\ (P,Q) \mapsto \prod_{i=0}^{e-1} w_{p,h}([p^i]P,Q)^{p^{e-1-i}}.$$

Proof. Let $h(x) = \sum_{i=0}^{c} h_i x^i$ be given by Vercauteren's theorem and let $h(p) = rm$. Since

$$f_{r,P}^{m} = f_{p,h,P} \cdot \prod_{j=0}^{c} f_{p^j,P}^{h_j},$$

we have that

$$w_r(P,Q)^m = w_{p,h}(P,Q) \cdot \prod_{j=0}^{c} w_{p^j}(P,Q)^{h_j}.$$

Hence

$$\begin{aligned}\prod_{i=0}^{e-1} w_{p,h}([p^i]P,Q)^{p^{e-1-i}} &= \prod_{i=0}^{e-1} \left(w_r([p^i]P,Q)^m \cdot \prod_{j=0}^{c} w_{p^j}([p^i]P,Q)^{-h_j} \right)^{p^{e-1-i}} \\ &= \prod_{i=0}^{e-1} w_r([p^i]P,Q)^{mp^{e-1-i}} \cdot \prod_{j=0}^{c} \left(\prod_{i=0}^{e-1} w_{p^j}([p^i]P,Q)^{p^{e-1-i}} \right)^{-h_j},\end{aligned}$$

which by Lemmas 1 and 2 is a product of pairings. □

Using Theorem 3 and the polynomials h from Table 3, we found that the β Weil pairings for BN, BLS12, KSS and BLS24 curves can be defined as follows:

$$\text{KSS} : (P,Q) \mapsto \left[\left(\frac{f_{p,h,P}(Q)}{f_{p,h,Q}(P)} \right)^{p^2} \left(\frac{f_{p,h,[p]P}(Q)}{f_{p,h,Q}([p]P)} \right)^{p} \frac{f_{p,h,[p^2]P}(Q)}{f_{p,h,Q}([p^2]P)} \right]^{(p^9-1)(p^3+1)}, \tag{4}$$

$$\text{BN} : (P,Q) \mapsto \left[\left(\frac{f_{p,h,P}(Q)}{f_{p,h,Q}(P)} \right)^{p} \frac{f_{p,h,[p]P}(Q)}{f_{p,h,Q}([p]P)} \right]^{(p^6-1)(p^2+1)}, \tag{5}$$

$$\text{BLS12} : (P,Q) \mapsto \left[\left(\frac{f_{z,P}(Q)}{f_{z,Q}(P)} \right)^{p} \frac{f_{z,[p]P}(Q)}{f_{z,Q}([p]P)} \right]^{(p^6-1)(p^2+1)}, \tag{6}$$

$$\text{BLS24} : (P,Q) \mapsto \left[\frac{f_{z,P}^{p^3}(Q) \cdot f_{z,[p]P}^{p^2}(Q) \cdot f_{z,[p^2]P}^{p}(Q) \cdot f_{z,[p^3]P}(Q)}{f_{z,Q}^{p^3}(P) \cdot f_{z,Q}^{p^2}([p]P) \cdot f_{z,Q}^{p}([p^2]P) \cdot f_{z,Q}([p^3]P)} \right]^{(p^{12}-1)(p^4+1)}. \tag{7}$$

For all four β Weil pairings, computing $[p]P$ has approximately the same cost as computing $[z]P$.

Parallelization of Pairings. Given two processors, the Weil pairing can be trivially parallelized since the numerator and denominator of the Weil pairing are independent operations. The ate pairing requires two serial operations, the Miller loop and the final exponentiation. The next lemma can be used to parallelize the computation of the Miller loop. We know of no way to parallelize the final exponentiation.

Lemma 3. *Let a and b be non-negative integers, and let $R \in E(\mathbb{F}_{q^k})$. Then (i) $f_{a+b,R} = f_{a,R} \cdot f_{b,R} \cdot \ell_{[a]R,[b]R}/v_{[a+b]R}$, where v_P denotes the vertical line through P; and (ii) $f_{ab,R} = f_{b,R}^{a} \cdot f_{a,[b]R}$.*

The method of Aranha et al. [4] for parallelizing the computation of a Miller function $f_{s,R}$ is the following. We first write $s = 2^w s_1 + s_0$ with $s_0 < 2^w$. Applying Lemma 3, we obtain

$$f_{s,R} = f_{s_1,R}^{2^w} \cdot f_{2^w,[s_1]R} \cdot f_{s_0,R} \cdot \ell_{[2^w s_1]R,[s_0]R}/v_{[s]R}. \tag{8}$$

If s_0 is small, then the Miller function $f_{s_0,R}$ can be computed relatively cheaply. Thus the computation of $f_{s,R}$ can be parallelized by computing $f_{s_1,R}^{2^w}$ on one processor and $f_{2^w,[s_1]R}$ on a second processor. The parameter w should be carefully selected in order to balance the time of the two function computations. The relevant criteria for selecting w include the Hamming weight of s_1 (which determines the number of additions in the Miller loop for the first function), and the cost of the w-fold squaring in the first function relative to the cost of computing $s_1 R$ in the second function. This idea can be extended to c processors by writing $s = 2^{w_{c-1}} s_{c-1} + \cdots + 2^{w_1} s_1 + s_0$.

Remark 1. (unsuitability of composite-order BN curves) Consider a BN curve at the 192-bit security level. For such a curve, we desire a (sparse) BN parameter z of approximately 160 bits. From the optimal pairing framework, we choose a suitable vector $[2z, z+1, -z, z]$ corresponding to the following pairing (with the final exponentiation omitted):

$$(P,Q) \mapsto f_{2z,Q} \cdot f_{z+1,Q}^{p} \cdot f_{z,Q}^{-p^2} \cdot f_{z,Q}^{p^3} \cdot \ell_{[-zp^2]Q,[zp^3]Q} \cdot \ell_{[p(z+1)]Q,[-zp^2+zp^3]Q}(P).$$

Computation of the lines is relatively inexpensive. However, at first, it appears one must evaluate multiple Miller functions. Fortunately, for parameterized curves, one can (usually) rearrange terms such that the computational bottleneck is $f_{z,Q}$ with only a few lines comprising the remaining computation. In the above case, we obtain

$$(P,Q) \mapsto f_{z,Q}^{2+p-p^2+p^3} \cdot \ell_{[z]Q,[z]Q} \cdot \ell_{[zp]Q,[p]Q} \cdot \ell_{[-zp^2]Q,[zp^3]Q} \cdot \ell_{[p(z+1)]Q,[-zp^2+zp^3]Q}(P).$$

At the 192-bit security level, we require that r have a prime divisor of at least 384 bits. We can easily choose r to be (a 640-bit) prime. However, given that the optimal pairing framework gives a maximum Miller length of around $(\log n)/4$ for BN curves where n is a large prime divisor of r, we should be tempted to choose r with a 384-bit prime divisor. The fact that the coordinates of the vector $[2z, z+1, -z, z]$ have small coefficients when written in base z allowed us to write the pairings as a power of $f_{z,Q}$ multiplied by a few lines. However, for composite values of r, the vector with 96-bit elements which we obtain from the optimal pairing framework does not, in general, have coordinates which we can relate to each other. We would therefore require approximately 4 independent Miller functions, negating most of the benefit of computing an optimal pairing, rather

than the Tate pairing. The possibility of choosing a vector whose elements are part of a short addition chain may still exist but the vectors produced by the LLL algorithm [21] do not appear to maintain such structure. Thus, composite-order BN curves would appear to yield inferior performance compared to prime-order BN curves.

4 BLS12 Pairings

In this section, we consider the BLS12 curve $Y^2 = X^3 + 4$ defined with the parameter selection $z = -2^{107} + 2^{105} + 2^{93} + 2^5$ which yields a 638-bit prime p and a 427-bit prime r.

Extension Field Arithmetic for Pairings with k = 12. A tower extension for $\mathbb{F}_{p^{12}}$ can be constructed as follows:

$$\begin{aligned}\mathbb{F}_{p^2} &= \mathbb{F}_p[u]/(u^2 - \beta), \text{ where } \beta \in \mathbb{F}_p,\\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[v]/(v^3 - \xi), \text{ where } \xi \in \mathbb{F}_{p^2}, \text{ and}\\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[w]/(w^2 - \gamma), \text{ where } \gamma \in \mathbb{F}_{p^6}.\end{aligned}$$

For our choice of parameters, we have the optimal $\beta = -1$, $\xi = u + 1$, $\gamma = v$. Table 4 gives the computational costs of the tower extension field arithmetic for curves with $k = 12$ in terms of a 640-bit multiplication (m_{640}) and inversion (i_{640}) in $\mathbb{F}_p$, with p a 638-bit prime.[1] The cost of additions is ignored because of their lower overall performance impact due to the larger field size in comparison with [2,24]. Moreover, $\tilde{m}$, $\tilde{s}$, $\tilde{i}$ denote the cost of multiplication, squaring and inversion in $\mathbb{F}_{p^2}$ respectively.[2] $\mathbb{G}_{\Phi_6(p^2)}$ denotes the order-$\Phi_6(p^2)$ subgroup of $\mathbb{F}^*_{p^{12}}$, where Φ_k denotes the k-th cyclotomic polynomial.

Miller Loop. For the parameter selection $z = -2^{107} + 2^{105} + 2^{93} + 2^5$, the Miller loop computation of $f_{z,Q}$ requires 107 point doublings and associated line evaluations, 3 point additions with line evaluations, 109 sparse multiplications, and 106 squarings in $\mathbb{F}_{p^{12}}$. The computational costs of these operations can be found in [2, Table 1]. We obtain a BLS12 Miller loop cost of $107(3\tilde{m} + 6\tilde{s} + 4m_{640}) + 3(11\tilde{m} + 2\tilde{s} + 4m_{640}) + 109(13\tilde{m}) + 106(12\tilde{m}) = 3043\tilde{m} + 648\tilde{s} + 440m_{640} = 10865m_{640}$.

Final Exponentiation. The final exponentiation consists of raising the Miller loop result $f \in \mathbb{F}_{p^k}$ to the $e = (p^k - 1)/r$-th power. This task can be broken into two parts since

$$e = (p^k - 1)/r = [(p^k - 1)/\Phi_k(p)] \cdot [\Phi_k(p)/r].$$

Computing $f^{(p^k-1)/\Phi_k(p)}$ is considered easy, costing only a few multiplications, inversions, and inexpensive p-th powerings in $\mathbb{F}_{p^k}$. Raising to the power $d =$

[1] In the case of software implementation, this selection of the size of p facilitates the usage of *lazy reduction* techniques as recommended in [2,24].

[2] For further details on how these costs were deduced, the reader is referred to [2,24].

Table 4. Costs of arithmetic operations in a tower extension field $\mathbb{F}_{p^{12}}$

Field	Mult.	Squaring	Inversion
$\mathbb{F}_{p^2}$	$\tilde{m} = 3m_{640}$	$\tilde{s} = 2m_{640}$	$\tilde{i} = 4m_{640} + i_{640}$
$\mathbb{F}_{p^6}$	$6\tilde{m}$	$\tilde{m} + 4\tilde{s}$	$9\tilde{m} + 3\tilde{s} + \tilde{i}$
$\mathbb{F}_{p^{12}}$	$18\tilde{m}$	$12\tilde{m}$	$23\tilde{m} + 11\tilde{s} + \tilde{i}$
$\mathbb{G}_{\Phi_6(p^2)}$	$18\tilde{m}$	$9\tilde{s}$	Conjugation

Operation	Cost
Sparse Mult.	$13\tilde{m}$
Sparser Mult.	$7\tilde{m}$
Compressed Squaring	$6\tilde{s}$
Simult. decompression of n field elements	$n(3\tilde{m} + 3\tilde{s}) + (n-1)3\tilde{m} + \tilde{i}$
$p/p^2/p^3$-Frobenius	$10m/15m/15m$

$\Phi_k(p)/r$ is a more challenging task. Observing that p-th powering is much less expensive than multiplication, Scott et al. [29] give a systematic method for reducing the expense of exponentiating by d. In the case of BLS12 curves, it can be shown that the exponent d can be written as $d = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \lambda_3 p^3$ where $\lambda_0 = z^5 - 2z^4 + 2z^2 - z + 3$, $\lambda_1 = z^4 - 2z^3 + 2z - 1$, $\lambda_2 = z^3 - 2z^2 + z$, and $\lambda_3 = z^2 - 2z + 1$. The exponentiation f^d can be computed using the following addition-subtraction chain:

$$f \to f^{-2} \to f^z \to f^{2z} \to f^{z-2} \to f^{z^2-2z} \to f^{z^3-2z^2} \to f^{z^4-2z^3} \\ \to f^{z^4-2z^3+2z} \to f^{z^5-2z^4+2z^2},$$

which requires 5 exponentiations by z, 2 multiplications in $\mathbb{F}_{p^{12}}$, and 2 cyclotomic squarings. This allows f^d to be computed as

$$f^d = f^{z^5-2z^4+2z^2} \cdot (f^{z-2})^{-1} \cdot f \cdot (f^{z^4-2z^3+2z} \cdot f^{-1})^p \cdot (f^{z^3-2z^2} \cdot f^z)^{p^2} \cdot (f^{z^2-2z} \cdot f)^{p^3},$$

which requires an additional 8 multiplications in $\mathbb{F}_{p^{12}}$ and 3 Frobenius maps. This implies that the hard part of the final exponentiation requires 2 cyclotomic squarings, 5 exponentiations by z, 10 multiplications in $\mathbb{F}_{p^{12}}$, and 3 Frobenius maps.

In total, the cost of computing the final exponentiation is 1 inversion in $\mathbb{F}_{p^{12}}$, 2 cyclotomic squarings, 12 multiplications in $\mathbb{F}_{p^{12}}$, 4 Frobenius maps, and 5 exponentiations by z. It can be shown that exponentiation by our choice of the z parameter requires 107 compressed squarings, simultaneous decompression of 4 field elements, and 3 multiplications in $\mathbb{F}_{p^{12}}$ when Karabina's exponentiation technique [18] is employed. The cost of an exponentiation by z is $107(6\tilde{s}) + 4(3\tilde{m} + 3\tilde{s}) + 3(3\tilde{m}) + \tilde{i} + 3(18\tilde{m}) = 75\tilde{m} + 654\tilde{s} + \tilde{i}$, whence the total cost of the final exponentiation is $(23\tilde{m} + 11\tilde{s} + \tilde{i}) + 2(9\tilde{s}) + 12(18\tilde{m}) + 60m_{640} + 5(75\tilde{m} + 654\tilde{s} + \tilde{i}) = 614\tilde{m} + 3299\tilde{s} + 6\tilde{i} = 8464m_{640} + 6i_{640}$.

Optimal Pairing Cost. From the above, we conclude that the estimated cost of the optimal ate pairing for our chosen BLS12 curve is $10865m_{640}+8464m_{640}+6i_{640} = 19329m_{640} + 6i_{640}$.

Parallelization. Figure 1 illustrates the execution path for the β Weil pairing (6) when the four Miller functions are computed in parallel using 4 processors. As with the optimal ate pairing, Lemma 3 was repeatedly applied to each Miller function in the β Weil pairing in order to obtain a parallel implementation using 8 processors.

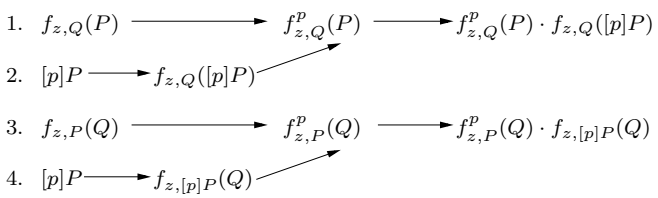


Fig. 1. Execution path for computing the β Weil pairing for BLS12 curves on 4 processors

5 KSS Pairings

In this section, we consider the KSS curve $Y^2 = X^3 + 2$ defined with the parameter selection $z = -2^{64} - 2^{61} + 2^{46} + 2^{12}$.

Extension Field Arithmetic for Pairings with k = 18. An element in $\mathbb{F}_{p^{18}}$ can be represented using the following towering scheme:

$$\begin{aligned}\mathbb{F}_{p^3} &= \mathbb{F}_p[u]/(u^3+2),\\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^3}[v]/(v^2-u),\\ \mathbb{F}_{p^{18}} &= \mathbb{F}_{p^6}[w]/(w^3-v).\end{aligned}$$

Table 5 gives the computational costs of the tower extensions field arithmetic for curves with $k = 18$, where m_{512}, i_{512} denote the cost of multiplication and inversion in $\mathbb{F}_p$, with p a 512-bit prime. Moreover, $\hat{m}$, $\hat{s}$, $\hat{\imath}$ denote the cost of multiplication, squaring and inversion in $\mathbb{F}_{p^3}$ respectively.

Computation of the Optimal Ate Pairing. For the parameter selection $z = -2^{64} - 2^{51} + 2^{46} + 2^{12}$, the Miller loop executes 64 point doublings with line evaluations, 4 point additions with line evaluations, 67 sparse multiplications and 63 squarings in $\mathbb{F}_{p^{18}}$. We obtain a KSS Miller loop cost of $64(3\hat{m}+6\hat{s}+6m_{512})+4(11\hat{m}+2\hat{s}+6m_{512})+67(13\hat{m})+63(11\hat{m}) = 1800\hat{m}+392\hat{s}+408m_{512} = 13168m_{512}$. Furthermore, the final step executes 1 squaring in $\mathbb{F}_{p^{18}}$, one p-power Frobenius, 1 multiplication in $\mathbb{F}_{p^{18}}$, 2 point additions with line evaluation, one point doubling with line evaluation, 1 sparse multiplication, 1 sparser multiplication,

Table 5. Costs of arithmetic operations in a tower extension field $\mathbb{F}_{p^{18}}$

Field	Mult.	Squaring	Inversion
$\mathbb{F}_{p^3}$	$\hat{m} = 6m_{512}$	$\hat{s} = 5m_{512}$	$\hat{i} = 12m_{512} + i_{512}$
$\mathbb{F}_{p^6}$	$3\hat{m}$	$2\hat{m}$	$2\hat{m} + 2\hat{s} + \hat{i}$
$\mathbb{F}_{p^{18}}$	$18\hat{m}$	$11\hat{m}$	$20\hat{m} + 8\hat{s} + \hat{i}$
$\mathbb{G}_{\varphi_6}(\mathbb{F}_{p^3})$	$18\hat{m}$	$6\hat{m}$	Conjugation

Operation	Cost
Sparse Mult.	$13\hat{m}$
Sparser Mult.	$7\hat{m}$
Compressed Squaring	$4\hat{m}$
Simult. decompression of n field elements	$n(3\hat{m} + 3\hat{s})+$ $(n-1)3\hat{m} + \hat{i}$
pth-Frobenius	$15m$

and the computation of the isomorphism $\psi(Q)$. Thus the KSS final step cost is $11\hat{m} + 18\hat{m} + 2(11\hat{m} + 2\hat{s} + 6m_{512}) + 3\hat{m} + 6\hat{s} + 6m_{512} + 20\hat{m} + 28m_{512} = 74\hat{m} + 10\hat{s} + 40m_{512} = 534m_{512}$. The final exponentiation executes in total one inversion in $\mathbb{F}_{p^{18}}$, 8 cyclotomic squarings, 54 multiplications in $\mathbb{F}_{p^{18}}$, 29 p-power Frobenius, and 7 exponentiations by z [11]. The computational cost of an exponentiation by z is 64 compressed squarings, decompression of 4 field elements and 3 multiplications in $\mathbb{F}_{p^{18}}$, for a total cost of $64(6\hat{s}) + 4(3\hat{s} + 3\hat{m}) + 9\hat{m}+\hat{i}+3(18\hat{m}) = 75\hat{m}+396\hat{s}+\hat{i}$. Hence, the total cost of the final exponentiation is $20\hat{m} + 8\hat{s} + \hat{i} + 8(6\hat{m}) + 54(18\hat{m}) + 435m_{512} + 7(75\hat{m} + 396\hat{s} + \hat{i}) = 1565\hat{m} + 2780\hat{s} + 8\hat{i} + 435m_{512} = 23821m_{512} + 8i_{512}$ Finally, the total cost of computing the KSS optimal ate pairing is $13168m_{512} + 534m_{512} + 23821m_{512} + 8i_{512} = 37523m_{512} + 8i_{512}$.

Computation of the β Weil Pairing. The most expensive part of the β Weil pairing for KSS curves (4) are the six Miller functions $f_{z,R}$. For parallel implementation using 4 cores, repeated applications of Lemma 3 can be used to write $z = 2^w z_1 + z_0$ such that $f_{z,R}$ can be computed in the following way:

$$f_{z,R} = f_{z_1,R}^{2^w} \cdot f_{2^w,[z_1]R} \cdot f_{z_0,R} \cdot (\ell_{2^w} \cdot [z_1]R, [z_0]R)/v_{[z]R}.$$

For the KSS parameter $z = -2^{64} - 2^{51} + 2^{46} + 2^{12}$, we chose $w = 36$, $z_1 = -2^{28} + 2^{15} + 2^{10}$, $z_0 = 2^{12}$ and split the two most expensive Miller functions $f_{z,Q}^p([p]P)$ and $f_{z,Q}([p^2]P)$. Figure 2 illustrates an execution path. At the end, it is necessary for each core to compute the additional functions $(f_{3,R}^p \cdot \ell_{[z]R,[3p]R})^{p^i}$ and the exponentiation by $(p^9 - 1) \cdot (p^3 + 1)$.

For the case of an 8-core implementation, we simply reschedule these functions so that each core takes approximately the same time.

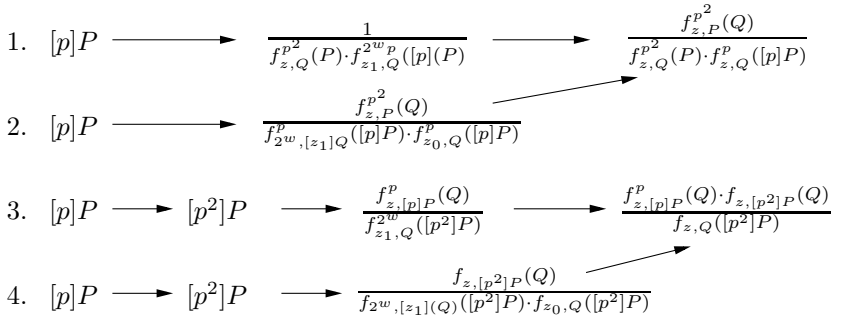


Fig. 2. Execution path for computing the β Weil pairing for KSS curves on 4 processors

6 BN Pairings

In this section, we consider the BN curve $Y^2 = X^3+5$ defined with the parameter selection $z = 2^{158} - 2^{128} - 2^{68} + 1$. The extension fields are $\mathbb{F}_{p^2} = \mathbb{F}_p[u]/(u^2+1)$, $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3-\xi)$ with $\xi = u+2$, and $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2-v)$.

Computation of the Optimal Ate Pairing. The Miller loop executes 160 point doublings with line evaluations, 6 point additions with line evaluations, 164 sparse multiplications, 1 sparser multiplication and 159 squarings in $\mathbb{F}_{p^{12}}$. We obtain a BN Miller loop cost of $160(3\tilde{m}+6\tilde{s}+4m_{640})+6(11\tilde{m}+2\tilde{s}+4m_{640})+164(13\tilde{m})+7\tilde{m}+159(12\tilde{m}) = 4593\tilde{m}+972\tilde{s}+664m_{640} = 16387m_{640}$.

Furthermore, the final step executes $\psi(Q)$, $\psi^2(Q)$, 2 point additions with line evaluation, 1 sparser multiplication and 1 multiplication in $\mathbb{F}_{p^{12}}$. The p-th power Frobenius can be computed at a cost of about $5m_{640}$ and the p^2-th power Frobenius can be computed at a cost of about $4m_{640}$. Thus the BN final step cost is $2(11\tilde{m}+2\tilde{s}+4m_{640})+7\tilde{m}+18\tilde{m}+9m_{640} = 47\tilde{m}+4\tilde{s}+17m_{640} = 166m_{640}$. The final exponentiation executes in total 1 inversion in $\mathbb{F}_{p^{12}}$, 3 cyclotomic squarings, 12 multiplications in $\mathbb{F}_{p^{12}}$, 2 p-th power Frobenius, 1 p^2-th power Frobenius, 1 p^3-th power Frobenius, and 3 exponentiations by z [11]. The computational cost of an exponentiation by z is: 158 compressed squarings, decompression of 3 field elements and 3 multiplications in $\mathbb{F}_{p^{12}}$, for a total cost of $158(6\tilde{s}) + 3(3\tilde{s}+3\tilde{m}) + 6\tilde{m} + \tilde{i} + 3(18\tilde{m}) = 69\tilde{m} + 957\tilde{s} + \tilde{i}$. Hence, the total cost of the final exponentiation is $23\tilde{m}+11\tilde{s}+\tilde{i}+3(9\tilde{s})+12(18\tilde{m})+50m_{640}+3(69\tilde{m}+957\tilde{s}+\tilde{i}) = 446\tilde{m}+2909\tilde{s}+62m_{640}+4i_{640} = 7218m_{640}+4i_{640}$. Finally, the total cost of computing the BN optimal ate pairing is $16387m_{640}+166m_{640}+7218m_{640}+4i_{640} = 23771m_{640}+4i_{640}$.

Computation of the β Weil Pairing. For BN curves, we consider the β pairing presented by Aranha et al. [3]. Lemma 3 was repeatedly applied in order to estimate the cost of a parallel implementation using 8 processors.

7 Comparisons

Estimates for Serial Implementations of the Optimal Ate Pairings. The customary way to estimate the cost of a pairing is to count multiplications in the underlying finite fields. Notice that in the case of software implementations in modern desktop platforms, field elements $a \in \mathbb{F}_p$ can be represented with $\ell = 1 + \lfloor \log_2(p) \rfloor$ binary coefficients a_i packed in $n_{64} = \lceil \frac{\ell}{64} \rceil$ 64-bit processor words. If Montgomery representation is used to implement field multiplication in $\mathbb{F}_{p_{640}}$ and $\mathbb{F}_{p_{512}}$ with complexity $O(2n_{64}^2 + n_{64})$, then it is reasonable to estimate that we have $m_{640} \approx (210/136) \cdot m_{512} \approx 1.544 \cdot m_{512}$.

Table 6 summarizes the costs in terms of finite field multiplications for computing the optimal ate pairing over our choice of KSS, BN, BLS12 and BLS24 curves at the 192-bit security level.[3] As can be seen, our estimates predict that the optimal ate pairing over BLS12 curves is the most efficient choice at the 192-bit security level, with KSS, BN and BLS24 curves being significantly slower. The main computational bottleneck for BLS24 curves is their very expensive final exponentiation.

Table 6. Cost estimates of the optimal ate pairing for KSS, BN, BLS12 and BLS24 curves at the 192-bit security level. Note that $m_{480} = m_{512}$ in a 64-bit processor

Curve	Phase	Mult. in $\mathbb{F}_p$	Mult. in $\mathbb{F}_{p_{512}}$
KSS	Miller Loop	$13168m_{512}$	$13168m_{512}$
	Final Step	$534m_{512}$	$534m_{512}$
	Final Exp.	$23821m_{512}$	$23821m_{512}$
	ML + FS + FE	$37523m_{512}$	$37523m_{512}$
BN	Miller Loop	$16387m_{640}$	$25301m_{512}$
	Final Step	$166m_{640}$	$256m_{512}$
	Final Exp.	$7218m_{640}$	$11145m_{512}$
	ML + FS + FE	$23771m_{640}$	$36702m_{512}$
BLS12	Miller Loop	$10865m_{640}$	$16775m_{512}$
	Final Exp.	$8464m_{640}$	$13068m_{512}$
	ML + FE	$19329m_{640}$	$29843m_{512}$
BLS24	Miller Loop	$14927m_{480}$	$14927m_{512}$
	Final Exp.	$25412m_{480}$	$25412m_{512}$
	ML + FE	$40339m_{480}$	$40339m_{512}$

Estimates for Multi-core Implementations of the Optimal Ate and β Weil Pairings. Table 7 (see also Figure 3) shows estimated speedups for the parallel version of the optimal ate pairing using the partitions in Table 8 and all the β Weil pairing variants considered here. All speedup factors are with respect to the serial version of the KSS optimal ate pairing. It can be seen that

[3] In the case of BN and KSS curves it is necessary to compute several extra lines and Frobenius maps. We refer to these steps as the "Final step". We stress that there is no analogous final step in the case of BLS12 and BLS24 curves.

the estimated performance for BLS12 curves when using 8 cores is of a factor-3.29 acceleration, which is the highest speedup we obtain. Perhaps the most notable observation from Table 7 is that, for eight-core implementations, the β Weil pairing becomes more efficient than the optimal ate pairing for all the four curves considered.

Table 7. Estimated speedups for the parallel version of the optimal ate pairing versus the β Weil pairing. All speedup factors are with respect to the serial version of the KSS optimal ate pairing.

	Number of threads			
Estimated speedup KSS	**1**	**2**	**4**	**8**
Optimal ate	1.00	1.17	1.28	1.33
β Weil	0.47	0.91	1.54	2.51
Estimated speedup BN	**1**	**2**	**4**	**8**
Optimal ate	1.02	1.36	1.61	1.76
β Weil	0.41	0.81	1.42	2.16
Estimated speedup BLS12	**1**	**2**	**4**	**8**
Optimal ate	1.26	1.56	1.76	1.88
β Weil	0.64	1.25	2.20	3.29
Estimated speedup BLS24	**1**	**2**	**4**	**8**
Optimal ate	0.93	1.05	1.12	1.14
β Weil	0.40	0.78	1.49	2.39

Table 8. Parameters w_i, $0 < i < c$, which define the partition of the form $s = 2^{w_{c-1}} s_{c-1} + \cdots + 2^{w_1} s_1 + s_0$ for splitting the Miller loop according to Equation (8) when computing a multi-thread optimal ate pairing among c processing units

	Number of threads (c)		
Curve	2	4	8
KSS	36	54, 39, 21	63, 58, 52, 45, 36, 26, 14
BN	86	129, 93, 50	149, 137, 122, 105, 85, 61, 33
BLS12	57	85, 61, 33	98, 90, 81, 70, 56, 40, 21
BLS24	26	38, 28, 15	44, 41, 37, 32, 26, 19, 10

Timings. We implemented the KSS, BN, BLS12 and BLS24 pairings following the techniques described in [2] on two different 64-bit 32nm platforms, an Intel Core i5 540M Nehalem and an Intel Core i7 2630QM Sandy Bridge. Field arithmetic was implemented in Assembly for maximum efficiency and high-level code was implemented in the C programming language. The GCC 4.7.0 compiler suite was used with compilation flags for loop unrolling, inlining of small functions to reduce function call overheads, and optimization level `-O3`. The implementation was done on top of the RELIC cryptographic toolkit [1]. The code will eventually be incorporated into the library.

Table 9. Experimental results for serial/parallel executions of the KSS, BN and BLS12 optimal ate and β Weil pairings. Timings are presented in millions of clock cycles. The speedups are with respect to the serial version of the KSS optimal ate pairing. The columns marked with (*) present estimates based on per-thread data.

	Number of threads			
Platform 1 – Intel Core i5 Nehalem 32nm	**1**	**2**	**4***	**8***
KSS optimal ate – latency	23.40	20.91	19.75	19.17
KSS optimal ate – speedup	1.00	1.12	1.18	1.22
KSS β Weil – latency	–	–	15.04	9.18
KSS β Weil – speedup	–	–	1.56	2.55
BN optimal ate – latency	23.22	17.28	14.63	13.40
BN optimal ate – speedup	1.01	1.35	1.59	1.73
BN β Weil – latency	–	–	16.65	11.17
BN β Weil – speedup	–	–	1.39	2.08
BLS12 optimal ate – latency	18.67	15.15	13.49	12.58
BLS12 optimal ate – speedup	1.25	1.54	1.73	1.86
BLS12 β Weil – latency	–	19.38	10.80	7.24
BLS12 β Weil – speedup	–	1.21	2.17	3.23
BLS24 optimal ate – latency	26.32	24.00	22.82	22.27
BLS24 optimal ate – speedup	0.89	0.98	1.03	1.05
BLS24 β Weil – latency	–	–	17.83	10.26
BLS24 β Weil – speedup	–	–	1.31	2.28
Platform 2 – Intel Core i7 Sandy Bridge 32nm	**1**	**2**	**4**	**8***
KSS optimal ate – latency	17.73	15.76	14.95	14.52
KSS optimal ate – speedup	1.00	1.12	1.19	1.22
KSS β Weil – latency	–	–	11.36	6.97
KSS β Weil – speedup	–	–	1.56	2.54
BN optimal ate – latency	17.43	13.00	10.98	10.05
BN optimal ate – speedup	1.02	1.36	1.61	1.76
BN β Weil – latency	–	–	12.58	8.45
BN β Weil – speedup	–	–	1.41	2.10
BLS12 optimal ate – latency	14.08	11.41	10.11	9.48
BLS12 optimal ate – speedup	1.26	1.55	1.75	1.87
BLS12 β Weil – latency	–	14.58	8.13	5.47
BLS12 β Weil – speedup	–	1.22	2.18	3.24
BLS24 optimal ate – latency	19.97	18.27	17.21	16.86
BLS24 optimal ate – speedup	0.89	0.97	1.03	1.05
BLS24 β Weil – latency	–	–	13.75	7.90
BLS24 β Weil – speedup	–	–	1.29	2.24

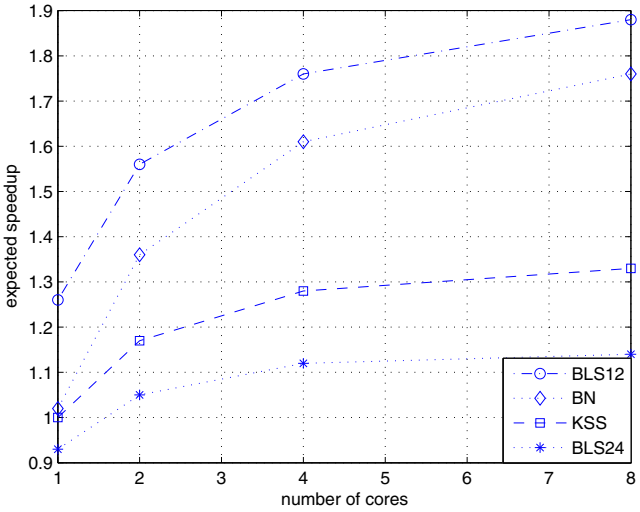


Fig. 3. Expected speedups for KSS, BN, BLS12 and BLS24 optimal ate pairings at the 192-bit security level. All speedup factors are with respect to the serial version of the KSS optimal ate pairing.

The $m_{640} \approx 1.544 \cdot m_{512}$ estimate used above was experimentally confirmed with carefully crafted Assembly code for multiplication and Montgomery reduction. Implementing the double-precision arithmetic needed for efficient application of lazy reduction proved to be slightly cumbersome due to the exhaustion of the 16 general-purpose registers available in the target platform (one of the registers is mostly reserved for keeping track of stack memory, aggravating the effect). Naturally, this issue had a bigger performance impact on the larger 638-bit field, introducing higher penalties for reading and writing values stored into memory. By using a very efficient implementation of the Extended Euclidean Algorithm imported from the GMP[4] library, we obtained inversion-to-multiplication ratios in $\mathbb{F}_p$ of around 16, suggesting the use of the projective coordinate system instead of the affine coordinates recommended in [28] and [20], even after considering the action of the norm map to simplify the inversion operation in extension fields. Affine coordinates were only competitive for the BLS24 curve.

The resulting timings for the two platforms are presented in Table 9 (measured with the Turbo Boost feature disabled). Timings for the parallel implementation of pairings which were estimated to be slower than the reference performance of the KSS pairing are omitted. We obtained results confirming our performance estimates, i.e., the BLS12 curve is the most efficient choice for pairing computation at the 192-bit security level across all the considered scenarios. In particular, our fastest serial implementation on the Intel Core i5 Nehalem machine can compute a pairing in approximately 19 million cycles, more than 3 times faster than the current state-of-the-art. The previous speed record for a single pairing computation without precomputation at this security level was presented in [28, Table

[4] GUN Multiple Precision Arithmetic Library: `http://www.gmplib.org`

2, column 4 halved] and achieves a latency of 60 million cycles on a very similar machine when a factor of 1.22 is applied to the timings to adjust for the effect of Turbo Boost.[5] Additionally, the β Weil pairing presents itself as the most efficient and scalable choice of pairing in a multiprocessor machine with more than 4 processing units.

References

1. Aranha, D.F., Gouvêa, C.P.L.: RELIC is an Efficient LIbrary for Cryptography, `http://code.google.com/p/relic-toolkit/`
2. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster Explicit Formulas for Computing Pairings over Ordinary Curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (2011)
3. Aranha, D.F., Knapp, E., Menezes, A., Rodríguez-Henríquez, F.: Parallelizing the Weil and Tate Pairings. In: Chen, L. (ed.) Cryptography and Coding 2011. LNCS, vol. 7089, pp. 275–295. Springer, Heidelberg (2011)
4. Aranha, D.F., López, J., Hankerson, D.: High-Speed Parallel Software Implementation of the η_T Pairing. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 89–105. Springer, Heidelberg (2010)
5. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
6. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing Elliptic Curves with Prescribed Embedding Degrees. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 257–267. Springer, Heidelberg (2003)
7. 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)
8. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Designs, Codes and Cryptography 37, 133–141 (2006)
9. Costello, C., Lauter, K., Naehrig, M.: Attractive Subfamilies of BLS Curves for Implementing High-Security Pairings. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 320–342. Springer, Heidelberg (2011)
10. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology 23, 224–280 (2010)
11. Fuentes-Castañeda, L., Knapp, E., Rodríguez-Henríquez, F.: Faster Hashing to $\mathbb{G}_2$. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 412–430. Springer, Heidelberg (2012)
12. Galbraith, S.D., Harrison, K., Soldera, D.: Implementing the Tate Pairing. In: Fieker, C., Kohel, D.R. (eds.) ANTS 2002. LNCS, vol. 2369, pp. 324–337. Springer, Heidelberg (2002)
13. Galbraith, S., Paterson, K., Smart, N.: Pairings for cryptographers. Discrete Applied Mathematics 156, 3113–3121 (2008)
14. Granger, R., Hess, F., Oyono, R., Thériault, N., Vercauteren, F.: Ate Pairing on Hyperelliptic Curves. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 430–447. Springer, Heidelberg (2007)

[5] This was confirmed with the author via private communication.

15. Hess, F.: Pairing Lattices. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 18–38. Springer, Heidelberg (2008)
16. Hess, F., Smart, N., Vercauteren, F.: The eta pairing revisited. IEEE Transactions on Information Theory 52, 4595–4602 (2006)
17. Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 126–135. Springer, Heidelberg (2008)
18. Karabina, K.: Squaring in cyclotomic subgroups. Mathematics of Computation (to appear)
19. Koblitz, N., Menezes, A.: Pairing-Based Cryptography at High Security Levels. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)
20. Lauter, K., Montgomery, P.L., Naehrig, M.: An Analysis of Affine Coordinates for Pairing Computation. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 1–20. Springer, Heidelberg (2010)
21. Lenstra, A., Lenstra, H., Lovasz, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 515–534 (1982)
22. Miller, V.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17, 235–261 (2004)
23. NSA Suite B Cryptography, `www.nsa.gov/ia/programs/suiteb_cryptography/`
24. Pereira, G., Simplício Jr., M., Naehrig, M., Barreto, P.: A family of implementation-friendly BN elliptic curves. Journal of Systems and Software 84, 1319–1326 (2011)
25. Pollard, J.: Monte Carlo methods for index computation mod p. Mathematics of Computation 32, 918–924 (1978)
26. Schirokauer, O.: Discrete logarithms and local units. Philosophical Transactions of the Royal Society London A 345, 409–423 (1993)
27. Scott, M.: Computing the Tate Pairing. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
28. Scott, M.: On the Efficient Implementation of Pairing-Based Protocols. In: Chen, L. (ed.) Cryptography and Coding 2011. LNCS, vol. 7089, pp. 296–308. Springer, Heidelberg (2011)
29. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 78–88. Springer, Heidelberg (2009)
30. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory 56, 455–461 (2010)

Improved Broadcast Encryption Scheme with Constant-Size Ciphertext

Renaud Dubois[1], Aurore Guillevic[1,2], and Marine Sengelin Le Breton[1]

[1] Thales Communications and Security – 160 Boulevard de Valmy – BP 82 92704 Colombes Cedex – France
firstname.lastname@thalesgroup.com
[2] Crypto Team, DI, ENS – 45 rue d'Ulm – 75230 Paris Cedex 05 – France

Abstract. The Boneh-Gentry-Waters (BGW) [3] broadcast encryption scheme is optimal regarding the overhead size. This performance relies on the use of a *pairing*. Hence this protocol can benefit from public key improvements. The main lasting constraint is the computation time at receiver end as it depends on the number of revoked users. In this paper we describe two modifications to improve BGW bandwidth and time complexity. First we rewrite the protocol with an asymmetric pairing over Barreto-Naehrig (BN) curves instead of a symmetric one over supersingular curves. This modification leads to a practical gain of 60% in speed and 84% in bandwidth. The second tweaks allows to reduce the computation time from $O(n-r)$ to $\min(O(r), O(n-r))$ for the worst case (and better for the average case). We give performance measures of our implementation for a 128-bit security level of the modified protocol on a smartphone.

Keywords: Broadcast encryption, asymmetric pairings, Barreto-Naehrig curves, Android.

1 Introduction

A broadcast encryption scheme is a protocol allowing a broadcaster to send messages to a large set $\mathcal{U}$ of *users* or receivers, $n = \#\mathcal{U}$. The set evolves at each session in a dynamic way such that the broadcaster may choose any subset $\mathcal{S}$ of *privileged users* or *members* from $\mathcal{U}$. $\mathcal{R}$ is the set of revoked users (non-members), $r = \#\mathcal{R}$. A broadcast protocol is secure under (t, n)-collusion if for all subset $\mathcal{R} \subset \mathcal{U}$ with $r \leqslant t$, the revoked users from $\mathcal{R}$ are not able to decipher. The classical application of broadcast encryption protocols are pay-TV systems, broadcast of content and over the air re-keying mechanisms (OTAR) in radio systems. The content or payload is encrypted under a private key. Then the private key is encrypted in a manner described by the chosen protocol. An *overhead* is added to the encrypted data. It contains the encrypted session key and a description of $\mathcal{S}$. The system constraints are the **bandwidth consumption**, related to the overhead size ω, the sender computation time τ_s, public key (resp. secret key) memory PK_s (resp. SK_s), the **users** (receivers) **computation time** τ_u, public

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 196–202, 2013.

key (resp. secret key) memory PK_u (resp. SK_u). The naive solution encrypt the session key one time per user, leading to a linear cost in bandwidth. Many schemes have been suggested to provide an efficient broadcast.

Our Contributions. A practical instantiation was not explained in [3]. A straightforward implementation of the protocol uses a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We propose to design BGW with an appropriate asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. In this way, the elements in $\mathbb{G}_1$ have the optimal size in public key cryptography, i.e. 256 bits for the example above, rather than half a RSA modulus size. We adapt the protocol and set in the right groups $\mathbb{G}_1$ or $\mathbb{G}_2$ the different elements (public and private keys, bandwidth elements), knowing that the elements in $\mathbb{G}_1$ have the smallest size, those of $\mathbb{G}_2$ have quite medium size (at most half an RSA modulus) and those of $\mathbb{G}_T$ are close to an RSA modulus size. The resulting bandwidth consumption is divided by 6 at a 128-bit security level. We adapt accordingly the security proof. We analyse the impact of Cheon's attacks [4] on ℓ-BDHE and propose a resistant elliptic curve. We provide an efficient trade-off between memory and precomputation. Finally our practical implementation on a smartphone shows that with all our improvements, this BGW broadcast encryption scheme can be efficiently used for commercial applications.

This paper is organized as follows: in Sec. 2 we describe how BGW can benefit from the use of an asymmetric pairing and give hints to adapt the security proof. In Sec. 3, we detail our choice of a pairing-friendly elliptic curve and consider modifications due to Cheon's attacks. In Sec. 4, we describe how to use well chosen pre-computation to reduce dramatically the computation cost. Finally, in Sec. 5 we give our results of a complete implementation of the protocol on a smartphone. Full version of this paper will be available at [5].

2 BGW with an Asymmetric Pairing

Boneh, Gentry and Waters [3] describe a scheme with a minimal overhead. The scheme use a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Definition and properties can be found in [2, Ch. IX]. We will use the additive notation for both $\mathbb{G}_1$ and $\mathbb{G}_2$ and the multiplicative notation for $\mathbb{G}_T$. In the original paper, the scheme is described with a symmetric pairing. To this end, supersingular elliptic curves shall be used, which is inefficient. We adapt the scheme to an asymmetric pairing in order to have a group $\mathbb{G}_1$ with smaller coefficients. We reorganise the elements and set in $\mathbb{G}_1$ those on which the bandwidth depends.

2.1 General Scheme

To reduce the public key size, the n users are organized into A groups of B users with $AB \geqslant n$. In [3] the authors suggest to choose $B = \lfloor \sqrt{n} \rfloor$ and $A = \lceil \frac{n}{B} \rceil$. We can also divide users into groups according to their country, subscription or other criterion due to the system (Pay-TV, OTAR). A user i is referenced by its group number (say a) and its range in that group (say b). Hence $i = \{a, b\}$ with

$1 \leqslant a \leqslant A$ and $1 \leqslant b \leqslant B$. The Hdr will contain A public elements (instead of a unique C_1), each one dedicated to a determined group of users. Here we see relevant to set all these elements in $\mathbb{G}_1$. There is still a C_0 element that we need to set in $\mathbb{G}_2$ in order to keep in $\mathbb{G}_1$ the user public and private keys and a part of the decryption.

Setup$_B(n)$ Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q$ as above. Let α a random element in $\mathbb{Z}_m$. Compute $P_i = \alpha^i P \in \mathbb{G}_1$ for $1 \leqslant i \neq B+1 \leqslant 2B$; this is the common public key. In each group of users, the user $i = \{a, b\}$ receives the set of (P_i) and an additional public key $Q_b = \alpha^b Q \in \mathbb{G}_2$. Then pick uniformly at random $\gamma_1, ..., \gamma_A \leftarrow \mathbb{Z}_m$ and set $V_1 = \gamma_1 P, ..., V_A = \gamma_A P \in \mathbb{G}_1$. The centralized public key is $\mathrm{PK}_s = (P, P_1, ..., P_B, P_{B+2}, ..., P_{2B}, V_1, ..., V_A, Q, Q_1) \in \mathbb{G}_1^{2B+A} \times \mathbb{G}_2^2$. The secret key for the user number b in the group a is $\mathrm{SK}_{u,\{a,b\}} = D_{a,b} = \gamma_a P_b \in \mathbb{G}_1$. Its public key is $\mathrm{PK}_{u,\{a,b\}} = \big(Q_b, (P_i)_{1 \leqslant i \leqslant 2B,\ i \neq B+1}\big)$. The user doesn't need the others Q_ℓ hence to save memory on his constrained device (e.g. smartphone, set-up box) we don't add them. Note that this scheme is relevant even for unbalanced group sizes. For large groups, the time computation will increase, but the bandwidth consumption will be the same : one group element (in $\mathbb{G}_1$) per group of users, whenever the size of the group is.

Encrypt$(\mathcal{S}, \mathrm{PK}_s)$ For each group a of users, denote by $\mathcal{S}_a$ the set of authorized users in this group. Pick a random k in $\mathbb{Z}_m$ and set $K = e(P_{B+1}, Q)^k = e(P_B, Q_1)^k \in \mathbb{G}_T$. Set Hdr $(\in \mathbb{G}_2 \times \mathbb{G}_1^A) =$
$\Big(kQ, k\big(V_1 + \sum_{j \in \mathcal{S}_1} P_{B+1-j}\big), k\big(V_2 + \sum_{j \in \mathcal{S}_2} P_{B+1-j}\big), ..., k\big(V_A + \sum_{j \in \mathcal{S}_A} P_{B+1-j}\big)\Big)$

Decrypt$\big(i = \{a, b\}, \mathcal{S}_a, \mathrm{Hdr}, \mathrm{SK}_{u,\{a,b\}}, \mathrm{PK}_{u,\{a,b\}}\big)$ Let denote Hdr $= (C_0, C_1, ..., C_A)$. The user i computes $K = e(C_a, Q_b)/e\big(D_{a,b} + \sum_{j \in \mathcal{S}_a, j \neq b} P_{B+1-j+b},\ C_0\big)$. The verification uses the relation $e([i]P, [j]Q) = e(P, Q)^{ij} = e([j]P, [i]Q)$. The following table gives the protocol complexity with an asymmetric pairing, BGW_1 denotes the one instance version, BGW_2 denotes the parallel instance version. ω is the bandwidth consumption, μ_s denotes the sender's memory, τ_s the time computation and respectively μ_u, τ_u denote the receiver's ones. r_a is the number of revoked users in the group a. Note that they are at most B users in a group a.

Schema	ω	μ_s	τ_s	μ_r	τ_r
BGW_1	$\mathbb{G}_2 \times \mathbb{G}_1$	$\mathbb{G}_1^{2n+1} \times \mathbb{G}_2^{n+1}$	$(n-r)\mathrm{Add}_{\mathbb{G}_1}$	$\mathbb{G}_1^{2n-1} \times \mathbb{G}_2$	$(n-r)\mathrm{Add}_{\mathbb{G}_1}$
BGW_2	$\mathbb{G}_2 \times \mathbb{G}_1^A$	$\mathbb{G}_1^{2B+A} \times \mathbb{G}_2^{B+1}$	$(n-r)\mathrm{Add}_{\mathbb{G}_1}$	$\mathbb{G}_1^{2B-1} \times \mathbb{G}_2$	$(B-r_a)\mathrm{Add}_{\mathbb{G}_1}$

Theoretical complexity for BGW protocol, asymmetric pairing

2.2 Idea of the Security Proof

In [3, §3.3], the authors prove the semantic security of the general system. We faced some trouble when adapting the security proof to an asymmetric pairing in the setting above. We need to add a copy in $\mathbb{G}_2$ of the inputs elements in $\mathbb{G}_1$ to the problem. This difficulty rises in the challenge phase. To generate a consistent input for the adversary, the challenger must have a copy in $\mathbb{G}_2$ of the inputs in $\mathbb{G}_1$. This is transparent with a symmetric pairing (an isomorphism from

$\mathbb{G}_1$ into $\mathbb{G}_2$ is available). Such a map usually does not exists for ordinary pairing-friendly elliptic curves. pairing. For ordinary elliptic curves, The trace map [2, IX.7.4] is degenerated, as $\mathbb{G}_2$ is commonly built as the trace-zero subgroup. With the notations from [6], the security proof must be written assuming that the pairing is of Type 3 : $\mathbb{G}_1 \neq \mathbb{G}_2$ and there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. Hence the adversary needs to receive $P^{'}$, that is why it must appears in the challenger inputs. The security relies on the ℓ-Bilinear Diffie-Hellman Exponent assumption which is a weaker problem than the Diffie-Hellman one. The difficulty of this problem was studied in [4], see Th. 1 and Th. 2. The main idea for the first theorem is to find a divisor d of $m-1$ in the range $2 \leqslant d \leqslant B$ or $B+2 \leqslant d \leqslant 2B$ to reduce the complexity from $O(\sqrt{m})$ to $O(\sqrt{m/d} + \sqrt{d})$.

3 Choice of the Pairing-Friendly Elliptic Curve

If the protocol relies exclusively on the Diffie-Hellman problem, for a N-bit security level, prime order groups $\mathbb{G}_1, \mathbb{G}_2$ of size $\log m = 2N$ bits are convenient. The group $\mathbb{G}_T \simeq \mathbb{F}_{p^k}^*$ is exposed to the less difficult index calculus attack, its size $k \log p$ must be a RSA modulus size. We have chosen a 128-bit security level and consider NIST recommendations on key sizes. A supersingular curve (over a prime field in large characteristic) has an embedding degree k at most 2 resulting in $\log p = 1536$, $\log m = 256 + \delta$ (i.e. enlarging m by a few δ bits will not impact on $\log p$) and $\rho = \log p / \log m \approx 6$. The well-known Barreto-Naehrig curves (BN, [1]) fit exactly the NIST recommended sizes of $\mathbb{G}_1$ and $\mathbb{G}_T$ with $k = 12$, $\log p = 256$, $k \log p = 3072$. In particular, for a 128 bit security level, using an asymmetric pairing decreases the size of the element in the group $\mathbb{G}_1$ by a factor of 6. To prevent from Cheon's attacks, we can increase the size of m by 12 bits but it result in increasing the size of $\mathbb{F}_{p^k}$ by 144 bits. To avoid this, we must generate a strong BN curve, without any integer d dividing m and less than 2^{12}. The BN curve generated by $x = \texttt{0x4000000000087F7F} = 248861 \cdot 18531172093771$ does not have a smooth order. For this curve, $12 \mid m-1$ and the next divisor is 248861; $2 \mid m+1$ and the next divisor is 480707. Because of the 12, we loose 4 bits. A bypass would be to index the users from 13 instead of 1. Our implementation doesn't depends on a particular p or m hence changing their value will not infer on the timings if their size remains the same.

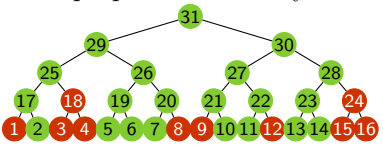
4 Reducing Time Complexity

The public keys are points on an elliptic curve hence addition is as cheap as subtraction. If the number of revoked users is small ($r \ll n/2$), the initial computation in $O(n-r)$ is quite slow. We can instead consider that the value $\Sigma_{n,i} = \sum_{1 \leqslant j \neq i \leqslant n} P_{n+1-j+i}$ is precomputed for each user i. Then $S = \Sigma_{n,i} - \sum_{j \in \mathcal{R}} P_{n+1-j+i}$ with $\mathcal{R}$ the set of revoked users. Now the complexity is $O(\min(r, n-r))$ (where O is the cost of a point addition, EllAdd). We can do better with a precomputed tree.

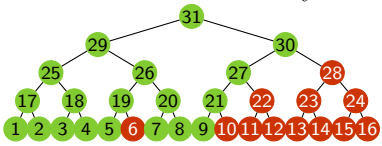
Binary Public Key Tree Precomputation. To decrease the computation time from $O(\min(r, n-r))$, we modify the public key into a binary public key tree T twice long obtained by

- sorting all users in a binary tree whose leaves are the users;
- precomputing for each node the sum of each public key of the nodes below.

For each encryption and decryption, choose the optimal including/excluding tree to compute the sum. A full description of the algorithm is given in [5]. In [3] the authors propose to store the previous sum S from a session to the next, subtract the new revoked users and add the no-longer revoked ones. This is efficient only if the proportion of newly revoked and re-authorized users is very small.



Random distribution



Ordered distribution

Example 1 (Computing S quickly in a tree). Consider a set of $n = 16$ users, indexed from 1 to 16 as illustrated in two above figures (revoked users are black). For user '2' compute the key session by subtracting the value in node '1,3,4,8,9,12,24' thus the cost is 6 EllAdd. Note that in the figure on the right (sorted tree) the cost is only 3 EllAdd (subtract node '30' and '6', add node '9'). Cost would have been 8 in the original scheme.

The average case is hard to analyse as it strongly depends on the distribution of revoked users in the tree. In practice the users are sorted by behaviour so that nodes that are close are mostly to be revoked together. In a real world application the behaviour is the subscription date or product. However some random revocations (rare events) appear with compromising, expirations, etc.

5 Implementation on a Smartphone

We chose to develop a very generic library in C language which can use any modulus and any type of pairing-friendly elliptic curve in Weiertraß representation over a prime finite field (i.e. in large characteristic). The BN curves and supersingular curves have been implemented. The library is a proprietary industrial library using a modular approach as in OpenSSL. It implements arithmetic over $\mathbb{F}_p$ using Montgomery multiplication, elliptic curve computation over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$ using the modified Jacobian coordinates. The pairing computation is specific for each $\mathbb{F}_{p^k}$ field.

We now present some computational results of our improved implementation for 128-bit security level. Our proof of concept consists in a standard PC to represent the sender, and a smartphone to represent the receiver. The smartphone can be personalized with any secret key of the system. Thus the given results are

the same as would be in a real system with a million smartphones. The smartphone is a dual core 1.2 Ghz Samsung Galaxy II with Android OS. The PC is a 3Ghz Intel(R) Core(TM)2 Duo CPU with 2.9 Gio RAM. The broadcaster runs the system initialization, the key attribution to a new user and the session key encryption. First, we simulate the decryption time for an authorized user on the PC to estimate the growing cost of decryption with respect to the total number of users n.Smartphones with Android platform use the Java programming language. Thanks to the Java Native Interface, we can load the library in C language, run the decryption on the smartphone and measure its timing. Results are presented in the following table.We measure the worst case $r = n/2$ of BGW_2 so the improvements described in Sec. 4 are not visible. The users are divided in A parallelized groups of B users with $B = \lceil \sqrt{n} \rceil$.

Number of users n	Setup	User init.	Encryption $r = n/2$	Decryption (simulation)	Decryption (smartphone)
50000	22.15 s	0.03 s	3.58 s	1.10 s	1.44 s
100000	40.45 s	0.03 s	7.03 s	1.13 s	1.79 s
200000	1 m 16 s	0.03 s	14.72 s	1.14 s	2.08 s
500000	3 m 07 s	0.05 s	32.97 s	1.16 s	2.65 s
1000000	6 m 09 s	0.07 s	1 m 04 s	1.18 s	3.33 s
5000000	30 m 42 s	0.16 s	5 m 11 s	1.27 s	6.09 s

Computation time on a 3 Ghz PC (encryption) and a smartphone

An acceptable decryption time on the smartphone must be less than 2 seconds from our point of view. Here this correspond to less than 200 000 users. For larger n, we need to reduce this time. The pairing computation is not very time consuming. The sum $\sum_{j \in \mathcal{S}_a, j \neq b} P_{B+1-j+b}$ is the most important part.With a first trick: addition over $\mathcal{S}_a$ when $n - r \ll n$ and subtraction over $\mathcal{R}_a$ (the revoked users of group a) when $r \ll n$, the worst case of $r = n/2$ become the upper bound. This means still at most $3.33s$ when $r = n/2$. With a precomputed tree, the average case will have faster encryption and decryption times than those presented.

6 Conclusion

We presented a well improved version of BGW suitable for use with a pairing on one of the fastest pairing-friendly elliptic curves. Our presentation can be easily adapted to other well-suited pairing friendly elliptic curves. We considered the attacks on the underlying non-standard problem. We also provided time computation on a prototype, the broadcaster hosted on a standard PC and each receiver hosted on a Samsung Galaxy II smartphone with Android OS. For large groups of users (more than 200000), the decryption time is up to $2s$ which can be too slow. Hence we proposed improvements based on a time-memory trade-off. Because of the use of an asymmetric pairing, the public key size remains reasonable, hence doubling this size is feasible in order to reduce under $2s$ the

decryption time. We then explained and justified all our choices to use in practice in a real Pay-TV or OTAR system the BGW protocol.

Acknowledgments. This work was supported in part by the French ANR-09-VERS-016 BEST Project. The authors express their gratitude to Philippe Painchault, Emeline Hufschmitt and Damien Vergnaud for helpful comments and careful proofreading. The authors thank the anonymous reviewers of the Pairing conference for their thorough reviews and useful comments.

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] Blake, I.F., Seroussi, G., Smart, N.P. (eds.): Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series, vol. 317. Cambridge University Press (2005)

[3] 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)

[4] Cheon, J.H.: Discrete logarithm problems with auxiliary inputs. J. Cryptology 23, 457–476 (2010)

[5] Dubois, R., Guillevic, A., Sengelin Le Breton, M.: Improved Broadcast Encryption Scheme with Constant-Size Ciphertext. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 196–202. Springer, Heidelberg (2013)

[6] Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)

Affine Pairings on ARM

Tolga Acar[1], Kristin Lauter[1], Michael Naehrig[1,2,⋆], and Daniel Shumow[1]

[1] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
{tolga,klauter,danshu}@microsoft.com
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
michael@cryptojedi.org

Abstract. We report on relative performance numbers for affine and projective pairings on a dual-core Cortex A9 ARM processor. Using a fast inversion in the base field and doing inversion in extension fields by using the norm map to reduce to inversions in smaller fields, we find a very low ratio of inversion-to-multiplication costs. In our implementation, this favors using affine coordinates, even for the current 128-bit minimum security level specified by NIST. We use Barreto-Naehrig (BN) curves and report on the performance of an optimal ate pairing for curves covering security levels between 128 and 192 bits. We compare with other reported performance numbers for pairing computation on ARM CPUs.

Keywords: Optimal ate pairing, BN curves, ARM architecture.

1 Introduction

For Elliptic Curve Cryptography (ECC) applications on NIST P-curves where prime fields are chosen with Generalized Mersenne primes for fast modular reduction and multiplication, the base field inversion-to-multiplication ratio is often reported to be 80 : 1 or higher. However, for pairing applications which require fixed embedding degrees to control efficiency and security, special primes like Mersenne primes cannot be used because there is no known way to generate pairing-friendly curves over those particular fields. Instead, more general prime fields arise, and much of the arithmetic in pairing computation is done in extension fields with degree 12 when using Barreto-Naehrig (BN) curves [3,14]. Computing in general prime fields using fast inversion techniques, a typical inversion-to-multiplication ratio can be much lower than 80 : 1.

In [9], inversion-to-multiplication ratios in extension fields were given which reflect faster inversion in extension fields by taking the norm down to smaller fields and doing inversion there. For example, in an extension field of degree 12, an inversion-to-multiplication ratio of 1.7 : 1 was reported for a 256-bit prime base field. Even for implementations with much faster field multiplies, using that

⋆ Acknowledges funding from the Netherlands Organisation for Scientific Research (NWO) under project PACE - Pairing Acceleration for Cryptography using Elliptic Curves.

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 203–209, 2013.

technique, the ratio decreases dramatically as the field extension degree increases, which leads to the argument made in [9] that for any implementation, as the security requirements and thus the field extension degrees grow, there exists a cross-over point after which it becomes more efficient to use affine coordinates in the pairing algorithm rather than projective coordinates. For the implementation of field arithmetic in 256-bit prime fields discussed in [9] on the x86 and x86-64 platforms, this cross-over point already occured when considering extension degree 2.

This led us to wonder what the cross-over point might be on other platforms, and how it would vary with different intrinsics or instruction sets. In this paper, we give performance numbers for affine and projective pairings on a dual-core Cortex A9 ARM processor. Our implementation targets a minimum 128-bit security level. It thus works with BN curves with embedding degree 12, and involves curve arithmetic in a degree-2 extension field by taking advantage of sextic twists as usual. The high-level pairing implementation is very close to that reported in [9]. We use intrinsics and assembly for the Montgomery multiplication implementation. Improving the underlying field multiplication algorithm would certainly increase the degree at which one would switch from projective to affine pairings.

In our implementation, affine coordinates are the better choice for pairing computation also on the ARM processor. Other implementations presented in the literature recently have faster field multiplies that are optimized for specific processor architectures to obtain pairing speed records [12,4,2], whereas our code is not optimized for any particular architecture. Our implementation of affine pairings compares favorably with all other reported ARM pairing timings we have found in the literature (see Section 4).

2 Platform-Specific Improvements on ARM

The multiplication routines in our implementation utilize multiply, multiply-accumulate, and Montgomery multiplication [11], and the compiler is Microsoft Visual C++ on ARM (Thumb-2).

We used a Tegra 2 development platform from NVidia to obtain the benchmark figures in Table 1. This system features a dual-core Cortex A9 ARM CPU running at 1GHz with 32KB/32KB (I/D) L1 cache per core, 1MB L2 cache, and 1GB DDR2-667 main memory. The entire benchmark program fits in the 1MB L2 cache, and the core routines executed in tight loops fit in the 32KB instruction cache.

The Montgomery multiplication function implements the CIOS method in [8], and its performance is given in Table 1 for various moduli lengths. The C implementation relies on the compiler support for double-length unsigned integers (`unsigned __int64`). The intrinsics method uses a few compiler-supported ARM assembly instructions: `umull, umaal, umlal` while other operations are implemented in C. The `umull` is an unsigned 32-bit integer multiplication instruction that generates an unsigned 64-bit product. The `umlal` is a 32-bit multiply and 64-bit accumulate, and the `umaal` is a 32-bit multiply and double 32-bit

accumulate instruction. The assembly row reports the benchmark figures where the CIOS method is implemented in ARM Thumb-2 assembly language.

The difference between the assembly and the C-with-intrinsics implementation is in the Montgomery multiplication routine. Both implementations use the above instrinsics in other primitive functions (e.g., multiply and multiply-accumulate), such as inversion.

Table 1. Montgomery multiplication implementation choices and benchmark figures in micro seconds.

	Modulus length in bits					
Implementation	**160**	**224**	**288**	**480**	**640**	**3168**
Intrinsics	2.07	2.55	3.17	5.66	9.26	147
Assembly	1.97	2.41	2.93	5.15	9.04	128

While the use of intrinsics provides an improvement over the C version, the assembly implementation provides an incremental improvement over intrinsics. We experimented with several implementation approaches such as loop unrolling, different instruction ordering, conditional instructions, and multi-word load/stores. None of these approaches provided a measurable performance improvement on our reference platform. Thus, we did not use any of these techniques to generate the numbers on the table. Instead, we carefully crafted a straightforward assembly implementation of the Montgomery multiplication CIOS algorithm in [8] to form base reference benchmark numbers. The assembly implementation and intrinsics only leverage the core ARM instruction set, but do not utilize SIMD and NEON instructions.

3 Implementation and Performance

Here we present the timing results of our pairing implementation on BN curves for the ARM instruction sets, for security levels of 128 bits or higher. Our pairing code can be used to compute pairings on all 16 curves recently introduced in [14]. In particular, the code is not tailored for one specific curve. These curves are easy to generate, have a very compact representation and were chosen to provide very efficient implementation. The loop order $6u + 2$ for all curves is very sparse when represented in non-adjacent form. Furthermore, the curve of size 254 bits has recently been used to obtain the current software speed record for pairings as outlined in [2].

Due to space constraints, we present performance results for only three of the curves in [14], namely the curves bn254, bn446, and bn638 over prime fields of respective bit sizes 254, 446, and 638 bits. The curve bn254 roughly provides 128 bits of security and bn638 yields about 192 bits.

Our implementation uses the optimal ate pairing on BN curves. For the projective version we used the explicit formulas in [6], but we obtained better results

for the affine version. It uses the tower of field extensions $\mathbb{F}_{p^{12}}/\mathbb{F}_{p^6}/\mathbb{F}_{p^2}/\mathbb{F}_p$ via the method described in [9] to realize field arithmetic in $\mathbb{F}_{p^{12}}$. Fast methods for doing inversion in the base field were described in [5, Appendix D]. The final exponentiation is done using the Frobenius action and the addition chain from [15] as well as the special squaring functions from [7].

Our implementation results are shown in Table 2. We give timings for the finite field additions (**add**), subtractions (**sub**), multiplications (**M**), squarings (**S**) and inversions (**I**) as well as the inversion-to-multiplication ratio ($\mathbf{R} = \mathbf{I}/\mathbf{M}$) for all fields in the tower of extensions.

We give timings for several pairing functions that use different optimizations for different computing scenarios. The line entitled "20 at once (per pairing)" gives the average timing for one pairing out of 20 that have been computed at the same time. This function uses Montgomery's inversion-sharing trick as described in [9, Section 4.3]. The function corresponding to the line "product of 20" computes the product of 20 pairings. The lines with the attribute "1st arg. fixed" mean functions that compute multiple pairings, where the first input point is fixed for all pairings, and only the second point varies. In this case, the operations depending only on the first argument are done only once. We list separately the final exponentiation timings. They are included in the pairing timings of the other lines.

We do not give cycle counts for the ARM implementation in the tables since high-frequency counters are currently not supported in our development environment on the ARM. However, estimates for cycle counts can be easily read off from the values given in μs and ms by multiplying them by 10^3 and 10^6, respectively (note the clock frequency of 1GHz for the ARM processor).

3.1 Summary of our Implementation Performance Results

Here we summarize some of the results given in Table 2 on the performance of our implementation across platforms and security levels. As high-level points of comparison, we note that:

1. Affine coordinates are better than projective coordinates for optimal ate pairing computation at all security levels. The trend is toward bigger differences at higher security levels. The affine pairing is roughly 20% better at the 192-bit security level instead of 10% better at the 128-bit security level. For example, for the 254-bit curve, an affine pairing takes 51 milliseconds while the projective pairing takes 55 milliseconds, whereas for the 638-bit curve, an affine pairing takes 650 milliseconds while the projective pairing takes 768 milliseconds.
2. The inversion-to-multiplication ratio is lower in larger base fields. This largely explains observation 1 above.
3. In the degree-12 extension fields, the inversion-to-multiplication ratio is close to 1.7 : 1 at all security levels. There is very little variation in that, despite big differences in ratios in the base fields.
4. The percentage of the computation time spent on the final exponentiation goes up at the higher security levels, and this is true across platforms:

Table 2. Field arithmetic timings in 254-, 446-, and 638-bit prime fields and optimal ate pairing timings on corresponding BN curves. Field timings average over 1000 operations, pairing timings average over 20 pairings.

ARM, dual-core Cortex A9 @ 1GHz, Windows

254-bit prime field	**add** μs	**sub** μs	**M** μs	**S** μs	**I** μs	**R = I/M**
$\mathbb{F}_p$	0.67	0.61	1.72	1.68	18.35	10.67
$\mathbb{F}_{p^2}$	1.42	1.24	8.18	5.20	26.61	3.25
$\mathbb{F}_{p^6}$	4.43	3.96	69.83	48.24	136.68	1.96
$\mathbb{F}_{p^{12}}$	9.00	8.32	228.27	161.43	379.09	1.66

bn254		ms
projective		55.19
affine	single pairing	51.01
	20 at once (per pairing)	50.71
	20 at once, 1st argument fixed (per pairing)	46.06
	product of 20 (per pairing)	17.44
single final exponentiation		24.69

446-bit prime field	**add** μs	**sub** μs	**M** μs	**S** μs	**I** μs	**R = I/M**
$\mathbb{F}_p$	1.17	1.03	4.01	3.92	35.85	8.94
$\mathbb{F}_{p^2}$	2.37	2.07	17.24	10.84	54.23	3.15
$\mathbb{F}_{p^6}$	7.77	7.15	152.79	109.74	302.34	1.98
$\mathbb{F}_{p^{12}}$	15.65	14.88	498.58	364.34	846.21	1.70

bn446		ms
projective		195.56
affine	single pairing	184.28
	20 at once (per pairing)	183.54
	20 at once, 1st argument fixed (per pairing)	167.83
	product of 20 (per pairing)	62.33
single final exponentiation		86.75

638-bit prime field	**add** μs	**sub** μs	**M** μs	**S** μs	**I** μs	**R = I/M**
$\mathbb{F}_p$	1.71	1.53	8.22	8.18	56.09	6.82
$\mathbb{F}_{p^2}$	3.48	3.17	31.81	20.55	91.92	2.89
$\mathbb{F}_{p^6}$	10.63	10.09	261.87	186.21	535.42	2.04
$\mathbb{F}_{p^{12}}$	21.04	20.28	840.07	607.36	1454.38	1.73

bn638		ms
projective		768.06
affine	single pairing	649.85
	20 at once (per pairing)	650.08
	20 at once, 1st argument fixed (per pairing)	609.45
	product of 20 (per pairing)	164.82
single final exponentiation		413.37

For example, for the 254-bit curve, an affine pairing spends 48% of the time on the final exponentiation, whereas for the 638-bit curve, an affine pairing spends 63% of the time on the final exponentiation.

4 Related Work

For applications of pairings to privacy of electronic medical records using Attribute-Based Encryption for key management, some recent performance numbers for pairings on ARM processors were reported in [1]. The comments in [1, Section 6.1] give rough performance numbers for pairings on ARM: the Pairing-Based Crypto (PBC) [10] library computes pairings in 135 milliseconds on an ARM processor running on Apple A4 chip-based iPhone 4, running iOS 4 with 512MB of RAM and computing on a 224-bit MNT elliptic curve.

It is hard to compare across different hardware and operating systems, but as a point of reference, our implementation of affine optimal ate pairings computes pairings on curves of comparable security level, 222-bit BN curves, in 53 milliseconds, on the hardware Tegra 2 NVidia, Dual-core ARM Cortex A9, 1GHz, 1MB L2 cache, 32KB/32KB (I/D) L1 per core, DDR2-667. Note that MNT curves have embedding degree 6 instead of 12 as for BN curves, which means less security and faster extension field operations and final exponentiation.

Working on elliptic curves over binary fields $\mathrm{GF}(2^{271})$ and using embedding degree 4 on processors somewhat comparable to the ones considered here, the Imote2 platform (13MHz PXA271, a 32-bit ARMv5TE with 32 KB data cache and 32 KB instruction cache), [13, Table 3] shows a pairing computation in 140 milliseconds. Again these computations are not really comparable because of the 70-bit security level, different hardware and operating system, binary fields, different curve and embedding degree.

In [16] the authors report a performance of some optimal pairings on supersingular elliptic curves in characteristic 3, using the BREW emulator on 150 MHz and 225 MHz ARM9 processors. Their implementation achieves a pairing computation in 401 and 262 milliseconds respectively over the base field $\mathrm{GF}(3^{193})$ on curves claimed to be at the 80-bit security level.

Acknowledgements. We thank Patrick Longa and Diego F. Aranha for valuable comments on an earlier version of this work and interesting discussions. We also thank the anonymous referees for their helpful comments.

References

1. Akinyele, J.A., Lehmanny, C.U., Green, M.D., Pagano, M.W., Peterson, Z.N.J., Rubin, A.D.: Self-protecting electronic medical records using attribute-based encryption. Cryptology ePrint Archive, Report 2010/565 (2010), http://eprint.iacr.org/2010/565/
2. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster Explicit Formulas for Computing Pairings over Ordinary Curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (2011)

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. Beuchat, J.-L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
5. Ciet, M., Joye, M., Lauter, K., Montgomery, P.L.: Trading inversions for multiplications in elliptic curve cryptography. Des. Codes Cryptography 39(2), 189–206 (2006)
6. Costello, C., Lange, T., Naehrig, M.: Faster Pairing Computations on Curves with High-Degree Twists. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 224–242. Springer, Heidelberg (2010)
7. Granger, R., Scott, M.: Faster Squaring in the Cyclotomic Subgroup of Sixth Degree Extensions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 209–223. Springer, Heidelberg (2010)
8. Koç, Ç.K., Acar, T.: Analyzing and comparing Montgomery multiplication algorithms. IEEE Micro 16, 26–33 (1996)
9. Lauter, K., Montgomery, P.L., Naehrig, M.: An Analysis of Affine Coordinates for Pairing Computation. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 1–20. Springer, Heidelberg (2010)
10. Lynn, B.: The Pairing-Based Cryptography Library (PBC), http://crypto.stanford.edu/pbc/
11. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44(170), 519–521 (1985)
12. Naehrig, M., Niederhagen, R., Schwabe, P.: New Software Speed Records for Cryptographic Pairings. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 109–123. Springer, Heidelberg (2010), Corrected version http://www.cryptojedi.org/papers/dclxvi-20100714.pdf
13. Oliveira, L.B., Aranha, D.F., Gouvêa, C.P.L., Scott, M., Câmara, D.F., López, J., Dahab, R.: TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks. Computer Communications 34(3), 485–493 (2011)
14. Pereira, G.C.C.F., Simplício, Jr., M.A., Naehrig, M., Barreto, P.S.L.M.: A family of implementation-friendly BN elliptic curves. Journal of Systems and Software (2011) (to appear), doi:10.1016/j.jss.2011.03.083
15. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 78–88. Springer, Heidelberg (2009)
16. Yoshitomi, M., Takagi, T., Kiyomoto, S., Tanaka, T.: Efficient Implementation of the Pairing on Mobilephones Using BREW. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 203–214. Springer, Heidelberg (2008)

On the Implementation of a Pairing-Based Cryptographic Protocol in a Constrained Device

Sébastien Canard[1], Nicolas Desmoulins[1], Julien Devigne[1,2], and Jacques Traoré[1]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] UCBN, GREYC, Caen, France

Abstract. In this paper, we consider a pairing-based cryptographic protocol and the way to implement it on a restricted device such as a mobile phone or a smart card. Our aim is to show the different ways to do it, regarding (i) the capacity for the restricted device to implement a bilinear pairing and/or (ii) the performance regarding the implemented bilinear pairing. We show that there are different possibilities and study the security and efficiency of each of them. To illustrate our purpose, we make use of the Boneh-Boyen-Shacham group signature, which needs one on-line pairing computation.

1 Introduction

When operating in devices with restricted capabilities w.r.t. space, memory and computing performance, the implementation of some cryptographic algorithms sometimes need to be further studied. In these cases, it is important to find tricks to optimize the implementation until performance is acceptable by the customer. This is in particular the case when the studied cryptographic algorithm includes the use of one or several bilinear pairings.

In fact, bilinear pairings are today not studied enough to be embedded into any mobile phone or smart card, as it is the case for *e.g.* RSA or EC-DSA. Then, when one has to embed a pairing-based cryptographic algorithm onto *e.g.* a SIM card for mobile phones, one has to make some choices on the way to implement the whole algorithm, and in particular the bilinear pairing itself. In this paper, we study several possibilities, giving for each of them *pros and cons*.

To illustrate the different possibilities we have studied, we take in this paper the case of the implementation of the BBS group signature scheme [2] either in a mobile phone connected to the outside world, or in a SIM card connected to a mobile phone. Informally, in a group signature scheme [5], any member of the group can sign a document and any verifier can confirm that the signature has been computed by a member of the group. Moreover, group signatures are anonymous and unlinkable for every verifier except, in case of a dispute, for a given authority that knows some special information. In 2004, Boneh, Boyen and Shacham [2] have proposed a short group signature based on the use of a bilinear pairing (especially by the group member, during the group signature process).

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 210–217, 2013.

The group member is now represented by a mobile phone or a SIM card and is connected to some device (a PC or a mobile phone resp.).

The paper is organized as follows. In Section 2, we introduce our study by giving the description of the BBS group signature and a summary of the performances one can obtain regarding mathematical operations related to a bilinear pairing. In Section 3, we explain how one can avoid the need for group members (the mobile phone or the SIM card) to produce a bilinear pairing by replacing such operations by exponentiations in $\mathbb{G}_T$ and state the expected results. In Section 4, we show how the computation of a bilinear pairing can be delegated to some more powerful entity (the connected device).

2 Introduction to Our Study

In this section, we introduce our study by describing the BBS group signature scheme and then giving some implementation results regarding pairings and related groups.

2.1 The BBS Group Signature Scheme

Boneh, Boyen and Shacham have proposed at CRYPTO 2004 a short group signature scheme [2] based on the Strong Diffie-Hellman and the Decisional Linear assumptions[1].

KEY GENERATION. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order r and let g_1 (resp. g_2) be a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ be a bilinear map such that for all $(a, b) \in \mathbb{G}_1 \times \mathbb{G}_2$ and all $\alpha, \beta \in \mathbb{Z}$, $e([\alpha]a, [\beta]b) = e(a, b)^{\alpha\beta}$ and $e(g_1, g_2) \neq 1$.

Let $h \in \mathbb{G}_1$, $\zeta_1, \zeta_2 \in \mathbb{Z}_p^*$ and $u, v \in \mathbb{G}_1$ such that $[\zeta_1]u = [\zeta_2]v = h$. Let $\gamma \in \mathbb{Z}_p^*$ and $w = [\gamma]g_2$. Then, the tuple (ζ_1, ζ_2) composes the secret values to open a signature (*i.e.* revoke the anonymity), γ is the secret key to add group members and $(p, g_1, g_2, h, u, v, w)$ is the public key of the whole group.

Each group member obtains from the group manager a tuple $(A, x) \in \mathbb{G}_1 \times \mathbb{Z}_p^*$ such that $A = [1/(\gamma + x)]g_1$. This couple verifies $e(A, w + [x]g_2) = e(g_1, g_2)$. The value x is the member's secret and A is the key used to retrieve the identity of a member in case of opening (*i.e.* anonymity revocation).

GROUP SIGNATURE GENERATION. On input a message m and a tuple (A, x), a group signature is executed as follows:

- choose at random $\alpha, \beta \in \mathbb{Z}_p$ and compute $T_1 = [\alpha]u$, $T_2 = [\beta]v$ and $T_3 = A + [\alpha + \beta]h$;
- compute $\delta_1 = x\alpha$ and $\delta_2 = x\beta$;

[1] We used the additive notation for group laws in $\mathbb{G}_1$ and $\mathbb{G}_2$, which are elliptic curve groups over some finite field $\mathbb{F}_p$, and multiplicative notation for $\mathbb{G}_T$, which is a subgroup of $\mathbb{F}_{p^k}^*$.

- choose at random $r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2} \in \mathbb{Z}_p$;
- compute $t_1 = [r_\alpha]u$, $t_2 = [r_\beta]v$, $t_3 = [r_x]T_1 + [-r_{\delta_1}]u$, $t_4 = [r_x]T_2 + [-r_{\delta_2}]v$ and

$$t_5 = e(T_3, g_2)^{r_x} e(h, w)^{-r_\alpha - r_\beta} e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}; \quad (1)$$

- compute $c = \mathcal{H}(m \| T_1 \| T_2 \| T_3 \| t_1 \| t_2 \| t_3 \| t_4 \| t_5)$;
- compute $s_\alpha = r_\alpha + c\alpha$, $s_\beta = r_\beta + c\beta$, $s_x = r_x + cx$, $s_{\delta_1} = r_{\delta_1} + c\delta_1$ and $s_{\delta_2} = r_{\delta_2} + c\delta_2$.

A group signature is $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. Here (T_1, T_2, T_3) is a linear encryption of A (see [2] for such encryption scheme) and $(c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ is a proof of knowledge of a valid certificate (using the Fiat-Shamir heuristic [7]). The verification step consists then in verifying this proof of knowledge, using standard techniques, and the open procedure is the decryption of the linear encryption.

Implementation. We now focus on the group signature procedure and the way to implement it on *e.g.* the mobile phone of the group member.

As $e(h, w)$ and $e(h, g_2)$ depend only on public values, these pairings can be precomputed by *e.g.* the group manager (and directly put in the whole group public key), it only remains one additional bilinear pairing to compute: $e(T_3, g_2)$. As a conclusion, the group member should perform 7 random generations, 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$, 1 triple exponentiation in $\mathbb{G}_T$, 1 pairing evaluation and a few operations in $\mathbb{G}_1$, $\mathbb{G}_T$ and $\mathbb{Z}_p$ (which will be neglected in the following).

2.2 Implementation of a Bilinear Pairing

The security level implies minimal sizes for r (the size of the elliptic curve subgroup in which pairing operands live) and p^k (the size of the finite field which receives pairing outputs). The integers r and p are prime numbers, and k is the *embedding degree*. We have developed our own bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ with a Barreto-Naerhig elliptic curve E of equation $Y^2 = X^3 + 5$ over $\mathbb{F}_p$ (with p a prime number). More precisely, we have:

- $\mathbb{G}_1 = E[r](\mathbb{F}_p)$, where $E[r]$ denotes the r-torsion group;
- $\mathbb{G}_2 = E[r](\mathbb{F}_p) \cup \mathrm{Ker}(\pi_p - [p]) \subseteq E[r](\mathbb{F}_{p^k})$, where $\pi_p : E \longrightarrow E$ is the Frobenius endomorphism; and
- $\mathbb{G}_T = \mu_r \subset \mathbb{F}_{p^k}^*$ where μ_r is the group of r-th roots of unity.

We have then chosen a 128-bit security level, which gives us $\log_2 r = 256$ and $\log_2 p^k = 3248$. For $k = 12$, we obtain $|p| = |r| = 256$.

Some Optimizations. To accelerate the computation of pairings, some optimizations were used. In particular, we make use of the results given in [6] for some general low level optimizations (in particular the use of Jacobian coordinates and the joint point-and-line computation).

We have then implemented the Ate pairing as described in [1], which permits us to boost the pairing computation. Our results are given in Table 1 for the Samsung Galaxy S2 smartphone with a Dual-core Exynos 4210 1.2GHz processor ARM Cortex-A9 with the Android OS, v2.3 (Gingerbread).

Table 1. Our implementation benchmark

operation	notation	time computation (in ms)
Scalar multiplication in $\mathbb{G}_1 = E[r](\mathbb{F}_p)$	ϵ	5.7
Exponentiation in $\mathbb{G}_T = \mathbb{F}^*_{p^{12}}$	ζ	42
Ate pairing e	ψ	63

Dealing with Multi-scalar Multiplications. The BBS scheme moreover needs to implement the multi-scalar multiplication in $\mathbb{G}_1$ (resp. multi-exponentiation in $\mathbb{G}_T$), which are the most costly operations. One solution to improve multi-scalar multiplication (resp. multi-exponentiation) is to use the generalization of the Shamir's trick which is presented in [9]. In that case, the computation of $c = \sum_{i=1}^{\ell} [e_i]g_i$ (resp. $c = \prod_{i=1}^{\ell} g_i^{e_i}$) is accelerated since it is not necessary to compute each scalar multiplication (resp. modular exponentiation) and add (resp. multiply) the results since c can be computed globally. Using such trick, the computation of c needs approximately $\frac{2^{\ell+1}-1}{3\times 2^{\ell-1}}$ times the cost of a scalar multiplication (resp. modular exponentiation).

Thus, for a triple scalar multiplication in $\mathbb{G}_1$ (resp. modular exponentiation in $\mathbb{G}_T$), the expected time complexity is approximately 1.25ϵ (resp. 1.25ζ).

2.3 BBS on a Restricted Device

Using the above benchmark for pairings and operations in the different used groups, the whole BBS group signature represents the following complexity (neglecting random generation and multiplications in $\mathbb{Z}_p$): 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$, 1 triple exponentiation in $\mathbb{G}_T$, and 1 pairing, that is approximately (using the generalization of Shamir's trick) $7.3\epsilon + 1.25\zeta + \psi$. Using our above benchmark, we obtain an estimate of approximately 157 ms for this solution.

3 From a Bilinear Pairing to an Exponentiation in $\mathbb{G}_T$

We now give one possibility to implement such group signature, which is in particular explained in the paragraph "Performance" of Section 6 in [2].

3.1 Removing the Bilinear Pairing

One possibility is to consider that the group manager, when generating the tuple (A, x) for the group member, already pre-computes $\Lambda = e(A, g_2)$ and gives it to the group member. Then, using Λ and assuming that $\tilde{w} = e(h, w)$ and $\tilde{h} = e(h, g_2)$ are already precomputed (see above), we see easily that

$$\Lambda \tilde{h}^{\alpha+\beta} = e(A, g_2)e(h, g_2)^{\alpha+\beta} = e(A + [\alpha + \beta]h, g_2) = e(T_3, g_2), \qquad (2)$$

which corresponds to the result we need. Then, Equation (11) becomes (in $\mathbb{G}_T$)

$$t_5 = \Lambda^{r_x} \tilde{w}^{-r_\alpha - r_\beta} \tilde{h}^{r_x(\alpha+\beta) - r_{\delta_1} - r_{\delta_2}}$$

which is a new way for the group member (the mobile phone) to compute t_5.

As a result, using such technique, the group member has now to perform 7 random generations, 5 exponentiations in $\mathbb{G}_1$, 2 double exponentiations in $\mathbb{G}_1$, 1 triple exponentiation in $\mathbb{G}_T$, and *no pairing evaluations*.

3.2 Pros and Cons

Implementation. The main advantage of this method is that it is not necessary to embed a pairing on the mobile phone, since the phone does not have to compute any more pairing. However, in practice, this advantage is not as important as it seems since the mobile needs to perform multi-exponentiations in $\mathbb{G}_T$ and in $\mathbb{G}_1$. Thus, it is necessary to implement the algebraic structure of a bilinear pairing, without implementing a bilinear pairing. Thus, regarding the pure implementation aspects, the gain is not very important.

Efficiency. Regarding efficiency, the computation of the bilinear pairing plus a triple exponentiation is replaced by only a triple exponentiation (and no pairing!). The other operations are unchanged, except for some extra operations in $\mathbb{Z}_p$ which we neglect throughout the text (e.g. computing $r_x(\alpha + \beta) - r_{\delta_1} - r_{\delta_2}$).

In total, we obtain: 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$ and 1 triple exponentiation in $\mathbb{G}_T$, that is $7.3\epsilon + 1.25\zeta$. Using our above benchmark, we obtain an estimate of 94 ms for this solution.

Remark 1. As the most costly operation for a pairing is the final exponentiation in $\mathbb{G}_T$, the gain is not always as important as it can be (for other schemes than BBS). The different optimizations for this final exponentiation need to be compared to the existing methods regarding multi-exponentiation (see above). Then, depending on the number of components in the multi-exponentiations, the results regarding efficiency can be different.

4 Delegating the Pairing Computation

We here give another possibility which consists in delegating the computation of the bilinear pairings to a more powerful entity.

4.1 How to Delegate

We first remark that the pairing computation which we focus on is $e(T_3, g_2)$, where T_3 is part of the output group signature (and is consequently a public value since the group signature is public) and g_2 is a public parameter. As our pairing needs no secret key and as it takes on input public values, the output is also public (any verifier can compute it after the reception of a group signature). Our idea is then to delegate this computation to another entity. This entity can correspond to *e.g.* a more powerful laptop, or some kind of dedicated server (*e.g.* a *cloud* for cryptographic operations) where it is easier to implement a fast bilinear pairing. Another example is when the true group member corresponds to the SIM card while the external helper is the mobile phone.

After having computed T_3 (see above in Section 2.1), the mobile phone can send it to this powerful entity which computes $\tilde{t} = e(T_3, g_2)$ and sends the results to the mobile phone. In this case, Equation (1) becomes (in $\mathbb{G}_T$)

$$t_5 = \tilde{t}^{r_x} e(h, w)^{-r_\alpha - r_\beta} e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}. \tag{3}$$

In this case, the mobile phone has to perform 7 random generations, 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$, 1 triple exponentiation in $\mathbb{G}_T$, and *no pairing evaluations*.

4.2 Pros and Cons

Implementation. Again, in this solution, there is no need to implement a bilinear pairing in the mobile phone. Again also, it is still necessary to implement most of the algebraic structure of the bilinear pairing. Thus, regarding the pure implementation aspects, the gain is still not very important.

Efficiency. Regarding efficiency, the computation of the bilinear pairing plus a triple exponentiation is replaced by the sole triple exponentiation. However, we need to take into account the additional communication steps of this method (the computation of the bilinear pairing by the powerful entity is not taken into account as it can be executed in parallel with other computations performed by the mobile phone). In the previous methods, there is only one communication step of the whole group signature. Here, we add an additional communication for sending and receiving T_3 and $\tilde{t}$ respectively.

Practically speaking, we obtain (again neglecting random generation and multiplications in $\mathbb{Z}_p$): 5 scalar multiplications in $\mathbb{G}_1$, 2 double-scalar multiplications in $\mathbb{G}_1$ and 1 triple exponentiation in $\mathbb{G}_T$, which corresponds exactly to the above time complexity. This time, we also only need to compute additions in $\mathbb{Z}_p$ (and no multiplications as for the previous solution).

However, this does not include the additional communication. In practice, we can approximate the communication between a mobile phone and the exterior or between the SIM card and the mobile phone to 200 kbits/s (and thus approximately 17 ms for the communication in our case). Using an 3G/UMTS

communication, the resulting rate can reach 2 Mbits/s, which makes the communication step negligible.

Facing corrupted delegate. One additional problem with this method is that the powerful delegate can send to the mobile phone a wrong value $\tilde{t}$ so that the resulting group signature will be rejected. There exists in the literature verifiable delegation of cryptographic operation, but, to the best of our knowledge, no work has been done regarding pairings.

However, in some practical cases, this "attack" is not useful. In particular, if the group signature generated by the mobile signature necessarily goes via this delegate for the final sending to the true verifier, this one can easily send to the verifier anything it wants to get the group signature rejected. If such group signature is used for *e.g.* access control, the customer will see that she can not access the place she wants and thus detect that something is wrong.

Remark 2. In some cases, such as for the identity-based encryption scheme of Boneh-Franklin [3], the pairing evaluation includes a secret value, which may make this method impossible. In fact, as proposed by Lefranc and Girault [8], there exists some delegation technique for this case. For example, if one wants to compute $e(a, b)$ where a is secret and b is public, one possibility is for the mobile phone to compute $c = [\alpha]a$, where α is random, and for the delegate $d = e(c, b)$. The result $e(a, b)$ is then $d^{1/\alpha}$.

5 Conclusion

With our practical results, it seems that the second solution, which consists in replacing the bilinear pairing by operations in $\mathbb{G}_T$, is the best one. This also shows that the optimizations regarding operations in $\mathbb{G}_T$ are very important, as well as the communication rate between *e.g.* the mobile phone and the exterior. Note finally that the delegation technique can be extended to other operations related to the BBS signature scheme, such as proposed in [4], which can make the last solution better in some particular cases.

Acknowledgments. We are very grateful to Tanja Lange for her useful comments and suggestions.

References

1. Beuchat, J.-L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
2. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
3. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. SIAM J. Comput. 32(3), 586–615 (2003)

4. Canard, S., Coisel, I., De Meulenaer, G., Pereira, O.: Group Signatures are Suitable for Constrained Devices. In: Rhee, K.-H., Nyang, D. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 133–150. Springer, Heidelberg (2011)
5. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
6. Cheng, Z., Nistazakis, M.: Implementing Pairing-Based Cryptosystems. In: Proceedings of IWWST 2005 (2005)
7. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
8. Girault, M., Lefranc, D.: Server-Aided Verification: Theory and Practice. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 605–623. Springer, Heidelberg (2005)
9. Möller, B.: Algorithms for Multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)

The Tate-Lichtenbaum Pairing on a Hyperelliptic Curve via Hyperelliptic Nets

Yukihiro Uchida and Shigenori Uchiyama

Department of Mathematics and Information Sciences,
Tokyo Metropolitan University,
1-1 Minami-Osawa, Hachioji, Tokyo 192-0397, Japan
{yuchida,uchiyama-shigenori}@tmu.ac.jp

Abstract. Recently, Stange proposed a new algorithm to compute the Tate pairing on an elliptic curve. Her algorithm is based on elliptic nets, which are also defined by Stange as a generalization of elliptic divisibility sequences. In this paper, we define hyperelliptic nets as a generalization of elliptic nets to hyperelliptic curves. We also give an expression for the Tate-Lichtenbaum pairing on a hyperelliptic curve in terms of hyperelliptic nets. Using this expression, we give an algorithm to compute the Tate-Lichtenbaum pairing on a hyperelliptic curve of genus 2.

Keywords: Tate-Lichtenbaum pairing, hyperelliptic curve, hyperelliptic net.

1 Introduction

In the area of elliptic and hyperelliptic curve cryptography, various pairings such as the Weil and Tate-Lichtenbaum pairings play an important role. It is an important problem to find an algorithm to compute pairings efficiently. Among these pairings, the Tate-Lichtenbaum pairing is studied well because of its efficiency.

The Tate-Lichtenbaum pairing is usually computed by using Miller's algorithm as well as the Weil pairing (cf. [12,13]). Recently, Stange [17] proposed another algorithm to compute the Tate(-Lichtenbaum) pairing on an elliptic curve. Her algorithm is based on elliptic nets, which are defined as follows: Let A be a free finitely generated Abelian group and R be an integral domain. A map $W\colon A \to R$ is an *elliptic net* if $W(0) = 0$ and if

$$\begin{aligned} &W(p+q+s)W(p-q)W(r+s)W(r) \\ &\quad + W(q+r+s)W(q-r)W(p+s)W(p) \\ &\quad\quad + W(r+p+s)W(r-p)W(q+s)W(q) = 0 \end{aligned} \tag{1}$$

holds for all $p, q, r, s \in A$. Stange defined the elliptic net $W_{\boldsymbol{P}}$ associated to an elliptic curve E and points $\boldsymbol{P} = (P_1, \ldots, P_n)$ on E, and described the Tate pairing on E by using $W_{\boldsymbol{P}}$.

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 218–233, 2013.

In this paper, we generalize Stange's results to hyperelliptic curves. First, we define a hyperelliptic net associated to a hyperelliptic curve and points on its Jacobian variety and prove properties of hyperelliptic nets. Next, we give an expression for the Tate-Lichtenbaum pairing on a hyperelliptic curve in terms of hyperelliptic nets. Finally, we give an algorithm to compute the Tate-Lichtenbaum pairing on a hyperelliptic curve of genus 2 via hyperelliptic nets.

This paper is organized as follows: In Sect. 2, we review the theory of the hyperelliptic sigma function. In Sect. 3, we define hyperelliptic nets over the complex numbers and describe some properties of them. In Sect. 4, we define hyperelliptic nets over arbitrary fields by using an existence theorem of certain rational functions on a product of Jacobian varieties (Theorem 2), which is proved in the Appendix. In Sect. 5, we give an expression of the Tate-Lichtenbaum pairing on a hyperelliptic curve in terms of hyperelliptic nets. In Sect. 6, we give algorithms to compute terms of hyperelliptic nets and the Tate-Lichtenbaum pairing in the case of genus 2. Finally, we draw conclusions in Sect. 7.

Notation. For a matrix A, we denote by ${}^{\mathrm{t}}A$ the transpose of A. Unless otherwise stated, we regard a vector as a column vector. For a $2n \times 2n$ skew-symmetric matrix A, we denote by $\mathrm{Pf}\, A$ the Pfaffian of A.

2 The Hyperelliptic Sigma Function

In this section, we review the theory of the hyperelliptic sigma function and fix the definitions. For details, we refer the reader to [3,16].

Let C be a smooth projective curve over $\mathbb{C}$ defined by

$$y^2 = f(x) = x^{2g+1} + \lambda_{2g}x^{2g} + \cdots + \lambda_1 x + \lambda_0.$$

Then $f(x)$ has no multiple roots, the genus of C is g, and C has the unique point ∞ at infinity. We define $\lambda_{2g+1} = 1$ for convention.

For $j = 1, 2, \ldots, g$, we define differential forms

$$\omega_j = \frac{x^{j-1}dx}{2y}, \quad \eta_j = \frac{1}{2y}\sum_{k=j}^{2g-j}(k+1-j)\lambda_{k+1+j}x^k dx.$$

Let $[\omega', \omega'']$ and $[\eta', \eta'']$ be the period matrices with respect to ω_j and η_j for a symplectic basis of $H_1(C, \mathbb{Z})$ respectively. Let $\tau = \omega'^{-1}\omega''$. It is known that τ is symmetric and $\mathrm{Im}\,\tau$ is positive definite.

We write $\boldsymbol{e}(z) = \exp(2\pi\sqrt{-1}z)$ for $z \in \mathbb{C}$. We define the theta function with characteristics by

$$\vartheta\begin{bmatrix} a \\ b \end{bmatrix}(z, \tau) = \sum_{n \in \mathbb{Z}^g} \boldsymbol{e}\left(\frac{1}{2}{}^{\mathrm{t}}(n+a)\tau(n+a) + {}^{\mathrm{t}}(n+a)(z+b)\right),$$

where $z \in \mathbb{C}^g$ and $a, b \in \mathbb{R}^g$.

Let $\Lambda = \omega'\mathbb{Z}^g + \omega''\mathbb{Z}^g$. Then Λ is a lattice in $\mathbb{C}^g$. Let

$$\delta'' = {}^{\mathrm{t}}\left(\frac{1}{2}, \ldots, \frac{1}{2}\right), \quad \delta' = {}^{\mathrm{t}}\left(\frac{g}{2}, \frac{g-1}{2}, \ldots, \frac{1}{2}\right), \quad \delta = \begin{pmatrix}\delta'' \\ \delta'\end{pmatrix}.$$

We define the hyperelliptic sigma function on $\mathbb{C}^g$ by

$$\sigma(u) = c \exp\left(\frac{1}{2}{}^{\mathrm{t}}u\eta'\omega'^{-1}u\right)\vartheta[\delta](\omega'^{-1}u, \tau),$$

where c is a certain constant. For the precise definition, see [16, Sect. 4].

Let $l = \omega'l' + \omega''l'' \in \Lambda$, where $l', l'' \in \mathbb{Z}^g$. We define $\chi\colon \Lambda \to \{\pm 1\}$ by

$$\chi(l) = \exp\left(2\pi\sqrt{-1}\left({}^{\mathrm{t}}l'\delta'' - {}^{\mathrm{t}}l''\delta'\right) + \pi\sqrt{-1}\,{}^{\mathrm{t}}l'l''\right).$$

Proposition 1 (translational relation). *Let $u \in \mathbb{C}^g$ and $l = \omega'l' + \omega''l'' \in \Lambda$, where $l', l'' \in \mathbb{Z}^g$. Then we have*

$$\sigma(u + l) = \chi(l) \exp\left({}^{\mathrm{t}}(u + l/2)(\eta'l' + \eta''l'')\right)\sigma(u).$$

Proof. See [3, Theorem 1.1]. □

Proposition 2. *The sigma function $\sigma(u)$ is an odd function if $g \equiv 1, 2 \pmod 4$, and an even function if $g \equiv 0, 3 \pmod 4$.*

Proof. It follows from [14, Chap. II, Proposition 3.14]. □

We define the hyperelliptic $\wp$-functions by

$$\wp_{ij}(u) = -\frac{\partial^2}{\partial u_i \partial u_j}\log\sigma(u), \quad \wp_{ijk}(u) = -\frac{\partial^3}{\partial u_i \partial u_j \partial u_k}\log\sigma(u), \ \ldots,$$

where $i, j, k, \ldots \in \{1, 2, \ldots, g\}$ and $u = {}^{\mathrm{t}}(u_1, \ldots, u_g)$. Obviously, the $\wp$-functions do not depend on the order of their indices. For example, $\wp_{ij}(u) = \wp_{ji}(u)$.

Let $J = \mathbb{C}^g/\Lambda$ be the Jacobian variety of C and $\kappa\colon \mathbb{C}^g \to J$ be the natural projection. By Proposition 1, the $\wp$-functions are periodic with respect to Λ. Thus we may regard $\wp_{ij}$, $\wp_{ijk}$, ... as meromorphic functions on J. We write $\wp_{ij}(P) = \wp_{ij}(u)$ for $P = \kappa(u)$.

Let $\lambda\colon C \to J$ be an embedding such that $\lambda(\infty) = O$. We define the theta divisor Θ on J by $\Theta = \lambda(C) + \cdots + \lambda(C)$ $(g-1$ times$)$.

Proposition 3. *The divisor of $\sigma(u)$ is $\kappa^{-1}(\Theta)$.*

Proof. See [15, pp. 3.80–82]. □

We define a function $\mathcal{F}_g$ on $\mathbb{C}^g \times \mathbb{C}^g$ by

$$\mathcal{F}_g(u,v) = \frac{\sigma(u+v)\sigma(u-v)}{\sigma(u)^2\sigma(v)^2}.$$

Buchstaber, Enolskii, and Leykin [4, Theorem 3.3] proved that $\mathcal{F}_g$ is explicitly expressed as a polynomial in the $\wp$-functions. Therefore we may regard $\mathcal{F}_g$ as a rational function on $J \times J$. When $P = \kappa(u)$ and $Q = \kappa(v)$, we write $\mathcal{F}_g(P,Q) = \mathcal{F}_g(u,v)$.

Finally, we review a classical relation satisfied by the sigma function.

Proposition 4 (Weierstrass, Frobenius, Caspary). *Let $n > 2^g$ be an integer and $u^{(1)}, u^{(2)}, \ldots, u^{(n)} \in \mathbb{C}^g$. We define an $n \times n$ matrix A by*

$$A = \left(\sigma(u^{(i)} + u^{(j)})\sigma(u^{(i)} - u^{(j)})\right)_{1 \le i,j \le n}.$$

Then we have $\det A = 0$. *In particular, if $g \equiv 1, 2 \pmod 4$ and n is even, then we have* $\mathrm{Pf}\, A = 0$.

Proof. See [19, Corollary 6.2] and the literature cited there. □

3 Hyperelliptic Nets over the Complex Numbers

We continue to use the notation in Sect. 2. Let n be a positive integer.

Definition 1. *For $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{Z}^n$, we define a meromorphic function $\Phi_{\boldsymbol{v}} \colon (\mathbb{C}^g)^n \to \mathbb{C}$ by*

$$\Phi_{\boldsymbol{v}}(u^{(1)}, \ldots, u^{(n)}) = \frac{\sigma(v_1 u^{(1)} + \cdots + v_n u^{(n)})}{\prod_{i=1}^n \sigma(u^{(i)})^{2v_i^2 - \sum_{j=1}^n v_i v_j} \prod_{1 \le i < j \le n} \sigma(u^{(i)} + u^{(j)})^{v_i v_j}}.$$

This definition is a generalization of the definition of Stange's net polynomial $\Omega_{\boldsymbol{v}}$ (see [18, Definition 3.1]). When $n = 1$, the functions $\Phi_{\boldsymbol{v}}$ coincide with the division polynomials defined in [8,9,19].

We describe some properties of $\Phi_{\boldsymbol{v}}$.

Proposition 5. *For any $\boldsymbol{v} \in \mathbb{Z}^n$, the function $\Phi_{\boldsymbol{v}}$ is periodic with respect to Λ in each variable.*

Proof. The proposition follows from Proposition 1. □

By Proposition 5, we may regard $\Phi_{\boldsymbol{v}}$ as a meromorphic function on J^n. When $P_i = \kappa(u^{(i)})$, we write $\Phi_{\boldsymbol{v}}(P_1, \ldots, P_n) = \Phi_{\boldsymbol{v}}(u^{(1)}, \ldots, u^{(n)})$.

Proposition 6. *For any $\boldsymbol{v} \in \mathbb{Z}^n$, $\Phi_{-\boldsymbol{v}} = -\Phi_{\boldsymbol{v}}$ if $g \equiv 1, 2 \pmod 4$, and $\Phi_{-\boldsymbol{v}} = \Phi_{\boldsymbol{v}}$ if $g \equiv 0, 3 \pmod 4$.*

Proof. The proposition immediately follows from Proposition 2. □

Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ be the standard basis of $\mathbb{Z}^n$.

Proposition 7. *Let $\boldsymbol{v} \in \mathbb{Z}^n$.*

(a) The function $\Phi_{\boldsymbol{v}}$ identically equals zero if and only if $\boldsymbol{v} = \boldsymbol{0}$.
(b) If $\boldsymbol{v} = \boldsymbol{e}_i$ or $\boldsymbol{v} = \boldsymbol{e}_i + \boldsymbol{e}_j$ with $i \neq j$, then $\Phi_{\boldsymbol{v}} = 1$.
(c) Let $\boldsymbol{P} = (P_1, \ldots, P_n) \in J^n$. If $i \neq j$, then $\Phi_{\boldsymbol{e}_i - \boldsymbol{e}_j}(\boldsymbol{P}) = \mathcal{F}_g(P_i, P_j)$.

Proof. (a) follows from Proposition 3. (b) and (c) are clear by definition. □

Proposition 8. *Let $\boldsymbol{P} = (P_1, \ldots, P_n) \in J^n$ and $\boldsymbol{v} \in \mathbb{Z}^m$. Let $T = (t_{ij})$ be an $n \times m$ matrix with entries in $\mathbb{Z}$. Then we have*

$$\Phi_{\boldsymbol{v}}(\boldsymbol{P}T) = \frac{\Phi_{T\boldsymbol{v}}(\boldsymbol{P})}{\prod_{i=1}^m \Phi_{T\boldsymbol{e}_i}(\boldsymbol{P})^{2v_i^2 - \sum_{j=1}^m v_i v_j} \prod_{1 \leq i < j \leq m} \Phi_{T(\boldsymbol{e}_i + \boldsymbol{e}_j)}(\boldsymbol{P})^{v_i v_j}},$$

where $\boldsymbol{P}T$ stands for $([t_{11}]P_1 + \cdots + [t_{n1}]P_n, \ldots, [t_{1m}]P_1 + \cdots + [t_{nm}]P_n)$.

Proof. The proposition follows from Definition 1 by a direct calculation. □

Proposition 9. *Let $\boldsymbol{P} = (P_1, \ldots, P_n) \in J^n$ and $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{Z}^n$. If $\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{v} + \boldsymbol{w}, \boldsymbol{v} - \boldsymbol{w} \neq \boldsymbol{0}$, then we have*

$$\frac{\Phi_{\boldsymbol{v}+\boldsymbol{w}}(\boldsymbol{P})\Phi_{\boldsymbol{v}-\boldsymbol{w}}(\boldsymbol{P})}{\Phi_{\boldsymbol{v}}(\boldsymbol{P})^2 \Phi_{\boldsymbol{w}}(\boldsymbol{P})^2} = \mathcal{F}_g([v_1]P_1 + \cdots + [v_n]P_n, [w_1]P_1 + \cdots + [w_n]P_n).$$

Proof. The proposition easily follows from Definition 1. □

The functions $\Phi_{\boldsymbol{v}}$ satisfy the following recurrence formula.

Theorem 1. *Let $m > 2^g$ be an integer and $\boldsymbol{v}^{(i)} \in ((1/2)\mathbb{Z})^n$ for $1 \leq i \leq m$. We assume that $\boldsymbol{v}^{(i)} + \boldsymbol{v}^{(j)}, \boldsymbol{v}^{(i)} - \boldsymbol{v}^{(j)} \in \mathbb{Z}^n$ for all $1 \leq i, j \leq m$. We define an $m \times m$ matrix A by*

$$A = \left(\Phi_{\boldsymbol{v}^{(i)} + \boldsymbol{v}^{(j)}} \Phi_{\boldsymbol{v}^{(i)} - \boldsymbol{v}^{(j)}}\right)_{1 \leq i,j \leq m}.$$

Then we have $\det A = 0$. In particular, if $g \equiv 1, 2 \pmod 4$ and m is even, then we have $\operatorname{Pf} A = 0$.

Remark 1. When $n = 1$, Theorem 1 was proved in [19, Theorem 6.4].

Proof. The theorem follows from Proposition 4. □

Example 1. We consider the case where $g = 1$ and $m = 4$. Then we have

$$\begin{aligned}\operatorname{Pf} A = {} & \Phi_{\boldsymbol{v}^{(1)} + \boldsymbol{v}^{(2)}} \Phi_{\boldsymbol{v}^{(1)} - \boldsymbol{v}^{(2)}} \Phi_{\boldsymbol{v}^{(3)} + \boldsymbol{v}^{(4)}} \Phi_{\boldsymbol{v}^{(3)} - \boldsymbol{v}^{(4)}} \\ & - \Phi_{\boldsymbol{v}^{(1)} + \boldsymbol{v}^{(3)}} \Phi_{\boldsymbol{v}^{(1)} - \boldsymbol{v}^{(3)}} \Phi_{\boldsymbol{v}^{(2)} + \boldsymbol{v}^{(4)}} \Phi_{\boldsymbol{v}^{(2)} - \boldsymbol{v}^{(4)}} \\ & + \Phi_{\boldsymbol{v}^{(1)} + \boldsymbol{v}^{(4)}} \Phi_{\boldsymbol{v}^{(1)} - \boldsymbol{v}^{(4)}} \Phi_{\boldsymbol{v}^{(2)} + \boldsymbol{v}^{(3)}} \Phi_{\boldsymbol{v}^{(2)} - \boldsymbol{v}^{(3)}} = 0.\end{aligned}$$

Let $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}, \boldsymbol{s} \in \mathbb{Z}^n$ and

$$\boldsymbol{v}^{(1)} = \boldsymbol{p} + \frac{1}{2}\boldsymbol{s}, \quad \boldsymbol{v}^{(2)} = \boldsymbol{q} + \frac{1}{2}\boldsymbol{s}, \quad \boldsymbol{v}^{(3)} = \boldsymbol{r} + \frac{1}{2}\boldsymbol{s}, \quad \boldsymbol{v}^{(4)} = \frac{1}{2}\boldsymbol{s}.$$

Then we have

$$\Phi_{\boldsymbol{p}+\boldsymbol{q}+\boldsymbol{s}}\Phi_{\boldsymbol{p}-\boldsymbol{q}}\Phi_{\boldsymbol{r}+\boldsymbol{s}}\Phi_{\boldsymbol{r}} + \Phi_{\boldsymbol{q}+\boldsymbol{r}+\boldsymbol{s}}\Phi_{\boldsymbol{q}-\boldsymbol{r}}\Phi_{\boldsymbol{p}+\boldsymbol{s}}\Phi_{\boldsymbol{p}} + \Phi_{\boldsymbol{r}+\boldsymbol{p}+\boldsymbol{s}}\Phi_{\boldsymbol{r}-\boldsymbol{p}}\Phi_{\boldsymbol{q}+\boldsymbol{s}}\Phi_{\boldsymbol{q}} = 0.$$

This formula coincides with the recurrence formula (1) defining Stange's elliptic nets. In other words, Theorem 1 is a generalization of Stange's recurrence formula.

The divisor of the function $\Phi_{\boldsymbol{v}}$ is computed as follows. Let $p_i\colon J^n \to J$ be the i-th projection and $s_m\colon J^m \to J$ be the summation of all components. Let

$$D_{J,\boldsymbol{v}} = ([v_1] \times \cdots \times [v_n])^* s_n^* \Theta - \sum_{1 \le k < l \le n} v_k v_l (p_k \times p_l)^* s_2^* \Theta - \sum_{k=1}^{n} \left(2v_k^2 - \sum_{l=1}^{n} v_k v_l \right) p_k^* \Theta.$$

Proposition 10. *For $\boldsymbol{v} \in \mathbb{Z}^n \setminus \{\boldsymbol{0}\}$, we have $\operatorname{div}(\Phi_{\boldsymbol{v}}) = D_{J,\boldsymbol{v}}$.*

Proof. The proposition follows from Proposition 3. □

Proposition 11. *For any $\boldsymbol{v} \in \mathbb{Z}^n$, $\Phi_{\boldsymbol{v}}$ is a rational function on J^n defined over $\mathbb{Q}(\lambda_0, \ldots, \lambda_{2g})$.*

Proof. By the Frobenius-Stickelberger-type formula ([16, Theorem 8.2]), we obtain a determinant expression for $\Phi_{\boldsymbol{v}}$ (when $n = 1$, it is given in [19, Theorem 5.5]). The proposition follows from this determinant expression. □

4 Hyperelliptic Nets over Arbitrary Fields

In this section, we define hyperelliptic nets over arbitrary fields. Let K be an arbitrary field. Let C be a smooth projective hyperelliptic curve of genus g defined over K defined by

$$y^2 + (b_g x^g + \cdots + b_0)y = x^{2g+1} + a_{2g}x^{2g} + \cdots + a_0,$$

where $a_0, \ldots, a_{2g}, b_0, \ldots, b_g \in K$. The curve C has the unique point ∞ at infinity. Let J be the Jacobian variety of C. Let Θ be the theta divisor as in Sect. 2.

Let n be a positive integer. In the previous section, we have defined a rational function $\Phi_{\boldsymbol{v}}$ on J^n for any $\boldsymbol{v} \in \mathbb{Z}^n$ under the assumption that C is defined over $\mathbb{C}$ and $b_0 = \cdots = b_g = 0$. We will define a similar rational function on J^n when C is defined over an arbitrary field K.

We first explain that the function $\mathcal{F}_g$ on $J \times J$ can be defined over an arbitrary field. Arledge and Grant [1] constructed a rational function H on $J \times J$ over an arbitrary field such that $\operatorname{div}(H) = s^*\Theta + d^*\Theta - 2p_1^*\Theta - 2p_2^*\Theta$, where $s, d\colon J \times J \to J$ are defined by $s(P, Q) = P + Q$ and $d(P, Q) = P - Q$. They constructed the function H by using determinants. By [16, Theorem 8.2], we obtain $\mathcal{F}_g(P, Q) =$

$\pm H(P,Q)$ over $\mathbb{C}$. Therefore, we can define the function $\mathcal{F}_g$ over an arbitrary field by using the function H.

We can also define $\Phi_{\boldsymbol{v}}$ over arbitrary fields. More precisely, we have the following theorem, whose proof is given in the Appendix.

Theorem 2. *There exists a family $\{\Phi_{\boldsymbol{v}}\}_{\boldsymbol{v}\in\mathbb{Z}^n}$ of rational functions on J^n defined over K such that Propositions 6–10 and Theorem 7 also hold for $\{\Phi_{\boldsymbol{v}}\}$.*

Hyperelliptic nets associated to hyperelliptic curves are defined as follows:

Definition 2. *Let $P_1,\dots,P_n \in J(K)$ with $P_i \notin \Theta$ for $1 \le i \le n$ and $P_i+P_j \notin \Theta$ for $1 \le i < j \le n$. We define $W_{P_1,\dots,P_n}\colon \mathbb{Z}^n \to K$ by*

$$W_{P_1,\dots,P_n}(\boldsymbol{v}) = \Phi_{\boldsymbol{v}}(P_1,\dots,P_n),$$

where $\Phi_{\boldsymbol{v}}$ is the function as in Theorem 2. We call $W_{P_1,\dots,P_n}$ the hyperelliptic net associated to C and $P_1,\dots,P_n$.

When $g=1$, the map $W_{P_1,\dots,P_n}$ is an elliptic net by Example 1.

Remark 2. Stange [17,18] defined elliptic nets by the recurrence (1). Then she proved that there exists a bijection between the set of elliptic nets and the set of elliptic curves with specified points [18, Theorem 7.4]. However, in the case of higher genera, it would be an open problem to show the existence of such a bijection. On the other hand, we can define a map from the set of hyperelliptic curves with n points on their Jacobian varieties to the set of maps $\mathbb{Z}^n \to K$ which satisfy a certain recurrence. It is sufficient for computing the Tate-Lichtenbaum pairing.

5 The Tate-Lichtenbaum Pairing

In this section, we first review the Tate-Lichtenbaum paring. Then we give an expression for the Tate-Lichtenbaum paring in terms of hyperelliptic nets.

Let K be a finite field with q elements and C be a projective irreducible smooth curve of genus g defined over K. We assume that C has a K-rational point ∞. Let m be a positive integer such that m divides $q-1$.

Let D be a divisor with $\overline{D} \in \mathrm{Pic}^0(C)[m]$. Since mD is linearly equivalent to 0, there exists a rational function f_D such that $mD = \mathrm{div}(f_D)$. For $\overline{E} \in \mathrm{Pic}^0(C)/m\,\mathrm{Pic}^0(C)$, choose a divisor $E = \sum_{i=1}^r n_i P_i$ such that D and E have no common points. We define $f_D(E) = \prod_{i=1}^r f_D(P_i)^{n_i}$.

Definition 3. *We define the* Tate-Lichtenbaum pairing

$$\tau_m\colon \mathrm{Pic}^0(C)[m] \times \mathrm{Pic}^0(C)/m\,\mathrm{Pic}^0(C) \to K^\times/(K^\times)^m$$

by

$$\tau_m(\overline{D},\overline{E}) = f_D(E).$$

Theorem 3. *The Tate-Lichtenbaum pairing τ_m is well-defined, bilinear, and non-degenerate.*

Proof. See [6, Sect. 1, Theorem]. □

Let J be the Jacobian variety of C and $\lambda\colon C \to J$ be an embedding such that $\lambda(\infty) = O$. We identify $J(K)$ with $\mathrm{Pic}^0(C)$ by extending λ. Then the Tate-Lichtenbaum pairing τ_m may be regarded as a pairing on the Jacobian variety

$$J(K)[m] \times J(K)/mJ(K) \to K^\times/(K^\times)^m.$$

For a point $P \in J$, we define a translation map $t_P\colon J \to J$ by $t_P(Q) = P + Q$. Let Θ be the theta divisor as in the previous section.

We rewrite the Tate-Lichtenbaum pairing by using rational functions on the Jacobian variety.

Lemma 1. *Let $P \in J(K)[m]$ and $Q \in J(K)$. Let $D = t^*_{-P}\Theta - \Theta$. Then mD is linearly equivalent to 0. Take a rational function f_P on J defined over K such that $\mathrm{div}(f_P) = mD$. Let $R, S \in (J \setminus \mathrm{supp}(D))(K)$ with $Q = R - S$. Then we have*

$$\tau_m(P, Q) = \frac{f_P(R)}{f_P(S)}$$

in $K^\times/(K^\times)^m$.

Proof. The lemma was implicitly proved by Lichtenbaum [10, pp. 126–127]. □

Stange [17, Theorem 6] proved that the Tate(-Lichtenbaum) pairing on an elliptic curve is expressed in terms of elliptic nets. We generalize it to the case of hyperelliptic curves.

Theorem 4. *We assume that the curve C is a hyperelliptic curve. Let $P, Q \in J(K)$ with $[m]P = O$. We assume that $P, Q, P + Q \notin \Theta$. Choose $S \in J(K)$ such that $S, S + P, S + Q \notin \Theta$. Let $W_{S,P,Q}$ be the hyperelliptic net associated to C, S, P, Q. Then we have*

$$\tau_m(P, Q) = \frac{W_{S,P,Q}(1, m, 1)W_{S,P,Q}(1, 0, 0)}{W_{S,P,Q}(1, m, 0)W_{S,P,Q}(1, 0, 1)}$$

in $K^\times/(K^\times)^m$.

Proof. Let

$$f_P(T) = \frac{\Phi_{1,0,0}(-T, P, Q)}{\Phi_{1,m,0}(-T, P, Q)}.$$

By Proposition 10, we have

$$\mathrm{div}(f_P) = -t^*_{-[m]P}\Theta + (1 - m)\Theta + mt^*_{-P}\Theta = m\left(t^*_{-P}\Theta - \Theta\right).$$

Since $Q = (-S) - (-S - Q)$, by Proposition 8 and Lemma 1,

$$\begin{aligned}\tau_m(P,Q) &= \frac{\Phi_{1,0,0}(S,P,Q)\Phi_{1,m,0}(S+Q,P,Q)}{\Phi_{1,m,0}(S,P,Q)\Phi_{1,0,0}(S+Q,P,Q)} \\ &= \frac{\Phi_{1,0,0}(S,P,Q)\Phi_{1,m,1}(S,P,Q)}{\Phi_{1,m,0}(S,P,Q)\Phi_{1,0,1}(S,P,Q)} \\ &= \frac{W_{S,P,Q}(1,m,1)W_{S,P,Q}(1,0,0)}{W_{S,P,Q}(1,m,0)W_{S,P,Q}(1,0,1)}\end{aligned}$$

in $K^\times/(K^\times)^m$. □

Corollary 1. *Under the assumption of Theorem 4, let $W_{P,Q}$ be the hyperelliptic net associated to C, P, Q. Then we have*

$$\tau_m(P,Q) = \frac{W_{P,Q}(m+1,1)W_{P,Q}(1,0)}{W_{P,Q}(m+1,0)W_{P,Q}(1,1)}$$

in $K^\times/(K^\times)^m$. Let W_P be the hyperelliptic net associated to C, P. If $[2]P \notin \Theta$, we have

$$\tau_m(P,P) = \frac{W_P(m+2)W_P(1)}{W_P(m+1)W_P(2)}$$

in $K^\times/(K^\times)^m$.

Proof. The corollary follows from Proposition 8 and Theorem 4 by putting $S = P$. □

6 Algorithms for Curves of Genus 2

In this section, we give algorithms to compute terms of hyperelliptic nets and the Tate-Lichtenbaum pairing on a curve of genus 2. Our algorithms are generalizations of Stange's algorithms [17, Sect. 4].

We use the following notation in this section. Let K be a finite field with q elements and C be a hyperelliptic curve of genus 2 defined by

$$y^2 + (b_2x^2 + b_1x + b_0)y = x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0,$$

where $a_0, a_1, a_2, a_3, a_4, b_0, b_1, b_2 \in K$. Let J be the Jacobian variety of C and Θ be the theta divisor on J. Let $P, Q \in J(K)$ with $P, Q, P+Q \notin \Theta$ and $W_{P,Q}$ be the associated hyperelliptic net. For simplicity, we write $W(m,n) = W_{P,Q}(m,n)$.

6.1 Double and DoubleAdd

We compute the terms $W(m,0)$ and $W(m,1)$ by recurrence formulas. We first give some definitions.

We define a block V centered on k as follows:

$$\begin{aligned}V = [&[W(k-7,0), W(k-6,0), \ldots, W(k+8,0)], \\ &[W(k-3,1), W(k-2,1), \ldots, W(k+3,1)]].\end{aligned}$$

Let V be a given block centered on k. We define two functions:

(a) Double(V): Returns a block centered on $2k$.
(b) DoubleAdd(V): Returns a block centered on $2k+1$.

We compute blocks returned from these functions by the following formulas. Let A be a 6×6 skew-symmetric matrix defined by

$$A = \big(W(m_i + m_j, n_i + n_j)W(m_i - m_j, n_i - n_j)\big)_{1 \le i,j \le 6},$$

where $m_1, \ldots, m_6, n_1, \ldots, n_6 \in \mathbb{Z}$. Then, by Theorem 1, we have

$$\operatorname{Pf} A = 0. \tag{2}$$

By substituting suitable integers for $m_1, \ldots, m_6, n_1, \ldots, n_6$, we obtain recurrence formulas needed to compute terms $W(m, 0)$ and $W(m, 1)$. We use the values in Table 1, where $-3 \le j \le 4$.

Table 1. Values of the m_i and n_i

Term to compute	m_1	m_2	m_3	m_4	m_5	m_6	n_1	$n_2, \ldots, n_6$
$W(2k, 0)$	$k+1$	$k-1$	3	2	1	0	0	0
$W(2k-1, 0)$	k	$k-1$	3	2	1	0	0	0
$W(2k+j, 1)$	k	$k+j$	3	2	1	0	1	0

By the expansion formula of a Pfaffian, we can expand (2) as follows:

$$\sum_{i=2}^{6} (-1)^i W(m_1 + m_i, n_1 + n_i)W(m_1 - m_i, n_1 - n_i) \operatorname{Pf} A^{1,i} = 0,$$

where $A^{1,i}$ is the submatrix of A obtained by removing the first and i-th rows and columns. When we substitute the values in Table 1 for the m_i and n_i, the term $W(m_1 + m_2, n_1 + n_2)$ equals $W(2k, 0)$, $W(2k+1, 0)$, and $W(2k+j, 1)$ respectively. Moreover, the other terms equal one of the following:

(a) terms in the block centered on k,
(b) $W(m, 0)$ for $1 \le m \le 5$ or
(c) $W(m, 1)$ for $-4 \le m \le 3$.

Therefore we can compute the blocks centered on $2k$ and $2k+1$. Note that we need the division by $W(m_1 - m_2, n_1 - n_2) \operatorname{Pf} A^{1,2}$ to compute the terms of a hyperelliptic net. For all values in Table 1, the expression for $\operatorname{Pf} A^{1,2}$ is the same:

$$\operatorname{Pf} A^{1,2} = W(5, 0) - W(4, 0)W(2, 0)^3 + W(3, 0)^3.$$

This value depends only on P. We denote it by $\Delta(P)$, that is,

$$\Delta(P) = \operatorname{Pf} A^{1,2} = W(5, 0) - W(4, 0)W(2, 0)^3 + W(3, 0)^3.$$

The algorithm to compute $W(m, 0)$ and $W(m, 1)$ is shown in Algorithm 1. Note that Algorithm 1 requires the assumption that $W(-4, 1), W(-3, 1), \ldots, W(3, 1)$, $W(2, 0)$, and $\Delta(P)$ are all non-zero.

Algorithm 1. Hyperelliptic Net Algorithm

Input: Initial terms $W(m,0)$ for $-6 \leq m \leq 9$ and $W(m,1)$ for $-4 \leq m \leq 4$ of a hyperelliptic net and an integer $m = (d_k d_{k-1} \dots d_1)_2$ with $d_k = 1$
Output: Hyperelliptic net elements $W(m,0)$ and $W(m,1)$

```
V ← [[W(−6,0), W(−5,0), ..., W(9,0)], [W(−2,1), W(−1,1), ..., W(4,1)]]
for i = k − 1 down to 1 do
    if d_i = 0 then
        V ← Double(V)
    else
        V ← DoubleAdd(V)
    end if
end for
return V[0,7] and V[1,3]  // terms W(m,0) and W(m,1) respectively
```

6.2 Initial Values

In this subsection, we consider how to compute the initial values. In the case of genus 2, expressions for the $W(m,n)$ in terms of some coordinates of points on the Jacobian variety are huge except for small m and n. In the following, we compute the values of the $W(m,n)$ by addition on the Jacobian variety and Proposition 9.

Let $(t^2+u_{11}t+u_{12}, v_{11}t+v_{12})$ and $(t^2+u_{21}t+u_{22}, v_{21}t+v_{22})$ be the Mumford representations of P and Q respectively. We first compute $W(m,n)$ for small m and n by the following formulas:

$$W(0,0) = 0, \quad W(1,0) = W(0,1) = W(1,1) = 1,$$

$$\begin{aligned} W(2,0) = {} & (-4u_{12} + 6{u_{11}}^2 + (-{b_2}^2 - 4a_4)u_{11} + b_1b_2 + 2a_3)v_{12} \\ & + 2{v_{11}}^3 + (3b_1 - 3b_2u_{11}){v_{11}}^2 + ((-8u_{11} + {b_2}^2 + 4a_4)u_{12} + 2{u_{11}}^3 \\ & + ({b_2}^2 - 2a_4){u_{11}}^2 + (2a_3 - 2b_1b_2)u_{11} - b_0b_2 + {b_1}^2 - 2a_2)v_{11} \\ & + 2b_2{u_{12}}^2 + (b_2{u_{11}}^2 - 4b_1u_{11} - a_3b_2 + 2a_4b_1 - 2b_0)u_{12} - b_2{u_{11}}^4 \\ & + (a_4b_2 + b_1){u_{11}}^3 + (-a_3b_2 - a_4b_1 + 3b_0){u_{11}}^2 \\ & + (a_2b_2 + a_3b_1 - 2a_4b_0)u_{11} - a_2b_1 + a_3b_0. \end{aligned}$$

The expression for $W(2,0)$ is obtained from the following (cf. [8, p. 403]):

$$\frac{\sigma(2u)}{\sigma(u)^4} = \wp_{12}(u)\wp_{122}(u) - \wp_{22}(u)\wp_{112}(u) - \wp_{111}(u).$$

Then the other initial terms are computed by the formulas

$$\frac{W(m+1,i)W(m-1,i)}{W(m,i)^2} = \mathcal{F}_2([m]P + [i]Q, P) \tag{3}$$

for $i = 0, 1$. Here we use $W(1,0) = 1$. Note that we need the division by $W(m-1,i)$ in (3). Therefore we require the assumption that $W(m,0) \neq 0$ for $2 \leq m \leq 8$ and that $W(m,1) \neq 0$ for $-3 \leq m \leq 3$.

The function $\mathcal{F}_2(P,Q)$ is computed by

$$\mathcal{F}_2(P,Q) = -(v_{11}^2 - b_2u_{11}v_{11} + b_1v_{11} - u_{11}u_{12} + u_{11}^3 - a_4u_{11}^2 + a_3u_{11}) + (v_{21}^2 - b_2u_{21}v_{21} + b_1v_{21} - u_{21}u_{22} + u_{21}^3 - a_4u_{21}^2 + a_3u_{21}) - u_{12}u_{21} + u_{11}u_{22}.$$

This expression is obtained from [4, Theorem 3.3] or [16, Theorem 8.2].

Summarizing the above discussion, we obtain the following theorem.

Theorem 5. *Assume that the following values are all non-zero:*

$$W(2,0), W(3,0), \ldots, W(8,0), W(-4,1), W(-3,1), \ldots, W(3,1), \Delta(P). \quad (4)$$

Then we can compute the terms $W(m,0)$ and $W(m,1)$ in $O(\log m)$ operations in K.

6.3 The Tate-Lichtenbaum Pairing

Now we can compute the Tate-Lichtenbaum pairing via hyperelliptic nets. Let m be a positive integer with $m \mid (q-1)$ and assume that $[m]P = O$. By Corollary 1 and Theorem 5, we have the following corollary.

Corollary 2. *If the values in (4) are all non-zero, then we can compute the pairing $\tau_m(P,Q)$ with $O(\log m)$ operations in K.*

Note that the values in (4) are all non-zero for general points P and Q. In fact, by Theorem 2 and Proposition 7 (a), $W_{P,Q}(m,n)$ is a non-zero function for all $(m,n) \neq (0,0)$. Moreover, by [7, Lemma 3.2], we have

$$\Delta(P) = -W(2,0)^2W(3,0)^2 \begin{vmatrix} 1 & 1 & 1 \\ u_1(P) & u_1([2]P) & u_1([3]P) \\ u_2(P) & u_2([2]P) & u_2([3]P) \end{vmatrix},$$

where $(t^2 + u_1(R)t + u_2(R), v_1(R)t + v_2(R))$ is the Mumford representation of $R \in J$. We can verify that the determinant is not zero for a general point P with a computer algebra system. The author used Maxima [11].

6.4 An Example

The following example was computed with PARI/GP [20].

Let $q = 47$ and $m = 23$. We consider the curve $C\colon y^2 = x^5 + x + 41$ defined over $\mathbb{F}_{47}$. Let $\overline{D} = (x^2+6x+16, 31x+3)$ and $\overline{E} = (x^2+29x+24, 22x+14)$, where we use Mumford representations. Let P and Q be the points on J corresponding to $\overline{D}$ and $\overline{E}$ respectively.

By using the algorithm described in this section, we have $W(m+1,1) = 43$ and $W(m+1,0) = 8$. Therefore, by Corollary 1,

$$\tau_m(\overline{D}, \overline{E}) = \frac{43}{8} \bmod (\mathbb{F}_{47}^{\times})^{23} = 23 \bmod (\mathbb{F}_{47}^{\times})^{23}.$$

7 Conclusions

In this paper, we first defined hyperelliptic nets associated to hyperelliptic curves and points on their Jacobian varieties. Next, we described the Tate-Lichtenbaum pairing on a hyperelliptic curve in terms of hyperelliptic nets. Finally, we gave an algorithm to compute the Tate-Lichtenbaum pairing on a hyperelliptic curve of genus 2 by using hyperelliptic nets.

Our algorithm has the same order of complexity as Miller's algorithm. Since our algorithm is a generalization of Stange's algorithm, it has the same advantages as hers. For example, our algorithm requires few inversions, and the complexity is independent of the Hamming weight.

We have another advantage in the case of genus 2. To compute the Tate-Lichtenbaum pairing by using Miller's algorithm, we need to evaluate some rational functions at points defined over quadratic extensions of the field of definition. Operations in these extension fields are reduced by computing norms or resultants. However, our algorithm does not require such computations since our algorithm does not involve operations in any extension fields.

If we consider a hyperelliptic curve of genus greater than 2, the recurrences for hyperelliptic nets become too complicated. Thus algorithms based on the recurrences may not be practical. More efficient algorithms to compute terms of hyperelliptic nets are expected in future work.

Acknowledgments. The authors would like to thank the referees for useful comments. The first author was partially supported by Grant-in-Aid for JSPS Fellows (21-744). The second author was partially supported by Grant-in-Aid for Scientific Research (C) (20540125).

References

1. Arledge, J., Grant, D.: An explicit theorem of the square for hyperelliptic Jacobians. Michigan Math. J. 49, 485–492 (2001)
2. Bosch, S., Lütkebohmert, W., Raynaud, M.: Néron Models. Ergebnisse der Mathematik und ihrer Grenzgebiete, 3. Folge, Band 21. Springer, Berlin (1990)
3. Buchstaber, V.M., Enolskii, V.Z., Leykin, D.V.: Kleinian functions, hyperelliptic Jacobians and applications. Rev. Math. Math. Phys. 10, 1–125 (1997)
4. Buchstaber, V.M., Enolskii, V.Z., Leykin, D.V.: A recursive family of differential polynomials generated by the Sylvester identity and addition theorems for hyperelliptic Kleinian functions. Funct. Anal. Appl. 31, 240–251 (1997)
5. Fantechi, B., Göttsche, L., Illusie, L., Kleiman, S.L., Nitsure, N., Vistoli, A.: Fundamental Algebraic Geometry: Grothendieck's FGA explained. Mathematical Surveys and Monographs, vol. 123. American Mathematical Society, Providence (2005)
6. Frey, G., Rück, H.-G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comp. 62, 865–874 (1994)
7. Hone, A.N.W.: Analytic solutions and integrability for bilinear recurrences of order six. Appl. Anal. 89, 473–492 (2010)
8. Kanayama, N.: Division polynomials and multiplication formulae of Jacobian varieties of dimension 2. Math. Proc. Cambridge Philos. Soc. 139, 399–409 (2005)

9. Kanayama, N.: Corrections to "Division polynomials and multiplication formulae in dimension 2". Math. Proc. Cambridge Philos. Soc. 149, 189–192 (2010)
10. Lichtenbaum, S.: Duality theorems for curves over p-adic fields. Invent. Math. 7, 120–136 (1969)
11. Maxima.sourceforge.net: Maxima, a Computer Algebra System. Version 5.25.1 (2011), http://maxima.sourceforge.net/
12. Miller, V.S.: Short programs for functions on curves (1986) (unpublished manuscript), http://crypto.stanford.edu/miller/
13. Miller, V.S.: The Weil pairing, and its efficient calculation. J. Cryptology 17, 235–261 (2004)
14. Mumford, D.: Tata Lectures on Theta I. Progress in Mathematics, vol. 28. Birkhäuser, Boston (1983)
15. Mumford, D.: Tata Lectures on Theta II. Progress in Mathematics, vol. 43. Birkhäuser, Boston (1984)
16. Ônishi, Y.: Determinant expressions for hyperelliptic functions (with an appendix by Shigeki Matsutani). Proc. Edinb. Math. Soc. 48, 705–742 (2005)
17. Stange, K.E.: The Tate Pairing Via Elliptic Nets. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 329–348. Springer, Heidelberg (2007)
18. Stange, K.E.: Elliptic nets and elliptic curves. Algebra Number Theory 5, 197–229 (2011)
19. Uchida, Y.: Division polynomials and canonical local heights on hyperelliptic Jacobians. Manuscripta Math. 134, 273–308 (2011)
20. PARI/GP, version 2.3.4, Bordeaux (2008), http://pari.math.u-bordeaux.fr/

Appendix: Proof of Theorem 2

In this appendix, we prove Theorem 2.

First we assume that $\mathrm{char}(K) = 0$. Without loss of generality, we may assume that $K = \mathbb{Q}(a_0, \ldots, a_{2g}, b_0, \ldots, b_g)$. Then there exists an embedding $K \to \mathbb{C}$. We may regard K as a subfield of $\mathbb{C}$ through this embedding.

Let $x' = x$ and $y' = y + (b_g x^g + \cdots + b_0)/2$. Then we have another defining equation of C:

$$(y')^2 = (x')^{2g+1} + \lambda_{2g}(x')^{2g} + \cdots + \lambda_0,$$

where $\lambda_0, \ldots, \lambda_{2g} \in \mathbb{Q}[a_0, \ldots, a_{2g}, b_0, \ldots, b_g]$. As in Sect. 3, we can define rational functions $\Phi_{\boldsymbol{v}}$ on J^n by using this defining equation. By Proposition 11, $\Phi_{\boldsymbol{v}}$ is defined over K. It follows immediately that Propositions 6– 10 and Theorem 1 also hold for $\{\Phi_{\boldsymbol{v}}\}$.

Next we assume that $\mathrm{char}(K) > 0$. We define $\Phi_{\boldsymbol{v}}$ by reduction modulo a prime ideal. We use Jacobians defined over discrete valuation rings (cf. [2, Chap. 9] or [5, Chap. 9]).

Let R be a discrete valuation ring of mixed characteristic. Let L be the field of fractions of R, K be the residue field of R, and $\mathfrak{p}$ be the maximal ideal of R. Note that, for any field K of positive characteristic, there exists such a discrete valuation ring R.

Let $\mathcal{C}$ be a smooth projective scheme over R defined by

$$y^2 + (b_g x^g + \cdots + b_0)y = x^{2g+1} + a_{2g}x^{2g} + \cdots + a_0,$$

where $a_0, \ldots, a_{2g}, b_0, \ldots, b_g \in R$. Let $C_L = \mathcal{C} \times_{\operatorname{Spec} R} \operatorname{Spec} L$ and $C_K = \mathcal{C} \times_{\operatorname{Spec} R} \operatorname{Spec} K$. Then C_L and C_K are smooth projective hyperelliptic curves over L and K respectively.

We denote by $\operatorname{Pic}_{\mathcal{C}/R}$ the Picard scheme of $\mathcal{C}$ over R. Let $\mathcal{J} = \operatorname{Pic}^0_{\mathcal{C}/R}$ be the union of connected components of the identity element of the fibers of $\operatorname{Pic}_{\mathcal{C}/R}$, that is, $\mathcal{J}$ is the union of J_L and J_K, where J_L and J_K are the Jacobian varieties of C_L and C_K respectively. Since $\mathcal{C}$ is smooth and projective over R, $\mathcal{J}$ is a projective Abelian R-scheme. The scheme $\mathcal{J}$ is called the Jacobian of $\mathcal{C}$.

Let $\mathcal{J}^n = \mathcal{J} \times_{\operatorname{Spec} R} \cdots \times_{\operatorname{Spec} R} \mathcal{J}$, $J_L^n = J_L \times_{\operatorname{Spec} L} \cdots \times_{\operatorname{Spec} L} J_L$, and $J_K^n = J_K \times_{\operatorname{Spec} K} \cdots \times_{\operatorname{Spec} K} J_K$ be n-fold fiber products. Then we have the following commutative diagram:

$$\begin{array}{ccccc} J_L^n & \xrightarrow{i} & \mathcal{J}^n & \xleftarrow{j} & J_K^n \\ \downarrow & & \downarrow & & \downarrow \\ \operatorname{Spec} L & \longrightarrow & \operatorname{Spec} R & \longleftarrow & \operatorname{Spec} K \end{array} .$$

We will define a rational function $\tilde{\Phi}_{\boldsymbol{v}}$ on J_K^n by $\tilde{\Phi}_{\boldsymbol{v}} = j^* i_* \Phi_{\boldsymbol{v}}$, where $\Phi_{\boldsymbol{v}}$ in the right-hand side has been already defined since L is a filed of characteristic zero. The morphism $i_* \colon \mathcal{K}(J_L^n) \to \mathcal{K}(\mathcal{J}^n)$ is an isomorphism of function fields. To verify that $j^* i_* \Phi_{\boldsymbol{v}}$ defines a non-zero rational function for any non-zero $\boldsymbol{v}$, it is sufficient to prove the following theorem.

Theorem 6. *Let v be the valuation of $\mathcal{K}(\mathcal{J}^n)$ induced by the special fiber of $\mathcal{J}^n$. Then, for any $\boldsymbol{v} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$, we have $v(i_* \Phi_{\boldsymbol{v}}) = 0$.*

Proof. The proof is similar to that of the case of elliptic curves [18, Theorem 4.4].

By Proposition 9,

$$\frac{\Phi_{\boldsymbol{v}+\boldsymbol{w}}(\boldsymbol{P}) \Phi_{\boldsymbol{v}-\boldsymbol{w}}(\boldsymbol{P})}{\Phi_{\boldsymbol{v}}(\boldsymbol{P})^2 \Phi_{\boldsymbol{w}}(\boldsymbol{P})^2} = \mathcal{F}_g([v_1]P_1 + \cdots + [v_n]P_n, [w_1]P_1 + \cdots + [w_n]P_n) \tag{5}$$

for $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{Z}^n$ with $\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{v}+\boldsymbol{w}, \boldsymbol{v}-\boldsymbol{w} \neq \mathbf{0}$. We define a function $\mathcal{F}_g(\boldsymbol{v}, \boldsymbol{w})$ on J_L^n by

$$\mathcal{F}_g(\boldsymbol{v}, \boldsymbol{w})(\boldsymbol{P}) = \mathcal{F}_g([v_1]P_1 + \cdots + [v_n]P_n, [w_1]P_1 + \cdots + [w_n]P_n).$$

Through the isomorphism i_*, we may regard that $\mathcal{F}_g(\boldsymbol{v}, \boldsymbol{w}) \in \mathcal{K}(\mathcal{J}^n)$. We prove that $v(\mathcal{F}_g(\boldsymbol{v}, \boldsymbol{w})) = 0$.

We assume that $v(\mathcal{F}_g(\boldsymbol{v}, \boldsymbol{w})) < 0$. We define a morphism $\varphi \colon J_K^n \to J_K \times J_K$ by $\varphi(\boldsymbol{P}) = ([v_1]P_1 + \cdots + [v_n]P_n, [w_1]P_1 + \cdots + [w_n]P_n)$. By the assumption and the property of $\mathcal{F}_g$, the image of φ is contained in $\Theta \times J_K \cup J_K \times \Theta$. Since J_K^n is irreducible, at least one of the following holds:

(a) $[v_1]P_1 + \cdots + [v_n]P_n \in \Theta$ for all $\boldsymbol{P} \in J_K^n$, or
(b) $[w_1]P_1 + \cdots + [w_n]P_n \in \Theta$ for all $\boldsymbol{P} \in J_K^n$.

We assume (a). Since the multiplication map $[v_i]$ is surjective whenever $v_i \neq 0$, we have $v_i = 0$ for all i. This contradicts the assumption $\boldsymbol{v} \neq \mathbf{0}$. Similarly, (b) contradicts the assumption $\boldsymbol{w} \neq \mathbf{0}$. Therefore we obtain $v(\mathcal{F}_g(\boldsymbol{v}, \boldsymbol{w})) \geq 0$.

Next we assume that $v(\mathcal{F}_g(\boldsymbol{v},\boldsymbol{w})) > 0$. By the same argument as above, we have $\boldsymbol{v}+\boldsymbol{w}=\boldsymbol{0}$ or $\boldsymbol{v}-\boldsymbol{w}=\boldsymbol{0}$, which contradicts the assumption. Therefore we obtain $v(\mathcal{F}_g(\boldsymbol{v},\boldsymbol{w}))=0$.

We define a function $M\colon \mathbb{Z}^n \to \mathbb{Z}$ by $M(\boldsymbol{v}) = v(\Phi_{\boldsymbol{v}})$ for $\boldsymbol{v} \neq \boldsymbol{0}$ and $M(\boldsymbol{0}) = 0$. We prove that M is a quadratic form.

If $\boldsymbol{v},\boldsymbol{w},\boldsymbol{v}+\boldsymbol{w},\boldsymbol{v}-\boldsymbol{w} \neq \boldsymbol{0}$, by (5) and $v(\mathcal{F}_g(\boldsymbol{v},\boldsymbol{w}))=0$, we have

$$M(\boldsymbol{v}+\boldsymbol{w}) + M(\boldsymbol{v}-\boldsymbol{w}) = 2M(\boldsymbol{v}) + 2M(\boldsymbol{w}). \tag{6}$$

By Proposition 6, we have $\Phi_{-\boldsymbol{v}} = \pm\Phi_{\boldsymbol{v}}$. Hence $M(-\boldsymbol{v}) = M(\boldsymbol{v})$. Therefore (6) holds if $\boldsymbol{v}=\boldsymbol{0}$ or $\boldsymbol{w}=\boldsymbol{0}$.

When $\boldsymbol{v}+\boldsymbol{w}=\boldsymbol{0}$ or $\boldsymbol{v}-\boldsymbol{w}=\boldsymbol{0}$, it is sufficient to prove that $M(2\boldsymbol{u}) = 4M(\boldsymbol{u})$ for all $\boldsymbol{u} \neq \boldsymbol{0}$. This is obtained by taking the sums of the four instances of (6) with $(\boldsymbol{v},\boldsymbol{w}) = (4\boldsymbol{u},\boldsymbol{u}),(3\boldsymbol{u},\boldsymbol{u}),(3\boldsymbol{u},\boldsymbol{u}),(2\boldsymbol{u},\boldsymbol{u})$ and subtracting the instance of (6) with $(\boldsymbol{v},\boldsymbol{w}) = (3\boldsymbol{u},2\boldsymbol{u})$.

Now we have proved that M is a quadratic form, we have $M(\boldsymbol{v}) = 0$ for all $\boldsymbol{v} \in \mathbb{Z}^n$ by Proposition 7 and [18, Lemma 4.5]. Therefore $v(\Phi_{\boldsymbol{v}}) = 0$ for all $\boldsymbol{v} \neq \boldsymbol{0}$. □

Now we have proved Theorem 6, we can define a function $\tilde{\Phi}_{\boldsymbol{v}}$ by $\tilde{\Phi}_{\boldsymbol{v}} = j^* i_* \Phi_{\boldsymbol{v}}$. By construction, the family $\{\tilde{\Phi}_{\boldsymbol{v}}\}_{\boldsymbol{v}\in\mathbb{Z}^n}$ of rational functions on J_K^n satisfies Propositions 6–10 and Theorem 1. This concludes the proof of Theorem 2.

Genus 2 Hyperelliptic Curve Families with Explicit Jacobian Order Evaluation and Pairing-Friendly Constructions*

Aurore Guillevic[1,2] and Damien Vergnaud[1]

[1] Équipe crypto DI, École Normale Supérieure, C.N.R.S., I.N.R.I.A.
45, rue d'Ulm, 75230 Paris Cedex 05, France
aurore.guillevic@ens.fr
[2] Laboratoire Chiffre, Thales Communications & Security S.A.,
160 bd de Valmy BP 82, 92704 Colombes Cedex France

Abstract. The use of elliptic and hyperelliptic curves in cryptography relies on the ability to compute the Jacobian order of a given curve. Recently, Satoh proposed a probabilistic polynomial time algorithm to test whether the Jacobian – over a finite field $\mathbb{F}_q$ – of a hyperelliptic curve of the form $Y^2 = X^5 + aX^3 + bX$ (with $a, b \in \mathbb{F}_q^*$) has a large prime factor. His approach is to obtain candidates for the zeta function of the Jacobian over $\mathbb{F}_q^*$ from its zeta function over an extension field where the Jacobian splits. We extend and generalize Satoh's idea to provide *explicit* formulas for the zeta function of the Jacobian of genus 2 hyperelliptic curves of the form $Y^2 = X^5 + aX^3 + bX$ and $Y^2 = X^6 + aX^3 + b$ (with $a, b \in \mathbb{F}_q^*$). Our results are proved by elementary (but intricate) polynomial root-finding techniques. Hyperelliptic curves with small embedding degree and large prime-order subgroup are key ingredients for implementing pairing-based cryptographic systems. Using our closed formulas for the Jacobian order, we propose two algorithms which complement those of Freeman and Satoh to produce genus 2 pairing-friendly hyperelliptic curves. Our method relies on techniques initially proposed to produce pairing-friendly elliptic curves (namely, the Cocks-Pinch method and the Brezing-Weng method). We show that the previous security considerations about embedding degree are valid for an elliptic curve and can be lightened for a Jacobian. We demonstrate this method by constructing several interesting curves with ρ-values around 4 with a Cocks-Pinch-like method and around 3 with a Brezing-Weng-like method.

Keywords: Hyperelliptic Curves, Genus 2, Order Computation, Ordinary Curves, Pairing-Friendly Constructions, Cocks-Pinch Method, Brezing-Weng Method.

1 Introduction

In 1985, the idea of using the group of rational points on an elliptic curve over a finite field in public-key cryptography was introduced independently by Miller [33]

* Extended abstract. The full version is available on ePrint, report 2011/604.

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 234–253, 2013.

and Koblitz [27]. The main advantage of using elliptic curves is efficiency since no sub-exponential algorithms are known for solving the discrete logarithm problem in these groups (and thus key sizes can remain small). In 1989, Koblitz [28] suggested using Jacobians of hyperelliptic curves in cryptography. Genus 1 hyperelliptic curves are elliptic curves; genus 2 and 3 hyperelliptic curves are more complicated but are an attractive replacement for elliptic curves in cryptography. They are as efficient as genus one curves for bandwidth but still have a slower group law.

As for any group used for the discrete logarithm problem, one needs the order of the group to contain a large prime factor. This raised the problem of finding hyperelliptic curves over a finite field whose Jacobian order is (almost) a prime. For elliptic curves over finite fields, the Schoof-Elkies-Atkin (SEA) algorithm [36,32] runs in polynomial time in any characteristic and in small characteristic, there are even faster algorithms based on the so-called p-adic method [34,32]. For genus 2 hyperelliptic curves, the p-adic method gives efficient point counting algorithms in small characteristic, but up to now, no algorithms as efficient as SEA are known when the characteristic of the underlying finite field is large (though substantial progress has recently been made in [21] and [23]). Using basic properties on character sums, Furukawa, Kawazoe and Takahashi [15] gave an explicit closed formula for the order of Jacobians of very special curves of type $Y^2 = X^5 + bX$ where $b \in \mathbb{F}_q$. Satoh [35] considered an intermediate approach and showed that point counting on specific Jacobians of certain genus 2 curves can be performed much faster than point counting on Jacobians of generic curves. He gave an algorithm to test whether the order of the Jacobian of a given hyperelliptic curve in the form $Y^2 = X^5 + aX^3 + bX$ has a large prime factor. His method relies on the fact that the Jacobian of the curve is $\mathbb{F}_{q^4}$-isogenous to a square of an elliptic curve defined over $\mathbb{F}_{q^4}$, hence their respective zeta functions are the same over $\mathbb{F}_{q^4}$ and can be computed by the SEA algorithm. Satoh's method obtains candidates for the zeta function of the Jacobian over $\mathbb{F}_q$ from the zeta function over $\mathbb{F}_{q^4}$. The methodology can be formalized as an efficient probabilistic polynomial algorithm but is not explicit and gives 26 possible orders to test for the Jacobian.

In recent years, many useful cryptographic protocols have been proposed that make use of a bilinear map, or *pairing*, between two groups in which the discrete logarithm problem is hard (*e.g.* [4,5]). Pairing-based cryptosystems can be constructed by using the Weil or Tate pairing on abelian varieties over finite fields. These pairings take as input points on an abelian variety defined over the field $\mathbb{F}_q$ and produce as output elements of an extension field $\mathbb{F}_{q^k}$. The degree of this extension is known as the *embedding degree*. In cryptography, abelian varieties obtained as Jacobians of hyperelliptic curves are often used. Suitable hyperelliptic curves for pairing-based cryptography are called *pairing-friendly*. Such pairing-friendly curves are rare and thus require specific constructions.

For a pairing-based cryptosystem to be secure and practical, the group of rational points on the Jacobian should have a subgroup of large prime order r, and the embedding degree k should be large enough so that the discrete loga-

rithm problem in $\mathbb{F}_{q^k}$ is difficult but small enough to make the pairing efficiently computable. The efficiency parameter in pairing-friendly constructions is the so-called *ρ-value*: for a Jacobian of hyperelliptic curve of genus g it is defined as $\rho = g \log q / \log r$. It measures the ratio of the bit-sizes of the order of the Jacobian and the subgroup order r. The problem of constructing pairing-friendly elliptic curves with small ρ-values has been studied extensively [12]. Unfortunately, there are very few results for constructing pairing-friendly hyperelliptic curves of genus $g \geq 2$ with small ρ-values [17,2]. Galbraith, Pujolas, Ritzenthaler and Smith [18] gave (supersingular) genus 2 pairing-friendly hyperelliptic curves with ρ-values close to 1 but only for embedding degrees $k \in \{4, 5, 6, 12\}$. Freeman, Stevenhagen and Streng presented in [13] a general method that produced pairing-friendly (ordinary) genus 2 pairing-friendly hyperelliptic curves with $\rho \simeq 8$ for all embedding degrees k. Kawazoe and Takahashi [26] (see also [25]) presented an algorithm which constructed hyperelliptic curves of the form $Y^2 = X^5 + bX$ (thanks to the closed formula for its Jacobian order). Following Satoh's approach, Freeman and Satoh [14] constructed pairing-friendly genus 2 hyperelliptic curves of the form $Y^2 = X^5 + aX^3 + bX$ and $Y^2 = X^6 + aX^3 + b$ (with $a, b \in \mathbb{F}_q^*$) by means of elliptic curves that become pairing-friendly over a finite extension of the underlying finite field. Constructions from [26,25,14] produce pairing-friendly Jacobians with $2.22 \leqslant \rho \leqslant 4$ only for embedding degrees divisible by 3 or 4.

Our Contributions. Satoh's approach to compute the Jacobian order of a hyperelliptic curve $Y^2 = X^5 + aX^3 + bX$ is not explicit. For each candidate, he has to check that the order is not weak for cryptographic use. In [22, § 4], Gaudry and Schost showed that the Jacobians of hyperelliptic curves of the form $Y^2 = X^6 + aX^3 + b$ are also isogenous to a product of two elliptic curves over an extension field. Satoh claimed that his method applies as well to this family but did not derive an algorithm for it.

Our first contribution is to extend and generalize Satoh's idea to provide *explicit* formulas for the zeta function of the Jacobian of genus 2 hyperelliptic curves of the form $Y^2 = X^5 + aX^3 + bX$ and $Y^2 = X^6 + aX^3 + b$ (with $a, b \in \mathbb{F}_q^*$). Our results are proved by elementary polynomial root-finding techniques. This permits to generate efficiently a random hyperelliptic curve, in one of these two forms, suitable for cryptographic use. These curves enable various improvements to make scalar multiplication in the Jacobian efficient (*e.g.* the Gallant-Lambert-Vanstone algorithm [19], Takashima's algorithm [38] or Gaudry's algorithm [20]). These large families of curves are still very specific but there is no evidence that they should be more vulnerable to discrete logarithm attacks than the absolutely simple Jacobians.

Two algorithms proposed in [14] to produce pairing-friendly genus 2 hyperelliptic curves are very general as they are still valid for arbitrary abelian varieties over any finite field. Assuming that the finite field is a prime field and the abelian variety is of the above form, we can consider any embedding degree. The security restrictions concerning the embedding degree (which must be a multiple of 3 or 4) made in [14] are unnecessary in this particular case. Satoh and Freeman exclude constructions which need an elliptic curve defined over a

quadratic extension of a prime field (with j-invariant in $\mathbb{F}_{p^2}$), resulting in restricted sets of parameters $a, b \in \mathbb{F}_p$. Using our closed formulas for the Jacobian order, we use two approaches that construct pairing-friendly elliptic curves and adapt them to produce pairing-friendly genus 2 curves. The first one is based on the Cocks-Pinch method [9] (see also [16, Algorithm IX.4]) of constructing individual ordinary pairing-friendly elliptic curves. The other is based on cyclotomic polynomials as originally proposed by Brezing and Weng [7] which generates families of curves while achieving better ρ-values. We adapt both constructions using the elliptic curve complex multiplication method (CM) [1,16] to compute one of the two elliptic curves to which the Jacobian is isogenous to (even if the curve j-invariant is in $\mathbb{F}_{p^2}$ rather than in a prime field $\mathbb{F}_p$). In particular, this method can construct pairing-friendly elliptic curves over $\mathbb{F}_{p^2}$ but unfortunately with $\rho \simeq 4$.

Our approach contains the previous constructions by Kawazoe and Takahashi [26] and is in a sense a specialization of Freeman and Satoh [14]. It also produces new families for ordinary genus 2 hyperelliptic curves. Explicit examples of cryptographically interesting curves are given.

2 Explicit Computation of $J_{\mathcal{C}_5}$ Order

Throughout this paper, $p \geq 5$ denotes a prime number and q a power of p. In this section, we consider the genus 2 hyperelliptic curve defined over a finite field $\mathbb{F}_q$:

$$\mathcal{C}_5(\mathbb{F}_q) : Y^2 = X^5 + aX^3 + bX, \text{ with } a, b \neq 0 \in \mathbb{F}_q \ .$$

The Jacobian of the curve is denoted $J_{\mathcal{C}_5}$ and it splits into two isogenous elliptic curves in an extension over $\mathbb{F}_q$ of degree 1, 2 or 4 [35]. These two elliptic curves admit a quadratic twist which is half the time defined on a smaller extension. As the trace computation is then more efficient, we will also consider directly these quadratic twists, as in [14].

2.1 Splitting the Jacobian $J_{\mathcal{C}_5}$ into Two Isogenous Elliptic Curves

Satoh showed in [35] that the Jacobian splits into two elliptic curves defined by

$$\begin{aligned} E_1(\mathbb{F}_q[\sqrt[4]{b}]) &: Y^2 = \quad \delta(X-1)(X^2 - \gamma X + 1) \text{ and} \\ E_2(\mathbb{F}_q[\sqrt[4]{b}]) &: Y^2 = -\delta(X-1)(X^2 - \gamma X + 1) \end{aligned}$$

with $\gamma = (2a - 12\sqrt{b})/(a + 2\sqrt{b})$ and $\delta = (a + 2\sqrt{b})/(64\sqrt[4]{b}^3)$. The isogeny between $J_{\mathcal{C}_5}$ and $E_1 \times E_2$ is defined over $\mathbb{F}_q[\sqrt[4]{b}]$. Using the notation $c = a/\sqrt{b}$ from [14], the curve parameters are

$$\gamma = \frac{2c - 12}{c + 2}, \ \delta = \frac{c + 2}{2^6 \sqrt[4]{b}} \text{ and } j(E_1) = j(E_2) = 2^6 \frac{(3c - 10)^3}{(c + 2)^2 (c - 2)}.$$

Since E_1 and E_2 are isogenous over $\mathbb{F}_q[\sqrt[4]{b}, \sqrt{-1}]$, they have the same order over this field. They also admit a 2-torsion point $P = (1, 0)$ over $\mathbb{F}_q[\sqrt[4]{b}]$ (their order is

therefore even). Let E_1' and E_2' denote the quadratic twists of E_1 and E_2 obtained by removing the term $1/(2^6\sqrt[4]{b})$ in δ. They are isogenous over $\mathbb{F}_q[\sqrt{b}, \sqrt{-1}]$.

$$\begin{array}{ccc}
 & (E_1 \times E_2)(\mathbb{F}_q[\sqrt[8]{b}]) & \overset{\text{isomorphism}}{\longleftrightarrow} & (E_1' \times E_2')(\mathbb{F}_q[\sqrt[8]{b}]) \\
 & \vert & & \vert \\
J_{\mathcal{C}_5}(\mathbb{F}_q[\sqrt[4]{b}]) & \overset{\text{isogeny}}{\longleftrightarrow} & (E_1 \times E_2)(\mathbb{F}_q[\sqrt[4]{b}]) & (E_1' \times E_2')(\mathbb{F}_q[\sqrt[4]{b}]) \\
\vert & & & \vert \\
J_{\mathcal{C}_5}(\mathbb{F}_q[\sqrt{b}]) & & & (E_1' \times E_2')(\mathbb{F}_q[\sqrt{b}]) \\
\vert & & & \\
J_{\mathcal{C}_5}(\mathbb{F}_q) & & &
\end{array}$$

The Jacobian $J_{\mathcal{C}_5}$ has *the same order* as the product $E_1 \times E_2$ over the extension field where the isogeny is defined. Computing the elliptic curve order is easy with the SEA algorithm [36,32] which computes the trace. As the computation is faster for the quadratic twist (which is defined over $\mathbb{F}_q[\sqrt{b}]$ instead of $\mathbb{F}_q[\sqrt[4]{b}]$, we will also consider the isogeny between $J_{\mathcal{C}_5}$ and $E_1' \times E_2'$.

$$\begin{aligned}
E_1'(\mathbb{F}_q[\sqrt{b}]) : Y^2 &= \quad (c+2)(X-1)(X^2 - \gamma X + 1) \text{ and} \\
E_2'(\mathbb{F}_q[\sqrt{b}]) : Y^2 &= -(c+2)(X-1)(X^2 - \gamma X + 1) \, .
\end{aligned}$$

It remains to compute the Jacobian order from $\#J_{\mathcal{C}_5}(\mathbb{F}_q[\sqrt[4]{b}])$ to $\#J_{\mathcal{C}_5}(\mathbb{F}_q)$. We develop explicit formulas using the zeta function properties. Going down directly from $\#J_{\mathcal{C}_5}(\mathbb{F}_{q^4})$ to $\#J_{\mathcal{C}_5}(\mathbb{F}_q)$ does not provide an explicit order. We compute step by step the explicit order, descending by quadratic extensions.

2.2 Computing Explicit Order Using Zeta Function

Let $Z_{J_{\mathcal{C}_5}}$ denote the zeta function of the Jacobian $J_{\mathcal{C}_5}$ which satisfies the following properties [35]:

1. $Z_{J_{\mathcal{C}_5}}(T, \mathbb{F}_q) \in \mathbb{Z}[T]$ i.e. the zeta function is a polynomial with integer coefficients;
2. the degree of the zeta function polynomial is $\deg Z_{J_{\mathcal{C}_5}}(T, \mathbb{F}_q) = 2g = 4$;
3. the Jacobian order is related to the zeta function by $\#J_{\mathcal{C}_5}(\mathbb{F}_q) = Z_{J_{\mathcal{C}_5}}(1, \mathbb{F}_q)$;
4. let $z_{1,q}, z_{2,q}, z_{3,q}, z_{4,q}$ be the four roots of $Z_{J_{\mathcal{C}_5}}(T, \mathbb{F}_q)$ in $\mathbb{C}$. Up to index permutation, we have $z_{1,q} z_{2,q} = q$ and $z_{3,q} z_{4,q} = q$;
5. the roots of $Z_{J_{\mathcal{C}_5}}(T, \mathbb{F}_{q^n})$ the zeta function of the Jacobian considered over a degree n extension $\mathbb{F}_{q^n}$ are those over $\mathbb{F}_q$ to the power n: $Z_{J_{\mathcal{C}_5}}(T, \mathbb{F}_{q^n}) = (T - z_{1,q}^n)(T - z_{2,q}^n)(T - z_{3,q}^n)(T - z_{4,q}^n)$.

Satoh's method to compute the Jacobian order is derived from the fact that if $J_{\mathcal{C}_5}$ is isogenous over $\mathbb{F}_q$ to $E_1 \times E_2$, then $Z_{J_{\mathcal{C}_5}}(T, \mathbb{F}_q) = Z_{E_1}(T, \mathbb{F}_q) \times Z_{E_2}(T, \mathbb{F}_q)$. We have $Z_{E_1}(T, \mathbb{F}_q) = T^2 - t_q T + q$ with t_q the trace of the Frobenius endomorphism and $\#E_1(\mathbb{F}_q) = q + 1 - t_q = Z_{E_1}(1, \mathbb{F}_q)$.

Let us denote $Z_{J_{\mathcal{C}_5}}(T, \mathbb{F}_q) = T^4 - a_q T^3 + b_q T^2 - q a_q T + q^2$ with

$$\begin{aligned}
a_q &= z_{1,q} + z_{2,q} + z_{3,q} + z_{4,q} \\
b_q &= z_{1,q} z_{2,q} + z_{1,q} z_{3,q} + z_{1,q} z_{4,q} + z_{2,q} z_{3,q} + z_{2,q} z_{4,q} + z_{3,q} z_{4,q} \\
&= 2q + (z_{1,q} + z_{2,q})(z_{3,q} + z_{4,q}) \, .
\end{aligned}$$

Our goal is to find two simple formulas for computing (a_q, b_q) in terms of (a_{q^2}, b_{q^2}) and apply the two formulas recursively. A careful computation gives

$$a_{q^2} = (a_q)^2 - 2b_q \tag{1}$$
$$b_{q^2} = (b_q)^2 - 4qb_q + 2q^2 - 2qa_{q^2} \tag{2}$$

Knowing a_{q^2} and b_{q^2}, we can solve[1] equation (2) for b_q then recover a_q using (1).

We have to determine where the isogeny is defined in order to solve the corresponding system. In each case, two solutions are possible for b_q. One of them induces a square root in a_q that must be an integer because the two coefficients a_q and b_q are integers. This solution can be chosen if the isogeny is actually defined over $\mathbb{F}_{q^2}$ *and* $\mathbb{F}_q$ or if the elliptic curve has an additional property. In these two cases the Jacobian splits over $\mathbb{F}_{q^2}$ *and* over $\mathbb{F}_q$.

When the isogeny between $J_{\mathcal{C}_5}$ and $E_1 \times E_2$ is defined over $\mathbb{F}_{q^4}$ but not over a subfield of $\mathbb{F}_{q^4}$ and the trace t_{q^4} of the two elliptic curves is such that $2q^2 + t_{q^4}$ is not a square, we see an other simplification for the zeta function coefficients: they are not squares but of the form two times a square. After easy (but cumbersome) calculation and a difficult identification of the special cases that do not correspond to the general solution, we obtain the following theorem:

Theorem 1. *Let $\mathcal{C}_5$ be a hyperelliptic curve defined over a finite field $\mathbb{F}_q$ by the equation $\mathcal{C}_5(\mathbb{F}_q) : Y^2 = X^5 + aX^3 + bX$ with $a, b \neq 0 \in \mathbb{F}_q$. Let E_1 and E_2 be the elliptic curves defined over $\mathbb{F}_q[\sqrt[4]{b}]$ and E_1', E_2' their quadratic twists defined over $\mathbb{F}_q[\sqrt{b}]$, isogenous over $\mathbb{F}_q[\sqrt{b}, \sqrt{-1}]$. Let t_q be the trace of $E_1(\mathbb{F}_q)$ if b is a fourth power, let t_q' be the trace of $E_1'(\mathbb{F}_q)$ if b is a square, let t_{q^2} be the trace of $E_1(\mathbb{F}_{q^2})$ if b is not a square in $\mathbb{F}_q$ and let t_{q^2}' be the trace of $E_1'(\mathbb{F}_{q^2})$ and t_{q^4} of $E_1(\mathbb{F}_{q^4})$ if b is neither a square nor a fourth power.*

1. *If b is a fourth power then $\#J_{\mathcal{C}_5}(\mathbb{F}_q) = (q + 1 - t_q)^2$ if $\sqrt{-1} \in \mathbb{F}_q$ and $\#J_{\mathcal{C}_5}(\mathbb{F}_q) = (q + 1 - t_q)(q + 1 + t_q)$ if $\sqrt{-1} \notin \mathbb{F}_q$.*
2. *If b is a square but not a fourth power ($q \equiv 1 \mod 4$) and $t_{q^2} + 2q$ is not a square, then $\#J_{\mathcal{C}_5}(\mathbb{F}_q) = (q - 1)^2 + (t_q')^2$.*
3. *If b is not a square and $\sqrt[4]{b} \in \mathbb{F}_{q^2}$ ($q \equiv 3 \mod 4$) and $t_{q^2} + 2q$ is not a square, then $\#J_{\mathcal{C}_5}(\mathbb{F}_q) = q^2 + 1 - t_{q^2}$.*
4. *If b is not a square and $\sqrt[4]{b} \notin \mathbb{F}_{q^2}$ ($q \equiv 1 \mod 4$) and $t_{q^4} + 2q^2$ is not a square, then $\#J_{\mathcal{C}_5}(\mathbb{F}_q)$ is equal to $q^2 + 1 - 2n(q+1) + 2n^2$ or $q^2 + 1 + 2n(q+1) + 2n^2$ where $n \in \mathbb{N}$ is such that $2q + t_{q^2}' = 2n^2$ if $q \equiv 5 \mod 8$ or $2q - t_{q^2}' = 2n^2$ if $q \equiv 1 \mod 8$.*

The case 1 of Th.1 is of no interest in cryptography as the Jacobian order factors trivially. In the cases 2 and 3, we may as well work directly with the elliptic curve $E_1(\mathbb{F}_{q^2})$ (of even order) since the arithmetic is not (yet) as efficient in genus two as in genus one. The case 4 provides ordinary Jacobians of hyperelliptic curves with explicit order and of cryptographic interest. The very special cases excluded

[1] Satoh [35] used only Equation (1) which resulted in a more intricate polynomial system with degree 16 polynomial equations to solve.

in the theorem with $2q^2 + t_{q^4}$ or $2q + t_{q^2}$ squares give a Jacobian either which splits over $\mathbb{F}_q$ or whose order is an elliptic curve order, as in case 3. They are detailed in the following remark.

Remark 1. With the same notations as in the previous theorem,

1. If b is a square but not a fourth power ($q \equiv 1 \mod 4$) and if $t_{q^2} + 2q = y^2$ (with $y \in \mathbb{N}^*$), the Jacobian splits and its order is one of $(q+1-y)^2$, $(q+1+y)^2$ or $(q+1-y)(q+1+y) = (q-1)^2 + (t_q^{'})^2$.
2. If b is not a square and $\sqrt[4]{b} \in \mathbb{F}_{q^2}$ ($q \equiv 3 \mod 4$) and if $t_{q^2} + 2q = y^2$ (with $y \in \mathbb{N}^*$), the Jacobian splits and its order is one of $(q+1-y)^2$, $(q+1+y)^2$ or $(q+1-y)(q+1+y) = q^2 + 1 - t_{q^2}$.
3. If b is not a square and $\sqrt[4]{b} \notin \mathbb{F}_{q^2}$ ($q \equiv 1 \mod 4$) and if $t_{q^4} + 2q^2 = y^2$ (with $y \in \mathbb{N}^*$) then we decompose $-\Delta(E_1^{'}(\mathbb{F}_{q^2})) = -(t_{q^2}^{'})^2 + 4q^2 = t_{q^4} + 2q^2 = y^2$ in the two factors $(2q + t_{q^2}^{'})(2q - t_{q^2}^{'})$. Let $2q + t_{q^2}^{'} = D_1 y_1^2$ and $2q - t_{q^2}^{'} = D_2 y_2^2$ with D_1, D_2 square-free integers.
   (a) if $D_1 \neq 2$ and $D_2 \neq 2$ then $\pm y + 2q^2$ is not a square and $\#J_{\mathcal{C}_5}(\mathbb{F}_q)$ is equal to $q^2 + 1 - y$ or $q^2 + 1 + y$.
   (b) if $D_1 = D_2 = 2$ then $\#J_{\mathcal{C}_5}(\mathbb{F}_q) = q^2 + 1 - 2n(q+1) + 2n^2$ (case 4 of Th. 1) can happen with $n \in \{y_1, -y_1, y_2, -y_2\}$. In the same time, $y + 2q^2 = (y_1 + y_2)^2$ and $-y + 2q^2 = (y_1 - y_2)^2$ are squares hence $\#J_{\mathcal{C}_5}(\mathbb{F}_q)$ can be $(q+1-s)^2$ with $s \in \{y_1 + y_2, -y_1 - y_2, y_1 - y_2, -y_1 + y_2\}$. The two last possibilities are $q^2 + 1 - y$ and $q^2 + 1 + y$.

The cases 1, 2 and 3 in the remark 1 are special (*e.g.* the cases 1 and 3 appear only when the elliptic curves E_1 and E_2 have complex multiplication by $i = \sqrt{-1}$). Moreover, in 1, 2 and 3b of Rem. 1 the Jacobian order splits. In 3a, the Jacobian order is equal to the order of a quartic twist of $E_1(\mathbb{F}_{q^2})$.

In practice, when the Th. 1 or Rem. 1 present several order possibilities one can easily discriminate between them by checking whether the scalar multiplication of a random point by the possible orders gives the infinity point.

In the two following examples, we took at random a prime $p \equiv 1 \mod 4$ of 128 bits and started with $a = -3$ and $b = -2$ until b was not a square mod p. Then let $c = a/\sqrt{b}$, $E_1^{'}(\mathbb{F}_{p^2}) : y^2 = (x-1)((c+2)x^2 - (2c-12)x + (c+2))$ and $t_{p^2}^{'}$ its trace. We deduced the Jacobian order and factorized it. We repeated this process with subsequent b-values until the Jacobian order was almost prime.

Example 1. $p = \texttt{0x84c4f7a6b9aee8c6b46b34fa2a2bae69} = 1 \mod 8$. The 17th test provided $b = -38$, $t_{p^2}^{'} = \texttt{0x702461acf6a929e295786868f846ab40} = 0 \mod 2$, $b_p = 2p - t_{p^2}^{'} = 2n^2$ as expected with $n = -\texttt{0x8c1fc81b9542ce23}$. We found $\#J_{\mathcal{C}_5}(\mathbb{F}_p) = 2^5 r$ with r a 250-bit prime of cryptographic size close to the 128-bit security level. $r = \texttt{0x226ddb780b2ded62d1d70138d9c7361794679a609fbe5ae85918c88f5b6ea7d}$.

Example 2. $p = \texttt{0xb081d45d7d08109c2905dd6187f7cbbd} = 5 \mod 8$. The 17th test provided $b = -41$, $t_{p^2}^{'} = \texttt{-0x11753eaa61f725ff118f63bb131c8b8f2} = 0 \mod 2$, $b_p = 2p + t_{p^2}^{'} = 2n^2$ as expected with $n = -\texttt{0x611e298cc019b06e}$. We found

$\#J_{\mathcal{C}_5}(\mathbb{F}_p) = 2 \cdot 5 \cdot r$ with r a 252-bit prime of cryptographic size close to the 128-bit security level:
$r =$ `0xc2b7a2f39d49b6b579d4c15a8440315cd1ccc424df912e6748c949008ebd989`.

3 Explicit Computation of $J_{\mathcal{C}_6}$ Order

In this section, we consider the genus 2 hyperelliptic curves defined over a finite field $\mathbb{F}_q$:

$$\mathcal{C}_6(\mathbb{F}_q) : Y^2 = X^6 + aX^3 + b \text{ with } a, b \neq 0 \in \mathbb{F}_q \ .$$

The Jacobian of the curve is denoted $J_{\mathcal{C}_6}$ and it splits into two isogenous elliptic curves in an extension over $\mathbb{F}_q$ of degree 1, 2, 3 or 6 [22,14]. The computation of the zeta function of $J_{\mathcal{C}_6}$ over $\mathbb{F}_q$ is similar to those of $J_{\mathcal{C}_5}$ from the previous section but with more technical details.

3.1 Decomposition into Two Isogenous Elliptic Curves

Freeman and Satoh showed in [14] that $J_{\mathcal{C}_6}$ is isogenous over $\mathbb{F}_q[\sqrt[6]{b}]$ to the Jacobian of another genus 2 hyperelliptic curve $\mathcal{C}_6'$ defined over $\mathbb{F}_q[\sqrt{b}]$. This Jacobian $J_{\mathcal{C}_6'}$ splits into two elliptic curves E_c and E_{-c} defined over $\mathbb{F}_q[\sqrt{b}]$ which are isogenous over $\mathbb{F}_q[\sqrt{b}, \sqrt{-3}]$. Let $c = a/\sqrt{b}$ and assume $c \neq \pm 2$. The two elliptic curves are defined (in a reduced form) by

$$\begin{aligned} E_c^{\text{red}}(\mathbb{F}_q[\sqrt{b}]) &: Y^2 = X^3 + 3(2c-5)X + c^2 - 14c + 22 \text{ and} \\ E_{-c}^{\text{red}}(\mathbb{F}_q[\sqrt{b}]) &: Y^2 = X^3 - 3(2c+5)X + c^2 + 14c + 22 \ . \end{aligned}$$

Freeman and Satoh remarked that both elliptic curves admit the same 3-torsion subgroup ([14, Proof of Prop. 4.2]). With Vélu's formulas adapted to finite fields (e.g. [31, p. 54]), we compute an isogeny from E_c into E_{-c} with kernel equal to this 3-torsion subgroup. Because of this isogeny, E_c and E_{-c} have the same order over $\mathbb{F}_q[\sqrt{b}, \sqrt{-3}]$ and moreover, this order is a multiple of 3.

$$\begin{array}{ccc} J_{\mathcal{C}_6}(\mathbb{F}_q[\sqrt[6]{b}]) & \overset{\text{isogeny}}{\longleftrightarrow} & J_{\mathcal{C}_6'}(\mathbb{F}_q[\sqrt[6]{b}]) \\ | & & | \\ J_{\mathcal{C}_6}(\mathbb{F}_q[\sqrt{b}]) & & J_{\mathcal{C}_6'}(\mathbb{F}_q[\sqrt{b}]) \overset{\text{isogeny}}{\longleftrightarrow} (E_c \times E_{-c})(\mathbb{F}_q[\sqrt{b}]) \\ | & & \\ J_{\mathcal{C}_6}(\mathbb{F}_q) & & \end{array}$$

In the two cases where b is not a cube, we have to deduce $Z_{J_{\mathcal{C}_6}}(T, \mathbb{F}_q)$ from $Z_{J_{\mathcal{C}_6}}(T, \mathbb{F}_{q^3})$ or $Z_{J_{\mathcal{C}_6}}(T, \mathbb{F}_{q^2})$ from $Z_{J_{\mathcal{C}_6}}(T, \mathbb{F}_{q^6})$ which is equivalent. Note that we do not see explicitly the simplification in the formula if we descent from $\mathbb{F}_{q^6}$ to $\mathbb{F}_{q^3}$ then to $\mathbb{F}_q$.

Eventually, we obtain the following theorem:

Theorem 2. *Let $\mathcal{C}_6$ be a hyperelliptic curve defined over a finite field $\mathbb{F}_q$ by the equation $\mathcal{C}_6(\mathbb{F}_q) : Y^2 = X^6 + aX^3 + b$ with $a, b \neq 0 \in \mathbb{F}_q$. Let E_c and E_{-c} be the elliptic curves defined over $\mathbb{F}_q[\sqrt{b}]$ isogenous over $\mathbb{F}_q[\sqrt{b}, \sqrt{-3}]$. Let t_{q^2} be the trace of $E_c(\mathbb{F}_{q^2})$ and let t_q be the trace of $E_c(\mathbb{F}_q)$ if it exists.*

1. *If b is a sixth power then* $\#J_{\mathcal{C}_6}(\mathbb{F}_q) = (q+1-t_q)^2$ *if* $\sqrt{-3} \in \mathbb{F}_q$ *and* $\#J_{\mathcal{C}_6}(\mathbb{F}_q) = (q+1-t_q)(q+1+t_q)$ *if* $\sqrt{-3} \notin \mathbb{F}_q$.
2. *If b is a square but not a third power and if* $3(4q-(t_q)^2)$ *is not a square then* $\#J_{\mathcal{C}_6}(\mathbb{F}_q) = q^2 - q + 1 + (1+q+t_q)t_q$.
3. *If b is a third power but not a square and if* $2q + t_{q^2}$ *is not a square then* $\#J_{\mathcal{C}_6}(\mathbb{F}_q) = q^2 + 1 - t_{q^2}$.
4. *If b is neither a cube nor a square and if* $2q + t_{q^2}$ *is not a square, then there exists* $n \in \mathbb{N}$ *such that* $2q - t_{q^2} = 3n^2$ *and* $\#J_{\mathcal{C}_6}(\mathbb{F}_q) = q^2+q+1+(q+1+n)3n$ *or* $\#J_{\mathcal{C}_6}(\mathbb{F}_q) = q^2 + q + 1 - (q+1-n)3n$.

Once more, the first case is not interesting in cryptography as the Jacobian order splits. The third case provides nothing more than an elliptic curve defined over $\mathbb{F}_{q^2}$. Whenever the group law computation on $J_{\mathcal{C}_6}(\mathbb{F}_q)$ is not as efficient as a point addition on $E_c(\mathbb{F}_{q^2})$, it will be more appropriate to work with the elliptic curve. The case 2 might be interesting. The case 4 provides interesting genus 2 hyperelliptic curves.

Remark 2. With the same notations as in the previous theorem,

1. If b is a square but not a third power and if $4q - (t_q)^2 = 3y^2$ then $\#J_{\mathcal{C}_6}(\mathbb{F}_q)$ equals one of $(q+1+(t_q+3y)/2)(q+1+(t_q-3y)/2) = q^2-q+1+(1+q+t_q)t_q$, $(q+1+(t_q+3y)/2)^2$, $(q+1+(t_q-3y)/2)^2$.
2. If b is a third power but not a square and if $2q + t_{q^2} = y^2$ is a square then $\#J_{\mathcal{C}_6}(\mathbb{F}_q)$ equals one of $(q+1-y)^2$, $(q+1+y)^2$, $(q+1-y)(q+1+y) = q^2+1-t_{q^2}$.
3. If b is neither a cube nor a square,
   (a) if $2q + t_{q^2} = s^2$, $s \in \mathbb{N}$, and $3(2q - t_{q^2})$ is not a square then $\#J_{\mathcal{C}_6}(\mathbb{F}_q) = q^2 - q + 1 - (1+q)s + s^2$ or $q^2 - q + 1 + (1+q)s + s^2$.
   (b) if $2q + t_{q^2} = s^2$ and $2q - t_{q^2} = 3n^2$, $\#J_{\mathcal{C}_6}(\mathbb{F}_q)$ splits and equals one of $q^2+q+1+(q+1+n)3n$, $q^2+q+1-(q+1-n)3n$, $q^2-q+1-(q+1-s)s$, $q^2-q+1+(q+1+s)s$, $q^2+1-(-t_{q^2}+3y)/2)$, $q^2+1-(-t_{q^2}-3y)/2)$, $(q+1+\frac{s-3n}{2})^2$, $(q+1-\frac{s-3n}{2})^2$, $(q+1+\frac{s+3n}{2})^2$, $(q+1-\frac{s+3n}{2})^2$.

Example 3. We consider the 127-bit Mersenne prime $p = 2^{127} - 1$ which allows efficient implementation of the modular arithmetic operations required in cryptography. Looking for a curve $\mathcal{C}_6$ over $\mathbb{F}_p$ with small parameters a and b and suitable for a cryptographic use, we found easily $\mathcal{C}_6(\mathbb{F}_p) : Y^2 = X^6 - 3X^3 - 92$ with $b = -92$ which is neither a square nor a cube. Let $\mathbb{F}_{p^2} = \mathbb{F}_p[X]/(X^2+1) = \mathbb{F}_p[i]$, $c = a/\sqrt{b} \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$ and $E_c(\mathbb{F}_{p^2}) : Y^2 + X^3 + 3(2c-5)X + c^2 - 14c + 22$. A few second computation gives us $t_{p^2} =$`0x6089c0341e5414a24bef1a1a93c54fd2` and $2p - t_{p^2} = 3n^2$ as expected with $n = \pm$ `0x74a69cde5282dbb6`. Hence $\#J_{\mathcal{C}_6}(\mathbb{F}_p) = p^2 + p + 1 + 3n(p+1) + 3n^2$. Using few random points on the Jacobian, we find $n < 0$ and that $\#J_{\mathcal{C}_6}(\mathbb{F}_p)$ has a 250-bit prime factor:
$r =$ `0x25ed097b425ed0974c75619931ea7f1271757b237c3ff3c5c00a037e7906557` and provides a security level close to 128-bits.

Efficiency. For a cryptographic application, we need $\#J_{\mathcal{C}}(\mathbb{F}_q)$ be a large prime r times a small (*i.e.* few bits) cofactor h. The prime r must be of size twice the security level in bits. The common method consists in randomly generating the coefficients a and b of the hyperelliptic curve and computing the order until it is a large prime r times a small cofactor.

Here $r \sim q^2$ hence the size of q is a few bits more than the security level in bits instead of twice with an elliptic curve. To compute the Jacobian order, we have to run SEA algorithm once for an elliptic curve defined over $\mathbb{F}_q$ if b is a square or over $\mathbb{F}_{q^2}$ otherwise. If b is a square our method is much faster than generating a cryptographic elliptic curve and if b is not a square, our method is roughly as efficient as finding an elliptic curve suitable for cryptography.

4 Pairing-Friendly Constructions

We have several constraints for suitable pairing-friendly constructions inherent to elliptic curves:

1. The embedding degree k must be small, in order to achieve the same security level in bits in the elliptic curve r-torsion subgroup $E(\mathbb{F}_p)[r]$ and in the finite field extension $\mathbb{F}_{p^k}$. In practice, this means $6 \leqslant k \leqslant 60$. More precise recommendations are given in [12, Tab. 1]. For a random elliptic curve, we have usually $k \simeq r$ so this is a huge constraint.
2. The trace t of the curve must satisfy $|t| \leqslant 2\sqrt{p}$.
3. The determinant of the curve $\Delta = t^2 - 4p = -Dy^2$ must have a very small square-free part $D < 10^9$ in order to run the CM-method in reasonable time.
4. The size $\log r$ of the subgroup must be close to the optimal case, that is $\rho = g \log p / \log r \sim 1$ with g the genus of the curve. Quite generic methods for elliptic curves achieve $1 \leqslant \rho \leqslant 2$. We will try to find constructions for genus 2 curves with $2 \leqslant \rho \leqslant 4$.

The two methods use the same shortcuts in formulas. Let E an elliptic curve and let $\#E(\mathbb{F}_p) = p + 1 - t = hr$ with r a large prime and h the cofactor. Hence $p \equiv t - 1 \mod r$. Let $\Delta = t^2 - 4p = -Dy^2$. The second useful formula is $Dy^2 = 4p - t^2 = 4hr - (t-2)^2$, hence $-Dy^2 \equiv (t-2)^2 \mod r$.

4.1 Cocks-Pinch Method

We first recall the method proposed by Cocks and Pinch in 2001 to construct pairing-friendly elliptic curves [9] (see also [16, Algorithm IX.4]):

We propose to adapt this method to the Jacobian families of cryptographic interest presented above. See the size recommendations in [2, Tab. 3.1] depending on the security level in bits to choose accordingly the embedding degree. First, we know explicitly the Jacobian order. Just as in the case of elliptic curves, the definition of the embedding degree is equivalent to ask for $r \mid \#J_{\mathcal{C}}(\mathbb{F}_p)$ and $r \mid \Phi_k(p)$. We will use the property $p \equiv \zeta_k \mod r$ as well. The aim is to express the other parameters, namely the square part y and the trace of the elliptic curve

Algorithm 1. Cocks-Pinch method to find a pairing-friendly elliptic curve

Input: Square-free integer D, size of r and embedding degree k to match the security level in bits, knowing that $\rho \approx 2$.

Output: Prime order r, prime number p, elliptic curve parameters $a, b \in \mathbb{F}_p$ such that $E(\mathbb{F}_p) : Y^2 = X^3 + aX + b$ has a subgroup of order r and embedding degree k with respect to r.

1 **repeat**
2 Pick at random a prime r of prescribed size until $-D$ is a square in the finite field $\mathbb{F}_r$ and $\mathbb{F}_r$ contains a primitive k-th root of unity ζ_k, that is $r \equiv 1 \mod k$.
3 As r divides $\Phi_k(p)$, we can rewrite it as $\Phi_k(p) \equiv 0 \mod r$. With properties of cyclotomic polynomials, we obtain $p \equiv \zeta_k \mod r$ with ζ_k a primitive k-th root of unity. Furthermore, $t \equiv 1 + p \mod r$ so this method chooses $t = 1 + \zeta_k$ in $\mathbb{F}_r$. Then $y = (t-2)/\sqrt{-D}$ in $\mathbb{F}_r$.
4 Lift t and y from $\mathbb{F}_r$ to $\mathbb{Z}$ and set $p = \frac{1}{4}(t^2 + Dy^2)$.
5 **until** p *is prime.*
6 **return** r, p, $a, b \in \mathbb{F}_p$

isogenous to the Jacobian over some extension field, in terms of $\zeta_k \mod r$. We will use the same notations as previously, see Th.1 and Th.2. Let i be a primitive fourth root of unity an ω be a primitive third root of unity in $\mathbb{F}_r$.

Pairing-Friendly Hyperelliptic Curve $\mathcal{C}_5$

If b is not a square in $\mathbb{F}_p$ but $\sqrt{b}, \sqrt[4]{b} \in \mathbb{F}_{p^2}$ ($p \equiv 3 \mod 4$), then $\#J_{\mathcal{C}_5}(\mathbb{F}_p) = \#E_1(\mathbb{F}_{p^2}) = p^2 + 1 - t_{p^2}$ (Th.1(3.)). A pairing-friendly Jacobian of this type has exactly the same order as the corresponding elliptic curve $E_1(\mathbb{F}_{p^2})$. Hence any pairing-friendly elliptic curve defined over a quadratic extension $\mathbb{F}_{p^2}$ (and of even order) will provide a pairing-friendly Jacobian of this type over the prime field $\mathbb{F}_p$, with the same order and the same ρ-value. Choosing the Jacobian instead of the elliptic curve will be appropriate only if the group law on the Jacobian over $\mathbb{F}_p$ is faster than the group law on the elliptic curve over $\mathbb{F}_{p^2}$. Note that the methods described in [12] are suitable for generating pairing-friendly elliptic curves over prime fields (in large characteristic), not over field extensions.

$\mathcal{C}_5$ with b a Square but Not a Fourth Power. This case is already almost solved in [14]. The Cocks-Pinch method adapted with $r \mid \#J_{\mathcal{C}_5}(\mathbb{F}_p) = (p-1)^2 + (t_p')^2$ instead of $r \mid p + 1 - t_p'$ produces indeed the same algorithm as [14, Alg. 5.5] followed by [14, Alg. 5.11] with $\pi = (t_p' - y\sqrt{-D})/2$, $d = 4$. We show that $d \mid k$ is unnecessary. It is completely hopeless to expect a prime power $q = \pi\overline{\pi} = p^n$ hence we assume that $q = p$ is prime.

Definition 1. *Embedding degree and embedding field[3, Def. 2.1 and 2.2] Let A be an abelian variety defined over $\mathbb{F}_q$, where $q = p^m$ for some prime p and*

integer m. Let $r \neq p$ be a prime dividing $\#A(\mathbb{F}_q)$. The embedding degree *of A with respect to r is the smallest integer k such that r divides $q^k - 1$.*

The minimal embedding field *of A with respect to r is the smallest extension of $\mathbb{F}_p$ containing the rth roots of unity $\mu_r \subset \mathbb{F}_p$.*

Let k be the embedding degree of the Jacobian $J_{\mathcal{C}_5}(\mathbb{F}_p)$: $r \mid \#J_{\mathcal{C}_5}(\mathbb{F}_p)$, $r \mid \Phi_k(p)$. From the Jacobian point of view, there is no security problem induced by a difference between embedding degree and embedding field because $\mathbb{F}_p$ is a prime field. From elliptic curve side, the one-dimensional part of the r-torsion arises in $E_1'(\mathbb{F}_{p^4})$, not below. An elementary observation about elliptic curve orders shows that

$$\begin{aligned}
\#E_1'(\mathbb{F}_p) &= p + 1 - t_p' \\
\#E_1'(\mathbb{F}_{p^2}) &= (p + 1 - t_p')(p + 1 + t_p') \\
\#E_1'(\mathbb{F}_{p^4}) &= (p + 1 - t_p')(p + 1 + t_p')((p+1)^2 + (t_p')^2)
\end{aligned}$$

and the last factor of $\#E_1'(\mathbb{F}_{p^4})$ is the Jacobian order. Hence $r \mid \#E_1'(\mathbb{F}_{p^4})$ but not underneath. The full r-torsion arises in $E_1'(\mathbb{F}_{p^{4k/\gcd(4,k)}})$ but the embedding field is $\mathbb{F}_{p^k}$. So the elliptic curve $E_1'(\mathbb{F}_{p^4})$ will not be suitable for a pairing implementation when $\gcd(k, 4) \in \{1, 2\}$ which does not matter because we are interested in Jacobians suitable for pairing, not elliptic curves. See Fig. 1.

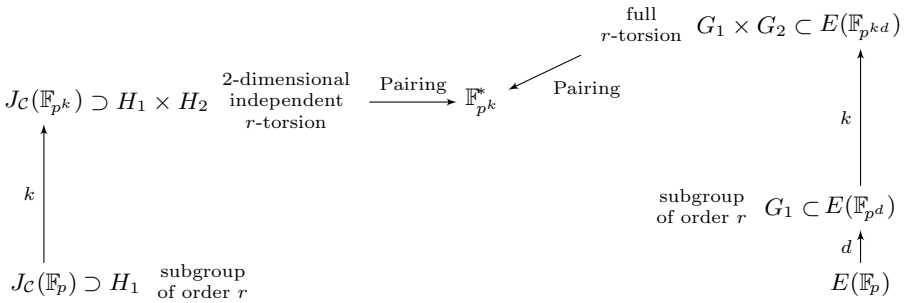


Fig. 1. Difference between Jacobian and elliptic curve embedding degree

Moreover we note that taking an even trace t_p' and a prime $p \equiv 1 \mod 4$ permits always to find valid parameters, namely a $c \in \mathbb{F}_p$ satisfying the j-invariant equation, hence coefficients $a, b \in \mathbb{F}_p$ of $\mathcal{C}_5$.

$\mathcal{C}_5$ with b not a square and $p \equiv 1 \mod 4$. In this case we have $\sqrt[4]{b} \notin \mathbb{F}_{p^2}$, $\sqrt[4]{b} \in \mathbb{F}_{p^4}$ and $\#J_{\mathcal{C}_5}(\mathbb{F}_p) = p^2 + 1 + 2n^2 - 2n(1+p) = (p-n)^2 + (n-1)^2$ with $2p \pm t_{p^2}' = 2n^2$. The isogenous elliptic curve is defined over $\mathbb{F}_{p^2}$. We have $\Delta = (t_{p^2}')^2 - 4p^2 = (t_{p^2}' + 2p)(t_{p^2}' - 2p)$. With $2p - t_{p^2}' = 2n^2$ we obtain $2p + t_{p^2}' = 4p - 2n^2$ and find $\Delta = -4n^2(2p - n^2)$. With $2p + t_{p^2}' = 2n^2$ we obtain $2p - t_{p^2}' = 4p - 2n^2$ and find

also $\Delta = -4n^2(2p - n^2)$. In both cases let $Dy^2 = 2p - n^2$ thus $\Delta = -D(2ny)^2$ and $p = (Dy^2 + n^2)/2$. The Jacobian order is a sum of two squares in p and n hence $n = (p+i)/(1+i) = (p+i)(1-i)/2 \mod r$. Furthermore $y^2 \equiv (2p-n^2)/D \mod r$ with $p \equiv \zeta_k \mod r$ and we find that

$n \equiv (\zeta_k + i)(1 - i)/2 \mod r$ and $y \equiv \pm(\zeta_k - i)(1 + i)/(2\sqrt{D}) \mod r$.

The trace will be even by construction as $t'_{p^2} = \pm(2p - 2n^2)$ and to find valid parameters, $p \equiv 1 \mod 4$ is required. To find the coefficients of the curve $\mathcal{C}_5(\mathbb{F}_p)$, do the following (Alg. 2).

Algorithm 2. Pairing-friendly Jacobian of type $J_{\mathcal{C}_5}$, Th.1(4)

Input: Square-free integer D, size of r and embedding degree k to match the security level in bits, knowing that $\rho \approx 4$.

Output: Prime order r, prime number p, Jacobian parameters $a, b \in \mathbb{F}_p$ such that the Jacobian of the curve $\mathcal{C}_5(\mathbb{F}_p) : Y^2 = X^5 + aX^3 + bX$ has a subgroup of order r and embedding degree k with respect to r.

1 **repeat**
2 Choose a prime r of prescribed size with $i, \sqrt{D}, \zeta_k \in \mathbb{F}_r$.
3 Let $n = (\zeta_k + i)(1 - i)/2$ and $y = \pm(\zeta_k - i)(1 + i)/(2\sqrt{D}) \in \mathbb{F}_r$.
4 Lift n and y from $\mathbb{F}_r$ to $\mathbb{Z}$ and set $p = (n^2 + Dy^2)/2$.
5 **until** $p \equiv 1 \mod 4$ *and* p *is prime.*
6 Run the CM method to find the j-invariant of an elliptic curve $E'_1(\mathbb{F}_{p^2})$ of trace $\pm t'_{p^2}$ and $\Delta = -4D(ny)^2$.
7 Solve $j(E'_1) = 2^6 \frac{(3c-10)^3}{(c-2)(c+2)^2}$ in $\mathbb{F}_{p^2}$ and choose the solution satisfying $c^2 \in \mathbb{F}_p$.
8 Choose $a, b \in \mathbb{F}_p$ such that $a \neq 0$ and $b = (a/c)^2$ (b is a square in $\mathbb{F}_{p^2}$ but not in $\mathbb{F}_p$).
9 **return** r, p, $a, b \in \mathbb{F}_p$

We adapt the program `cm.cpp` of Miracl[2] [37] to compute the j-invariant of an elliptic curve defined over $\mathbb{F}_{p^2}$ (instead of $\mathbb{F}_p$). Indeed, it is not convenient for step 5 as it searches for an elliptic curve defined over a prime field. We isolate parts of the program which compute the Weber polynomial of a number field of discriminant D. Then we call the `factor` function but to find a factor $\mod p$ of degree 2 (instead of degree 1) of the Weber polynomial when $D \not\equiv 3 \mod 8$ and a factor of degree 6 (instead of degree 3) when $D \equiv 3 \mod 8$. The papers [30,29] contain efficient formulas to recover Hilbert polynomial roots in $\mathbb{F}_p$ from Weber polynomial roots in $\mathbb{F}_p$ or $\mathbb{F}_{p^3}$. We find in $\mathbb{F}_{p^2}$ or $\mathbb{F}_{p^6}$ a root of the factor of degree 2 or 6 of Weber polynomial and apply the corresponding transformation to get an element in $\mathbb{F}_{p^2}$. We obtain the j-invariant of (an isogenous curve to) the curve $E'_1(\mathbb{F}_{p^2})$. We solve $j(E'_1) = 2^6 \frac{(3c-10)^3}{(c-2)(c+2)^2}$ and find for various examples

[2] We learned very recently that the MIRACL library status has changed. This library is now a commercial product of Certivox [8]. The CM software [11] can be an even more efficient alternative to compute class polynomials.

a solution $c \in \mathbb{F}_{p^2}$ satisfying $c^2 \in \mathbb{F}_p$. It comes from the appropriate restrictions $2p \pm t'_{p^2} = 2n^2$, $p \equiv 1 \mod 4$, n odd. Sometimes we have to choose a quadratic twist of $\mathcal{C}_5$, of the form $Y^2 = \nu(X^5 + aX^3 + bX)$ with $\nu \in \mathbb{F}_p$ non-square.

Example 4. $k = 6$, $D = 516505$, $\rho = 4.1$
$p =$`0x9d3e97371e27d006f11762f0d56b4fbf2caca7d606e92e8b6f35189723f46f57ed46e9650ce1cca1bd90dc393db35cc38970cb0abbe236bf2c4ac2f65f1b50afb135` (528 bits),
$r =$`0x679d8c817e0401203364615b9d34bdb3a0b89e70fa8d6807fa646e25140f25ad`(255b),
$n =$`0x28f34a88ab9271c2ea6d70f4a3dc758a025ad6e4ee51c16867763e8d940022de5`,
$y =$`-0x65110defe8f4669a158149675afaa23dba326d49ce841d7ef9855c7d8a65df95`,
$a = 1$, $b =$`0x85eb6f5b5594c1bca596a53066216ad79588cf39984314609bbd7a3a302241fc786703a19bc1ccb44fc9e09b9c17ac62fc38d6bf82851d3d8b753c79da7338ca56b0`,
$\mathcal{C}_5(\mathbb{F}_p) : Y^2 = 2(X^5 + aX^3 + bX)$.

Pairing-Friendly Hyperelliptic Curve $\mathcal{C}_6$

If b is a cube but not a square then $\#J_{\mathcal{C}_6}(\mathbb{F}_p) = p^2 + 1 - t_{p^2}$ (Th.2(3)). This case is close to the elliptic curve case. Actually, this is the same construction as finding a pairing-friendly elliptic curve over a field $\mathbb{F}_{p^2}$. But in practice the methods to find such pairing-friendly elliptic curves over $\mathbb{F}_p$ fail over $\mathbb{F}_{p^2}$. Indeed, the expression for p is $p^2 = \frac{1}{4}((t'_{p^2})^2 + Dy^2)$ but this is hopeless to find a prime square. We did not find in the literature any such construction.

$\mathcal{C}_6$ with b a square but not a cube. This case is treated in [14, Alg. 5.5, Alg. 5.11] and corresponds to $d = 3$ and $\pi = (t_p - y\sqrt{-D})/2$. This is also a Cocks-Pinch-like method with $r \mid p^2 - p + 1 + (1 + p)t_p + (t_p)^2$ and $r \mid \Phi_k(p)$. As above for $\mathcal{C}_5$, the condition "$3 \mid k$" is not necessary since we consider the embedding degree of the Jacobian, not the elliptic curve.

We found that $p \equiv 1 \mod 3$ and $p + 1 \pm t_p \equiv 0 \mod 3$ are enough to find always valid parameters. Freeman and Satoh pointed out that the equation $j(E_c) = 2^8 3^3 (2c-5)^3/((c-2)(c+2)^3)$ has a solution in $\mathbb{F}_p$ in only one third of the cases [14, § 6]. One can explain this phenomenon by simple arithmetic considerations.

The elliptic curve E_c has a 3-torsion point which means $p+1-t_p \equiv 0 \mod 3$, which happens one third of the cases when $p \equiv 1 \mod 3$. Assuming that $p \equiv 1 \mod 3$, if $p + 1 + t_p \equiv 0 \mod 3$ then $E_c(\mathbb{F}_p)$ has not 3-torsion point but its quadratic twist has. These two elliptic curves have the same j-invariant and admit a 3-torsion subgroup over $\mathbb{F}_{p^2}$. In practice we verify that the equation has a solution when $p + 1 \pm t_p \equiv 0 \mod 3$. Combining the two conditions $p \equiv 1 \mod 3$ and $p + 1 \pm t_p \equiv 0 \mod 3$, the equation from $j(E_c)$ has indeed a solution one third of the time ($\frac{1}{2} \cdot \frac{2}{3}$). When $p \equiv 1 \mod 3$ and $t_p \equiv 2 \mod 3$, we can always find a solution in step 2 of [14, Alg. 5.11] and finish to run this algorithm. When $p \equiv 1 \mod 3$ and $t_p \equiv 1 \mod 3$, we can still find a solution in step 2 and construct the coefficients of $\mathcal{C}_6(\mathbb{F}_p)$ in step 3 of [14, Alg. 5.11]. But in step 6, we have to choose not $\mathcal{C}_6$ itself but its quadratic twist.

$\mathcal{C}_6$ *with* b *neither a square nor a cube.* $\#J_{\mathcal{C}_6}(\mathbb{F}_p) = p^2 + p + 1 - (p+1)3n + 3n^2$. Here the parameters satisfy $2p - t_{p^2} = 3n^2$. Let $2p + t_{p^2} (= 4p - 3n^2) = Dy^2$. Hence

$$p = \frac{1}{4}\left(3n^2 + Dy^2\right) \ .$$

Note that $3 \nmid D$ otherwise p would not be prime. Solving $p^2 + p + 1 - (p+1)3n + 3n^2 \equiv 0 \mod r$ gives $p = (1-\omega^2)n + \omega^2$ or $p = (1-\omega)n + \omega$ with ω a primitive third root of unity. As $y^2 = (4p - 3n^2)/D \mod r$ and with $p \equiv \zeta_k \mod r$ we find

$$n \equiv (\zeta_k - \omega)/(1-\omega) \mod r \text{ and } y \equiv \pm(\omega\zeta_k + \omega^2)/\sqrt{D} \mod r \ .$$

The last version of the Cocks-Pinch method is presented in Alg. 3.

Algorithm 3. Pairing-friendly Jacobian of type $J_{\mathcal{C}_6}$, Th.2(4)

Input: Square-free integer D, $3 \nmid D$, size of r and embedding degree k to match the security level in bits, knowing that $\rho \approx 4$.

Output: Prime order r, prime number p, Jacobian parameters $a, b \in \mathbb{F}_p$ such that the Jacobian of the curve $\mathcal{C}_6(\mathbb{F}_p) : Y^2 = X^6 + aX^3 + b$ has a subgroup of order r and embedding degree k with respect to r.

1 **repeat**
2 Choose a prime r of prescribed size such that a third root of unity ω, $\sqrt{D}$ and $\zeta_k \in \mathbb{F}_r$.
3 Let $n = (\zeta_k - \omega)/(1-\omega)$ and $y = \pm(\omega\zeta_k + \omega^2)/\sqrt{D} \in \mathbb{F}_r$.
4 Lift n and y from $\mathbb{F}_r$ to $\mathbb{Z}$ and set $p = (3n^2 + Dy^2)/4$.
5 **until** $p \equiv 1 \mod 3$ *and* p *is prime.*
6 Run the CM method to find the j-invariant of an elliptic curve $E_c(\mathbb{F}_{p^2})$ of trace t_{p^2} and $\Delta = -3D(ny)^2$. More precisely, run the CM method with $3D$. Find a degree 2 or 6 factor of the Weber polynomial $\mod p$, then apply the right transformation from [30,29] to obtain a root in $\mathbb{F}_{p^2}$ of the corresponding Hilbert polynomial.
7 Solve $j(E_c) = 2^8 3^3 \frac{(2c-5)^3}{(c-2)(c+2)^3}$ in $\mathbb{F}_{p^2}$ and choose a solution $c \in \mathbb{F}_{p^2}$ such that $c^2 \in \mathbb{F}_p$. Choose $a, b \in \mathbb{F}_p$ such that $(a/c)^2$ is not a cube and $b = (a/c)^2$. Hence b is neither a square nor a cube.
8 **return** r, p, $a, b \in \mathbb{F}_p$

4.2 Brezing-Weng Method

The method proposed by Brezing-Weng is to use a polynomial ring built with a cyclotomic polynomial instead of a finite prime field $\mathbb{F}_r$. The parameters will be polynomials modulo a cyclotomic polynomial instead of integers modulo a prime. But the choice of D is limited to few values. We tried with D square-free in the range 1 - 35 according to the embedding degree $5 \leqslant k \leqslant 36$. We ran a search (with Magma [6]) over different cyclotomic fields and with a change of basis as in [24] and [25]. We obtained complete families with $\rho \simeq 3$ and recover constructions already mentioned in previous papers [26,14] and new complete families for other embedding degrees:

Example 5 ($k = 22, D = 2, \rho = 2.8$).

$$\begin{aligned} r &= \Phi_{88}(x) = x^{40} - x^{36} + x^{32} - x^{28} + x^{24} - x^{20} + x^{16} - x^{12} + x^8 - x^4 + 1 \\ n &= \tfrac{1}{2}\left(x^{28} - x^{22} - x^6 + 1\right) \\ y &= \tfrac{1}{2}\left(x^{17} + x^{11}\right) \\ t'_{p^2} &= \tfrac{1}{4}\left(-x^{56} + 2x^{50} - x^{44} + 4x^{34} + 4x^{22} - x^{12} + 2x^6 - 1\right) \\ p &= \tfrac{1}{8}\left(x^{56} - 2x^{50} + x^{44} + 8x^{28} + x^{12} - 2x^6 + 1\right) \\ x &\equiv 1 \mod 2 \end{aligned}$$

Example 6 ($k = 26, D = 2, \rho = 2.33$).

$$\begin{aligned} r &= \Phi_{104}(x) = x^{48} - x^{44} + x^{40} - x^{36} + x^{32} - x^{28} + x^{24} - x^{20} + x^{16} - x^{12} + x^8 - x^4 + 1 \\ n &= \tfrac{1}{2}\left(x^{28} - x^{26} - x^2 + 1\right) \\ y &= \tfrac{1}{2}\left(x^{15} + x^{13}\right) \\ t'_{p^2} &= \tfrac{1}{4}\left(-x^{56} + 2x^{54} - x^{52} + 4x^{30} + 4x^{26} - x^4 + 2x^2 - 1\right) \\ p &= \tfrac{1}{8}\left(x^{56} - 2x^{54} + x^{52} + 8x^{28} + x^4 - 2x^2 + 1\right) \\ x &\equiv 1 \mod 2 \end{aligned}$$

Some constructions ($k \in \{7, 17, 19, 23, 29, 31\}$) have a cyclotomic polynomial of too high degree for r. Hence there are very few possibilities for choosing a suitable integer x such that $p(x)$ and $r(x)$ are prime and of the desired size. Moreover the ρ-value is close to 4. It would be preferable to use the Cocks-Pinch-like method.

5 More Pairing-Friendly Constructions with $D = 1, 2, 3$

We observed that when $D = 1$, the obtained genus 2 hyperelliptic curve of the form $\mathcal{C}_5(\mathbb{F}_p)$ with b a square splits actually into two non-isogenous elliptic curves over $\mathbb{F}_p$. We observed the same decomposition for genus 2 hyperelliptic curve of the form $\mathcal{C}_6$ obtained with $D = 3$ and b a square but not a cube. A theoretical explanation can be found in [14, Proposition 3.10]. ¿From Rem. 1 1 we get the explicit decomposition. We give here a practical point of view from explicit zeta function computation. Let $E_1(\mathbb{F}_q)$ be an elliptic curve defined over a finite field $\mathbb{F}_q$ of trace t_q an satisfying $(t_q)^2 - 4q = -y^2$, *i.e.* $D = 1$. The zeta function of E_1 is $Z_{E_1}(T, \mathbb{F}_q) = T^2 - t_q T + q = (T - \frac{t_q + iy}{2})(T - \frac{t_q - iy}{2})$ with $i \in \mathbb{C}$ such that $i^2 = -1$. We will use the notation $\alpha = \frac{t_q + iy}{2}$. With the formula given in [14, Proposition 3.4] we find that the zeta function of the order 4 Weil restriction of $E_1(\mathbb{F}_q)$ is

$$Z_{J_{\mathcal{C}_5}}(T, \mathbb{F}_q) = (T - i\alpha)(T + i\alpha)(T - i\overline{\alpha})(T + i\overline{\alpha}) = (T^2 - yT + q)(T^2 + yT + q) \ .$$

Note that $q + 1 - y$ and $q + 1 + y$ are the orders of the two quartic twists of $E_1(\mathbb{F}_q)$. Hence the obtained Jacobian always splits into the two quartic twists of $E_1(\mathbb{F}_q)$.

For $J_{\mathcal{C}_6}(\mathbb{F}_q)$ and $D = 3$ when b is a square but not a cube, a similar computation explains the matter. Here E_c is an elliptic curve defined over $\mathbb{F}_q$ of trace t_q and such that $(t_q)^2 - 4q = -3y^2$. Let us denote $\alpha = \frac{t_q + i\sqrt{3}y}{2}$ one of the two roots of its zeta function. The zeta function of the order 3 Weil restriction of $E_c(\mathbb{F}_q)$ is

$$Z_{J_{\mathcal{C}_6}}(T, \mathbb{F}_q) = \left(T^2 + \tfrac{t+3y}{2}T + q\right)\left(T^2 + \tfrac{t-3y}{2}T + q\right) \ .$$

We recognize the two cubic twists of $E_c(\mathbb{F}_q)$. This confirm the results found in Rem 2.1.1. Trying with an order 6 Weil restriction, we find

$$Z_{J_{C_6}}(T, \mathbb{F}_q) = \left(T^2 - \tfrac{t-3y}{2}T + q\right)\left(T^2 - \tfrac{t+3y}{2}T + q\right) .$$

Hence the Jacobian splits into the two sextic twists of $E_c(\mathbb{F}_q)$. We recognize a case described in Rem. 2.3.b. Freeman and Satoh suggested to construct an order 8 Weil restriction when $D = 1, 2$ and an order 12 Weil restriction when $D = 3$. For $k = 32, 64, 88$ and $D = 2$ this order 8 Weil restriction corresponds to families previously found by Kawazoe and Takahashi.

5.1 Order-8 Weil Restriction When D = 1

Let $E(\mathbb{F}_p)$ an elliptic curve defined over a prime field $\mathbb{F}_p$, of trace t_p and satisfying $(t_p)^2 - 4p = -y^2$ (that is, $D = 1$). The two roots of its zeta function over $\mathbb{C}$ are $\alpha = (t_p + iy)/2$ and $\overline{\alpha}$. Let ζ_8 denotes an eighth root of unity. The zeta function of the order 8 Weil restriction of $E(\mathbb{F}_p)$ is

$$\begin{aligned} Z(T, \mathbb{F}_p) &= \left((T-\zeta_8\alpha)(T-\zeta_8^7\overline{\alpha})(T-\zeta_8^5\alpha)(T-\zeta_8^3\overline{\alpha})\right)\left((T-\zeta_8^3\alpha)(T-\zeta_8^5\overline{\alpha})(T-\zeta_8^7\alpha)(T-\zeta_8\overline{\alpha})\right) \\ &= (T^4 + tyT^2 + p^2)(T^4 - tyT^2 + p^2) \end{aligned}$$

We see this zeta function factors as two degree 4 zeta functions, that is into two genus 2 hyperelliptic curve zeta functions. So we start from an elliptic curve $E(\mathbb{F}_p)$ as above, with $(t_p)^2 - 4p = -y^2$ and search for suitable p, t, y such that there exists a genus 2 hyperelliptic curve of order $\#J_{\mathcal{C}}(\mathbb{F}_p) = p^2 + 1 \pm ty$ suitable for pairing-based cryptography.

To apply one of the two previous methods (Cocks-Pinch or Brezing-Weng), we have to find an expression of t and y in terms of p modulo r.

$$t = \zeta_8 + \zeta_8^7\zeta_k \text{ and } y = -\zeta_8^7 - \zeta_8\zeta_k \mod r .$$

To finish, $p = (t^2 + y^2)/4$.

Example 7 ($k = 8, D = 1, \rho = 3.0$).
$r = x^4 + 2x^2 + 4x + 2$
$t = x$
$y = \frac{1}{3}(-x^3 + 2x^2 - 3x + 2)$
$p = \frac{1}{36}(x^6 - 4x^5 + 10x^4 - 16x^3 + 26x^2 - 12x + 4)$
$x \equiv 4 \mod 6$

5.2 Order-8 Weil Restriction When D = 2

Let $E(\mathbb{F}_p)$ an elliptic curve defined over a prime field $\mathbb{F}_p$, of trace t_p and satisfying $(t_p)^2 - 4p = -2y^2$ (that is, $D = 2$). The two roots of its zeta function over $\mathbb{C}$ are $\alpha = (t_p + i\sqrt{2}y)/2$ and $\overline{\alpha}$. Let ζ_8 denotes an eighth root of unity. The zeta function of the order 8 Weil restriction of $E(\mathbb{F}_p)$ is

$$\begin{aligned} Z(T, \mathbb{F}_p) &= \left((T-\zeta_8\alpha)(T-\zeta_8^7\overline{\alpha})(T-\zeta_8^3\alpha)(T-\zeta_8^5\overline{\alpha})\right)\left((T-\zeta_8^5\alpha)(T-\zeta_8^3\overline{\alpha})(T-\zeta_8^7\alpha)(T-\zeta_8\overline{\alpha})\right) \\ &= (T^4 - 2yT^3 + 2y^2T^2 - 2ypT + p^2)(T^4 + 2yT^3 + 2y^2T^2 + 2ypT + p^2) \end{aligned}$$

and $\#J_{\mathcal{C}}(\mathbb{F}_p) = p^2 + 1 - 2yp + 2y^2 - 2y = (p-y)^2 + (y-1)^2$. We recognize the order of $J_{\mathcal{C}_5}(\mathbb{F}_p)$ when the considered isogeny is defined over $\mathbb{F}_{p^4}$ (and with n and y swapped). Hence it is the construction detailed above in Alg. 2 with $D = 2$.

5.3 Order-12 Weil Restriction When D = 3

Let $E(\mathbb{F}_p)$ an elliptic curve defined over a prime field $\mathbb{F}_p$, of trace t_p and satisfying $(t_p)^2 - 4p = -3y^2$ (i.e. $D = 3$). The two roots of its zeta function over $\mathbb{C}$ are $\alpha = (t_p + i\sqrt{3}y)/2$ and $\overline{\alpha}$. Let ζ_{12} denotes a twelfth root of unity. The zeta function of the order 12 Weil restriction of $E(\mathbb{F}_p)$ is

$$\begin{aligned} Z(T, \mathbb{F}_p) &= \left((T-\zeta_{12}\alpha)(T-\zeta_{12}^{11}\overline{\alpha})(T-\zeta_{12}^{7}\alpha)(T-\zeta_{12}^{5}\overline{\alpha})\right)\left((T-\zeta_{12}^{5}\alpha)(T-\zeta_{12}^{7}\overline{\alpha})(T-\zeta_{12}^{11}\alpha)(T-\zeta_{12}\overline{\alpha})\right) \\ &= \left(T^4 - \left(-p + t_p\tfrac{t_p+3y}{2}\right)T^2 + p^2\right)\left(T^4 - \left(-p + t_p\tfrac{t_p-3y}{2}\right)T^2 + p^2\right) \end{aligned}$$

which can be interpreted as the zeta functions of two Jacobians of hyperelliptic curves defined over $\mathbb{F}_p$ of order $p^2+p+1-t_p(t_p \pm 3y)/2$. For further simplifications, we can also write $\#J_{\mathcal{C}}(\mathbb{F}_p) = (p-1)^2 + ((t_p - 3y)/2)^2 = (p+1)^2 - 3((t_p + y)/2)^2$.

To apply the Cocks-Pinch or Brezing-Weng method, we use

$$t_p \equiv -\omega(\omega p - 1)/i \mod r,\ y \equiv -\omega(\omega p + 1)/\sqrt{3} \mod r$$

with ω a third root of unity and i a fourth root of unity. We found new families with $\rho = 3$ (with Brezing-Weng method). It would be interesting to know if these quite special curves provide more features such as compression due to twists of higher degree.

6 Conclusion

We provided *explicit* formulas for the zeta function of the Jacobian of genus 2 hyperelliptic curves of the form $Y^2 = X^5 + aX^3 + bX$ and $Y^2 = X^6 + aX^3 + b$ (with $a, b \in \mathbb{F}_q^*$). We also presented several algorithms to obtain pairing-friendly hyperelliptic families. The constructions require to run the CM method to find a j-invariant in $\mathbb{F}_{p^2}$. We explained the differences with a j-invariant in $\mathbb{F}_p$ and gave references to fill the gap. There are some special issues for $D = 1, 3$: a ρ-value of 2 can be achieved but the Jacobian is unfortunately not simple. However, it is possible to construct suitable curves with $D = 1$ and $D = 3$ that achieve ρ-value around 3 using Weil restriction of order 8 or 12. It is worth noting that it is also possible to adapt the Dupont-Enge-Morain technique [10] to our setting but unfortunately it provides curves with $\rho \simeq 4$. It remains open to construct pairing-friendly hyperelliptic curves with $1 \leqslant \rho < 2$.

Acknowledgments. This work was supported in part by the French ANR-09-VERS-016 BEST Project and by the Commission of the European Communities through the ICT program under contract ICT-2007-216676 ECRYPT II. Aurore Guillevic is grateful to Olivier Orcière for his help in early computations and hints on how to run the CM method. We would thank the reviewers of the Pairing Conference for their precise proofreading and their constructive and thorough remarks.

References

1. Atkin, A.O.L., Morain, F.: Elliptic curves and primality proving. Math. Comput. 61, 29–68 (1993)
2. Balakrishnan, J., Belding, J., Chisholm, S., Eisenträger, K., Stange, K., Teske, E.: Pairings on hyperelliptic curves. In: WIN - Women in Numbers: Research Directions in Number Theory. Fields Institute Communications, vol. 60, pp. 87–120. Amer. Math. Soc., Providence (2011)
3. Benger, N., Charlemagne, M., Freeman, D.M.: On the Security of Pairing-Friendly Abelian Varieties over Non-prime Fields. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 52–65. Springer, Heidelberg (2009)
4. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptology 17(4), 297–319 (2004)
6. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. 24(3-4), 235–265 (1997); Computational algebra and number theory, London (1993)
7. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Des. Codes Cryptography 37(1), 133–141 (2005)
8. Certivox. MIRACL Crypto SDK (2012), http://certivox.com/index.php/solutions/miracl-crypto-sdk/
9. Cocks, C., Pinch, R.G.: ID-based cryptosystems based on the Weil pairing (2001) (unpublished manuscript)
10. Dupont, R., Enge, A., Morain, F.: Building curves with arbitrary small mov degree over finite prime fields. J. Cryptology 18(2), 79–89 (2005)
11. Enge, A.: CM Software (February 2012), http://www.multiprecision.org/index.php?prog=cm
12. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptology 23(2), 224–280 (2010)
13. Freeman, D., Stevenhagen, P., Streng, M.: Abelian Varieties with Prescribed Embedding Degree. In: van der Poorten, A.J., Stein, A. (eds.) ANTS-VIII 2008. LNCS, vol. 5011, pp. 60–73. Springer, Heidelberg (2008)
14. Freeman, D.M., Satoh, T.: Constructing pairing-friendly hyperelliptic curves using weil restriction. J. Number Theory 131(5), 959–983 (2011)
15. Furukawa, E., Kawazoe, M., Takahashi, T.: Counting Points for Hyperelliptic Curves of Type $y^2 = x^5 + ax$ over Finite Prime Fields. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 26–41. Springer, Heidelberg (2004)
16. Galbraith, S.D.: Pairings. In: Blake, I.F., Seroussi, G., Smart, N.P. (eds.) Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series, vol. 317, ch. 9. Cambridge Univ. Press (2004)
17. Galbraith, S.D., Hess, F., Vercauteren, F.: Hyperelliptic Pairings. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 108–131. Springer, Heidelberg (2007)
18. Galbraith, S.D., Pujolas, J., Ritzenthaler, C., Smith, B.: Distortion maps for supersingular genus two curves. J. Math. Crypt. 3(1), 1–18 (2009)
19. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190–200. Springer, Heidelberg (2001)

20. Gaudry, P.: Fast genus 2 arithmetic based on theta functions. J. Math. Crypt. 1(3), 243–265 (2007)
21. Gaudry, P., Kohel, D., Smith, B.: Counting Points on Genus 2 Curves with Real Multiplication. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 504–519. Springer, Heidelberg (2011)
22. Gaudry, P., Schost, É.: On the Invariants of the Quotients of the Jacobian of a Curve of Genus 2. In: Bozta, S., Sphparlinski, I. (eds.) AAECC 2001. LNCS, vol. 2227, pp. 373–386. Springer, Heidelberg (2001)
23. Gaudry, P., Schost, É.: Genus 2 point counting over prime fields. J. Symb. Comput. 47(4), 368–400 (2012)
24. Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 126–135. Springer, Heidelberg (2008)
25. Kachisa, E.J.: Generating More Kawazoe-Takahashi Genus 2 Pairing-Friendly Hyperelliptic Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 312–326. Springer, Heidelberg (2010)
26. Kawazoe, M., Takahashi, T.: Pairing-Friendly Hyperelliptic Curves with Ordinary Jacobians of Type $y^2 = x^5 + ax$. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 164–177. Springer, Heidelberg (2008)
27. Koblitz, N.: Elliptic curve cryptosystems. Math. Comp. 48(177), 203–209 (1987)
28. Koblitz, N.: Hyperelliptic cryptosystems. J. Cryptology 1, 139–150 (1989)
29. Konstantinou, E., Kontogeorgis, A., Stamatiou, Y., Zaroliagis, C.: On the efficient generation of prime-order elliptic curves. J. Cryptology 23, 477–503 (2010)
30. Konstantinou, E., Stamatiou, Y., Zaroliagis, C.: Efficient generation of secure elliptic curves. International Journal of Information Security 6, 47–63 (2007)
31. Lercier, R.: Algorithmique des courbes elliptiques dans les corps finis. PhD thesis, École Polytechnique (1997)
32. Lercier, R., Lubicz, D., Vercauteren, F.: Point counting on elliptic and hyperelliptic curves. In: Avanzi, R.M., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., Vercauteren, F. (eds.) Handbook of Elliptic and Hyperelliptic Curve Cryptography. Discrete Mathematics and its Applications, vol. 34, ch. 17, pp. 239–263. CRC Press, Boca Raton (2005)
33. Miller, V.S.: Use of Elliptic Curves in Cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
34. Satoh, T.: On p-adic Point Counting Algorithms for Elliptic Curves over Finite Fields. In: Fieker, C., Kohel, D.R. (eds.) ANTS 2002. LNCS, vol. 2369, pp. 43–66. Springer, Heidelberg (2002)
35. Satoh, T.: Generating Genus Two Hyperelliptic Curves over Large Characteristic Finite Fields. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 536–553. Springer, Heidelberg (2009)
36. Schoof, R.: Elliptic curves over finite fields and the computation of square roots mod p. Math. Comput. 44, 483–494 (1998)
37. Scott, M.: MIRACL library (August 2011), http://www.shamus.ie
38. Takashima, K.: A new type of fast endomorphisms on jacobians of hyperelliptic curves and their cryptographic application. IEICE Transactions 89-A(1), 124–133 (2006)

Tate Pairing Computation on Jacobi's Elliptic Curves*

Sylvain Duquesne[1] and Emmanuel Fouotsa[2]

[1] IRMAR, UMR CNRS 6625, Université Rennes 1,
Campus de Beaulieu 35042 Rennes cedex, France
sylvain.duquesne@univ-rennes1.fr
[2] Département de Mathématiques, Université de Yaoundé 1
Faculté des Sciences, BP 812 Yaoundé, Cameroun
emmanuel.fouotsa@prmais.org

Abstract. We propose for the first time the computation of the Tate pairing on Jacobi intersection curves. For this, we use the geometric interpretation of the group law and the quadratic twist of Jacobi intersection curves to obtain a doubling step formula which is efficient but not competitive compared to the case of Weierstrass curves, Edwards curves and Jacobi quartic curves. As a second contribution, we improve the doubling and addition steps in Miller's algorithm to compute the Tate pairing on the special Jacobi quartic elliptic curve $Y^2 = dX^4 + Z^4$. We use the birational equivalence between Jacobi quartic curves and Weierstrass curves together with a specific point representation to obtain the best result to date among all the curves with quartic twists. In particular for the doubling step in Miller's algorithm, we obtain a theoretical gain between 6% and 21%, depending on the embedding degree and the extension field arithmetic, with respect to Weierstrass curves [6] and Jacobi quartic curves [23].

Keywords: Jacobi quartic curves, Jacobi intersection curves, Tate pairing, Miller function, group law, geometric interpretation, birational equivalence.

1 Introduction

While first used to solve the discrete logarithm problem on elliptic curve group [20,12], bilinear pairings are now useful to construct many public key protocols for which no other efficient implementation is known [18,3]. A survey of some of these protocols can be found in [9]. The efficient computation of pairings depends on the model chosen for the curve. Pairing computation on the Edwards model of elliptic curves have been done successively in [7], [17] and [1]. The recent results on pairing computation using elliptic curves of Weierstrass form can be found in [5,6]. Recently in [23] Wang et al. have computed the Tate pairing on

* This work was supported in part by French ANR projects no. 07-BLAN-0248 "ALGOL" and 09-BLAN-0020-01 "CHIC".

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 254–269, 2013.

Jacobi quartic elliptic curves using the geometric interpretation of the group law. In this paper, we focus on Jacobi intersection curves and the special Jacobi quartic elliptic curves $Y^2 = dX^4 + Z^4$ over the field of large characteristic p not congruent to 3 modulo 4.

We use the geometric interpretation of the group law of Jacobi intersection curves to obtain the first explicit formulas for the Miller function in Tate pairing computation in this case. For pairing computation with even embedding degree, we define and use the quadratic twist of this curve. This allows the Miller doubling stage to be slightly more efficient than when using Weierstrass curves, Edwards curves and Jacobi quartic curves. Moreover, for pairing computation with embedding degree divisible by 4, we define and use the quartic twist of the curve $Y^2 = dX^4 + Z^4$. Our result is an improvement of the result obtained by Wang et al. in [23] and is to our knowledge the best result to date on pairing computation among all curves with quartic twists.

The rest of this paper is organized as follows: Section 2 gives a background on the two forms of Jacobi elliptic curves mentioned above, including background on pairings that we will use in the remainder of the paper. In Section 3, we first look for Miller functions on Jacobi intersection curves using the geometric interpretation of the group law and then compute the Tate pairing on this curve. Section 4 presents the computation of the Tate pairing on the Jacobi quartic curve mentioned above using birational equivalence. Finally, we conclude in Section 5.

2 Background on Pairings and on Jacobi's Elliptic Curves

In this section we briefly review pairings on elliptic curves, Jacobi intersection curves and the Jacobi quartic curves. We also define twists of Jacobi's curves.

2.1 The Tate Pairing

In this section E is an elliptic curve defined over a finite field $\mathbb{F}_q$. The neutral element is denoted O. Let r be a large prime divisor of the group order $\sharp E(\mathbb{F}_q)$ and k the embedding degree of E with respect to r, i.e the smallest integer such that r divides $q^k - 1$. Consider a point $P \in E(\mathbb{F}_q)[r]$ and the function $f_{r,P}$ with divisor $\mathrm{Div}(f_{r,P}) = r(P) - r(O)$. Let $Q \in E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$ and μ_r be the group of r-th roots of unity in $\mathbb{F}_{q^k}^*$. The reduced Tate pairing e_r is defined as

$$e_r(P, Q) = f_{r,P}(Q)^{\frac{q^k-1}{r}} \in \mu_r.$$

If one knows the function $h_{R,S}$ such that $\mathrm{Div}(h_{R,S}) = (R) + (S) - (S + R) - (O)$ where R and S are two points of E, then the Tate pairing can be computed in an iterative way by Miller's algorithm [22] in Algorithm 1. This algorithm computes in the i-th iteration the evaluation at a point Q of the function $f_{i,P}$ having divisor $\mathrm{Div}(f_{i,P}) = i(P) - ([i]P) - (i-1)(O)$, called Miller's function.

Algorithm 1. Miller Algorithm

Input : $P \in E(\mathbb{F}_q)[r]$, $Q \in E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$
$r = (r_{n-1}, r_{n-2}, ...r_1, r_0)_2$ with $r_{n-1} = 1$.

Output: The Tate pairing of P and Q : $f_{r,P}(Q)^{\frac{q^k-1}{r}}$

1.Set $f \leftarrow 1$ and $R \leftarrow P$
2.For $i = n-2$ down to 0
Set $f \leftarrow f^2 \cdot h_{R,R}(Q)$ and $R \leftarrow 2R$
if $r_i = 1$ then $f \leftarrow f \cdot h_{R,P}(Q)$ and $R \leftarrow R + P$
3. $f \leftarrow f^{\frac{q^k-1}{r}}$

After $n-1$ iterations, the evaluation at Q of the function f having divisor $r(P) - r(O)$ is obtained. More informations on pairings can be found in [13] and [8].

Notation 1. *The following notations will be permanently used in this work.*
m, s : cost of multiplication and squaring in the base field $\mathbb{F}_q$
m_c: cost of the multiplication by the constant c in $\mathbb{F}_q$
M, S: cost of multiplication and squaring in the extension field $\mathbb{F}_{q^k}$

2.2 The Jacobi Intersection Curves

A Jacobi intersection form elliptic curve over $\mathbb{F}_q$ is defined by

$$E_a : \begin{cases} x^2 + y^2 = 1 \\ ax^2 + z^2 = 1 \end{cases} \text{ where } a \text{ belongs to } \mathbb{F}_q \text{ and } a(a-1) \neq 0.$$

The Jacobi intersection curve E_a is isomorphic to an elliptic curve on the Weierstrass form $y^2 = x(x-1)(x-a)$. The affine version of the unified addition formulas is given in [4] by $(x_3, y_3, z_3) = (x_1, y_1, z_1) + (x_2, y_2, z_2)$ such that :

$$x_3 = \frac{x_1y_2z_2 + y_1z_1x_2}{y_2^2 + z_1^2x_2^2}, y_3 = \frac{y_1y_2 - x_1z_1x_2z_2}{y_2^2 + z_1^2x_2^2}, z_3 = \frac{z_1z_2 - ax_1y_1x_2y_2}{y_2^2 + z_1^2x_2^2}$$

See [4,10] for further results on Jacobi intersection curves. An affine point (x, y, z) on a Jacobi intersection curves is represented by the projective homogeneous coordinates (X : Y : Z : T) satisfying

$$\begin{cases} X^2 + Y^2 = T^2 \\ aX^2 + Z^2 = T^2 \end{cases}$$

and $(x, y, z) = (X/T, Y/T, Z/T)$ with $T \neq 0$. The negative of $(X : Y : Z : T)$ is $(-X : Y : Z : T)$. The neutral element $P_0 = (0, 1, 1)$ is represented by $(0 : 1 : 1 : 1)$. By setting $T = 0$ we get four points at infinity: $\Omega_1 = (1 : s : t : 0)$, $\Omega_2 = (1 : s : -t : 0)$, $\Omega_3 = (1 : -s : t : 0)$ and $\Omega_4 = (1 : -s : -t : 0)$ where $1 + s^2 = 0$ and $a + t^2 = 0$.

Group Law on Jacobi Intersection Curves. The first formulas for addition law on points of Jacobi intersection curves given by Chudnovsky and Chudnovsky in [4] used projective homogeneous coordinates. In [15], Hisil et al. improved these formulas by representing points as a sextuplet $(X : Y : Z : T : XY : ZT)$ as follows:

The sum of the points represented by $(X_1 : Y_1 : Z_1 : T_1 : U_1 : V_1)$ and $(X_2 : Y_2 : Z_2 : T_2 : U_2 : V_2)$ where $U_1 = X_1Y_1$; $V_1 = Z_1T_1$ and $U_2 = X_2Y_2$; $V_2 = Z_2T_2$ is the point $(X_3 : Y_3 : Z_3 : T_3 : U_3 : V_3)$ such that:

$$\begin{aligned} X_3 &= X_1T_1Y_2Z_2 + Y_1Z_1X_2T_2, \\ Y_3 &= Y_1T_1Y_2T_2 - X_1Z_1X_2Z_2, \\ Z_3 &= Z_1T_1Z_2T_2 - aX_1Y_1X_2Y_2, \\ T_3 &= T_1^2Y_2^2 + Z_1^2X_2^2, \\ U_3 &= X_3Y_3, \\ V_3 &= Z_3T_3. \end{aligned}$$

with the algorithm:
$E \leftarrow X_1Z_2; F \leftarrow Y_1T_2; G \leftarrow Z_1X_2; H \leftarrow T_1Y_2; J \leftarrow U_1V_2; K \leftarrow V_1U_2;$
$X_3 \leftarrow (H + F)(E + G) - J - K; Y_3 \leftarrow (H + E)(F - G) - J + K;$
$Z_3 \leftarrow (V_1 - aU_1)(U_2 + V_2) + aJ - K; T_3 \leftarrow (H+G)^2 - 2K; U_3 \leftarrow X_3Y_3; V_3 \leftarrow Z_3T_3.$
This point addition costs $11m + 1s + 2m_a$.

The doubling of the point represented by $(X_1 : Y_1 : Z_1 : T_1 : U_1 : V_1)$ is the point $(X_3 : Y_3 : Z_3 : T_3 : U_3 : V_3)$ such that:

$$\begin{aligned} X_3 &= 2X_1Y_1Z_1T_1, \\ Y_3 &= -Z_1^2T_1^2 - aX_1^2Y_1^2 + 2(X_1^2Y_1^2 + Y_1^4), \\ Z_3 &= Z_1^2T_1^2 - aX_1^2Y_1^2, \\ T_3 &= Z_1^2T_1^2 + aX_1^2Y_1^2, \\ U_3 &= X_3Y_3, \\ V_3 &= Z_3T_3. \end{aligned}$$

with the algorithm: $E \leftarrow V_1^2; F \leftarrow U_1^2; G \leftarrow aF; T_3 \leftarrow E + G; Z_3 \leftarrow E - G; Y_3 \leftarrow 2(F + Y_1^4) - T_3; X_3 \leftarrow (U_1 + V_1)^2 - E - F; U_3 \leftarrow X_3Y_3; V_3 \leftarrow Z_3T_3.$
This point doubling costs $2m + 5s + 1m_a$.

2.3 The Jacobi Quartic Curve

A Jacobi quartic elliptic curve over a finite field $\mathbb{F}_q$ is defined by $E_d : y^2 = dx^4 + 2\delta x^2 + 1$ with discriminant $\triangle = 256d(\delta^2 - d)^2 \neq 0$. In [2] Billet and Joye proved that if $E : y^2 = x^3 + ax + b$ has a point of order 2 denoted $(\theta, 0)$ then E is birationally equivalent to the Jacobi quartic:

$$Y^2 = dX^4 - 2\delta X^2Z^2 + Z^4$$

where $d = -(3\theta^2 + 4a)/16$ and $\delta = 3\theta/4$. In the remainder of this paper, we will focus our interest on the special Jacobi quartic curve $E_d : Y^2 = dX^4 + Z^4$

because this curve has interesting properties such as quartic twist which contribute to an efficient computation of pairing.
The affine model of this curve is $y^2 = dx^4 + 1$ with $(x, y) = (\frac{X}{Z}, \frac{Y}{Z^2})$. The special Jacobi quartic curve E_d is birationally equivalent to the Weierstrass curve $E : y^2 = x^3 - 4dx$ using the maps

$$\varphi \begin{cases} (0:1:1) \longmapsto O \\ (0:-1:1) \longmapsto (0,0) \\ (X:Y:Z) \longmapsto \left(2\frac{(Y+Z^2)}{X^2}, 4\frac{Z(Y+Z^2)}{X^3}\right) \end{cases} ; \varphi^{-1} \begin{cases} (0,0) \longmapsto (0:-1:1) \\ (x,y) \longmapsto (2x:2x^3-y^2:y) \\ O \longmapsto (0:1:1) \end{cases}$$

Group Law on the Curve $Y^2 = dX^4 + Z^4$. Here we specialize formulas for point doubling and point addition on the curve E_d from the formulas on the affine model given in [16].

The point addition $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$ on the affine model of E_d is given by:

$$x_3 = \frac{x_1^2 - x_2^2}{x_1 y_2 - y_1 x_2} \text{ and } y_3 = \frac{(x_1 - x_2)^2}{(x_1 y_2 - y_1 x_2)^2}(y_1 y_2 + 1 + d x_1^2 x_2^2) - 1.$$

By replacing x_1 by $\frac{X_1}{Z_1}$, x_2 by $\frac{X_2}{Z_2}$, y_1 by $\frac{Y_1}{Z_1^2}$, y_2 by $\frac{Y_2}{Z_2^2}$, $x_3 = \frac{X_3}{Z_3}$ and y_3 by $\frac{Y_3}{Z_3^2}$ a simple calculation yields to

$$X_3 = X_1^2 Z_2^2 - Z_1^2 X_2^2,\ Z_3 = X_1 Z_1 Y_2 - X_2 Z_2 Y_1,$$
$$Y_3 = (X_1 Z_2 - X_2 Z_1)^2 (Y_1 Y_2 + (Z_1 Z_2)^2 + d(X_1 X_2)^2) - Z_3^2.$$

The point doubling $(x_3, y_3) = 2(x_1, y_1)$ on the affine model of E_d is given by :

$$x_3 = \frac{2y_1}{2 - y_1^2} x_1 \text{ and } y_3 = \frac{2y_1}{2 - y_1^2}\left(\frac{2y_1}{2 - y_1^2} - y_1\right) - 1.$$

By replacing x_1 by $\frac{X_1}{Z_1}$, y_1 by $\frac{Y_1}{Z_1^2}$, x_3 by $\frac{X_3}{Z_3}$ and y_3 by $\frac{Y_3}{Z_3^2}$, a simple calculation yields to:

$$X_3 = 2X_1 Y_1 Z_1,$$
$$Z_3 = Z_1^4 - dX_1^4,$$
$$Y_3 = 2Y_1^4 - Z_3^2.$$

2.4 Twists of Jacobi Curves

A twist of an elliptic curve E defined over a finite field $\mathbb{F}_q$ is an elliptic curve E' over $\mathbb{F}_q$ that is isomorphic to E over an algebraic closure of $\mathbb{F}_q$. The smallest integer t such that E and E' are isomorphic over $\mathbb{F}_{q^t}$ is called the degree of the twist. The points input of a pairing on a curve of embedding degree k take the form $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$. However many authors have shown that one can use the twist of a curve to take the input $Q \in E'(\mathbb{F}_{q^{k/t}})$ where operations can be performed more efficiently [11].

Let $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_q$ be an elliptic curve in Weierstrass form. The equation defining the twist E' has the form $y^2 = x^3 + a\omega^4 x + b\omega^6$ where $\omega \in \mathbb{F}_{q^k}$ and the isomorphism between E' and E is

$$\begin{array}{rccc}\psi: & E' & \longrightarrow & E \\ & (x', y') & \longmapsto & (x'/\omega^2, y'/\omega^3).\end{array}$$

Some details on twists can be found in [6].

Quadratic Twist of Jacobi Intersection Curves

Definition 1. *Let the Jacobi intersection curve E_a defined as in Subsection 2.2. A quadratic ($t = 2$) twist of E_a over the extension $\mathbb{F}_{q^{k/2}}$ of $\mathbb{F}_q$ (k even) is the curve*

$$\begin{cases} \delta^2 x^2 + y^2 = 1 \\ a\delta^2 x^2 + z^2 = 1 \end{cases}$$

Where $\{1, \delta\}$ is the basis of $\mathbb{F}_{q^k}$ as a $\mathbb{F}_{q^{k/2}}$-vector space and $\delta^2 \in \mathbb{F}_{q^{k/2}}$.

Proposition 1. *Let $E_{a,\delta}$ over $\mathbb{F}_{q^{k/2}}$ be a quadratic twist of E_a. The $\mathbb{F}_{q^k}$ isomorphism between $E_{a,\delta}$ and E_a is given by*

$$\begin{array}{rccc}\psi: & E_{a,\delta} & \to & E_a \\ & (x, y, z) & \mapsto & (\delta x, y, z)\end{array}$$

Twist of Jacobi Quartic Curves. To obtain the twist of the Jacobi quartic curve defined by $Y^2 = dX^4 + Z^4$, we use the birational maps defined in Subsection 2.3 and the twist of Weierstrass curves defined at the beginning of this subsection.

Definition 2. *A quartic twist of the Jacobi quartic curve $Y^2 = dX^4 + Z^4$ over the extension $\mathbb{F}_{q^{k/4}}$ of $\mathbb{F}_q$ is the curve*

$$E_{d,\omega} : Y^2 = d\omega^4 X^4 + Z^4$$

where $\omega \in \mathbb{F}_{q^k}$ is such that $\omega^2 \in \mathbb{F}_{q^{k/2}}$, $\omega^3 \in \mathbb{F}_{q^k} \backslash \mathbb{F}_{q^{k/2}}$ and $\omega^4 \in \mathbb{F}_{q^{k/4}}$. That is $\{1, \omega, \omega^2, \omega^3\}$ is a basis of $\mathbb{F}_{q^k}$ as a vector space over $\mathbb{F}_{q^{k/4}}$.

Proposition 2. *Let $E_{d,\omega}$ over $\mathbb{F}_{q^{k/4}}$ be a twist of E_d. The $\mathbb{F}_{q^k}$ isomorphism between $E_{a,\delta}$ and E_a is given by*

$$\begin{array}{rccc}\psi: & E_{d,\omega} & \to & E_d \\ & (X : Y : Z) & \mapsto & \left(\frac{X}{\omega^2} : \frac{Y}{\omega^6} : \frac{Z}{\omega^3}\right)\end{array}$$

3 Pairing on Jacobi Intersection Curves

3.1 Geometric Interpretation of the Group Law

The aim of this section is to find the function $h_{R,S}$. For this, we give more details on the geometric interpretation in [21] of the group law of Jacobi intersection curves. We consider $P_0 = (0, 1, 1)$ the $\mathbb{F}_q$-rational point on the curve which shall be the identity. Three points P_1, P_2, P_3 of the curve will sum to zero if and only if the four points P_0, P_1, P_2, P_3 are coplanar. The negation of a point $-P_1$ is

given as the residual intersection of the plane through P_1 containing the tangent line to the curve at P_0.

Let $f_{P_1,P_2}(x,y,z) = 0$ be the equation of the plane defined by the points P_1, P_2 and P_0. If $P_1 = P_2$ take f_{P_1,P_1} to be the tangent plane to the curve at P_1 passing through P_0. This plane intersects E_a at $R = -(P_1 + P_2) = -P_3$. Then $\text{Div}(f_{P_1,P_2}) = (P_1)+(P_2)+(R)+(P_0)-(\Omega)$ where $\Omega = (\Omega_1)+(\Omega_2)+(\Omega_3)+(\Omega_4)$ is a rational divisor.

Let $g_R(x,y,z) = 0$ be the equation of the plane passing through R and containing the tangent line to the curve at P_0. This plane intersects the curve E_a at the point $-R$. Then $\text{Div}(g_R) = (R) + 2(P_0) + (-R) - (\Omega)$
Define

$$h_{P_1,P_2} = \frac{f_{P_1,P_2}}{g_R}$$

then

$$\text{Div } (h_{P_1,P_2}) = (P_1) + (P_2) - (P_1 + P_2) - (P_0)$$

Theorem 1. *The functions f_{P_1,P_2} and g_R are defined as follows :*

$$f_{P_1,P_2}(x,y,z) = \alpha x + \beta(y-1) + \gamma(z-1)$$

with:

$$\alpha = \begin{cases} (z_2-1)(y_1-1) - (y_2-1)(z_1-1) & \text{if } P_1 \neq P_2, \\ x_1(-a(y_1-1) + z_1 - 1) & \text{if } P_1 = P_2. \end{cases}$$

$$\beta = \begin{cases} x_2(z_1-1) - x_1(z_2-1) & \text{if } P_1 \neq P_2, \\ y_1(z_1-1) & \text{if } P_1 = P_2. \end{cases}$$

$$\gamma = \begin{cases} x_1(y_2-1) - x_2(y_1-1) & \text{if } P_1 \neq P_2, \\ -z_1(y_1-1) & \text{if } P_1 = P_2. \end{cases}$$

and

$$g_{P_3}(x,y,z) = (z_3-1)(y-1) + (1-y_3)(z-1).$$

Proof 1.
1. Let $f_{P_1,P_2}(x,y,z) = \alpha x+\beta y+\gamma z+\theta = 0$ be the equation of the plane. Because $P_0 = (0,1,1)$ belongs to this plane we have $\theta = -\beta - \gamma$. Thus $f_{P_1,P_2}(x,y,z) = \alpha x + \beta y + \gamma z - \beta - \gamma = 0$.
If P_1 and P_2 are different then by evaluating the previous equation at the points P_1 and P_2 we obtain two linear equations in α, β and γ :

$$\alpha x_1 + \beta(y_1-1) + \gamma(z_1-1) = 0$$

$$\alpha x_2 + \beta(y_2-1) + \gamma(z_2-1) = 0$$

with the solutions

$$\alpha = \begin{vmatrix} y_1-1 & z_1-1 \\ y_2-1 & z_2-1 \end{vmatrix}, \ \beta = \begin{vmatrix} z_1-1 & x_1 \\ z_2-1 & x_2 \end{vmatrix}, \ \gamma = \begin{vmatrix} x_1 & y_1-1 \\ x_2 & y_2-1 \end{vmatrix}$$

If $P_1 = P_2 \neq P_0$ then the tangent line to the curve at P_1 is collinear to the vector $(y_1z_1, -x_1z_1, -ax_1y_1) = (x_1, y_1, 0) \wedge (ax_1, 0, z_1)$. Thus one can take $\overrightarrow{n} = x_1(-a(y_1-1)+z_1-1), y_1(z_1-1), -z_1(y_1-1)) = (\alpha, \beta, \gamma)$ as a normal vector to the plane.

2. Assume that $g_R(x,y,z) = ax + by + cz + d = 0$. The tangent line to the curve at P_0 is the intersection of the planes $v = 1$ and $w = 1$. Thus P_0 and one arbitrary point $(1,1,1)$ on the line belong to the plane. This implies that $a = 0$ and $b = -c - d$ such that $g_R(x,y,z) = c(-y+z) + d(-y+1) = 0$. Because $R = (u,v,w)$ belongs to the plane, we have $c = d(-v+1)/(v-w)$ and by replacing this value of c in $g_R(x,y,z) = c(-y+z)+d(-y+1) = 0$ we obtain the desired result.

3.2 Miller Function on Jacobi Intersection Curves

In this section we show how to use the geometric interpretation of the group law to compute pairings. We assume that k is even. Let $(x_Q, y_Q, z_Q) \in E_{a,\delta}(\mathbb{F}_{q^{k/2}})$. Twisting (x_Q, y_Q, z_Q) with δ ensures that the second argument of the pairing is on $E_a(\mathbb{F}_{q^k})$ and is of the form $Q = (\delta x_Q, y_Q, z_Q)$, where x_Q, y_Q and z_Q are in $\mathbb{F}_{q^{k/2}}$.

Addition. By Theorem 1,

$$h_{P_1,P_2}(\delta x_Q, y_Q, z_Q) = \frac{\alpha x_Q\delta+\beta(y_Q-1)+\gamma(z_Q-1)}{(z_3-1)y_Q+(1-y_3)z_Q+(y_3-z_3)} = \frac{z_Q-1}{(z_3-1)y_Q+(1-y_3)z_Q+(y_3-z_3)}\left(\alpha\frac{x_Q}{z_Q-1}\delta+\beta\frac{y_Q-1}{z_Q-1}+\gamma\right)$$

To obtain the expression of this function in projective coordinates X, Y, Z and T, we set $x_i = \frac{X_i}{T_i}$, $y_i = \frac{Y_i}{T_i}$ and $z_i = \frac{Z_i}{T_i}$; i=1, 2, 3. The point Q can be maintained in affine coordinates ($T_Q = 1$). The function becomes:

$$h_{P_1,P_2}(\delta x_Q, y_Q, z_Q) = \frac{T_3(z_Q-1)\left(\alpha'\frac{x_Q}{z_Q-1}\delta+\beta'\frac{y_Q-1}{z_Q-1}+\gamma'\right)}{T_1T_2[(Z_3-T_3)y_Q+(T_3-Y_3)z_Q+(Y_3-Z_3)]} = \frac{T_3(z_Q-1)}{T_1T_2[(Z_3-T_3)y_Q+(T_3-Y_3)z_Q+(Y_3-Z_3)]}(\alpha'M_1\delta+\beta'N_1+\gamma')$$

where $\alpha' = (Z_2-T_2)(Y_1-T_1)-(Y_2-T_2)(Z_1-T_1)$, $\beta' = X_2(Z_1-T_1)-X_1(Z_2-T_2)$, $\gamma' = X_1(Y_2-Z_2)-X_2(Y_1-T_1)$ and $M_1 = \frac{x_Q}{z_Q-1}$, $N_1 = \frac{y_Q-1}{z_Q-1}$.

we can easily see that $\frac{T_3(z_Q-1)}{T_1T_2[(Z_3-T_3)y_Q+(T_3-Y_3)z_Q+(Y_3-Z_3)]} \in \mathbb{F}_{q^{k/2}}$ so it can be discarded in pairing computation since the final output of Miller loop is raised to the power $(q^k-1)/r$ and $q^{k/2}-1$ is a factor of $(q^k-1)/r$ since k is even. Thus we only have to evaluate

$$(\alpha'M_1)\delta+\beta'N_1+\gamma'$$

Since $Q = (\delta x_Q, y_Q, z_Q)$ is fixed during pairing computation, the quantities $M_1 = \frac{x_Q}{z_Q-1}$, $N_1 = \frac{y_Q-1}{z_Q-1}$ can be precomputed in $\mathbb{F}_{q^{k/2}}$. Each of the multiplication of α' by $M_1 \in \mathbb{F}_{q^{k/2}}$ and β' by $N_1 \in \mathbb{F}_{q^{k/2}}$ costs $\frac{k}{2}m$. Computing the coefficients α' , β' and γ' requires $6m$ and the point addition in Subsection 2.2 requires $11m + 1s + 2c$. Thus the point addition and Miller value computation require a

total of $1M + (k+17)m + 1s + 2m_a$. The point P_2 is not changed during pairing computation and can be given in affine coordinates i.e. $T_2 = 1$. Applying such a mixed addition reduces the cost to $1M + (k+16)m + 1s + 2m_a$.

Doubling. By Theorem 1,

$$\begin{aligned} h_{P_1,P_1}(\delta x_Q, y_Q, z_Q) &= \frac{x_1(-a(y_1-1)+z_1-1)x_Q\delta + y_1(z_1-1)(y_Q-1) - z_1(y_1-1)(z_Q-1)}{(z_3-1)y_Q+(1-y_3)z_Q+(y_3-z_3)} \\ &= \frac{x_1(-a(y_1-1)+z_1-1)x_Q\delta + y_1(z_1-1)(y_Q-1) - z_1(y_1-1)(z_Q-1)}{(z_3-1)y_Q+(1-y_3)z_Q+(y_3-z_3)} \\ &= \frac{(z_Q-1)(x_1(-a(y_1-1)+z_1-1))\frac{x_Q}{z_Q-1}\delta + y_1(z_1-1)\frac{y_Q}{z_Q-1} - z_1(y_1-1)}{(z_3-1)y_Q+(1-y_3)z_Q+(y_3-z_3)} \end{aligned}.$$

In projective coordinates the function becomes:

$$\begin{aligned} h_{P_1,P_1}(\delta x_Q, y_Q, z_Q) &= \frac{T_3(z_Q-1)\left(\alpha_1'\frac{x_Q}{z_Q-1}\delta + \beta_1'\frac{y_Q}{z_Q-1} - \gamma_1'\right)}{T_1^3[(Z_3-T_3)y_Q+(T_3-Y_3)z_Q+(Y_3-Z_3)]} \\ &= \frac{T_3(z_Q-1)}{T_1^3[(Z_3-T_3)y_Q+(T_3-Y_3)z_Q+(Y_3-Z_3)]}\left(\alpha_1' M_2\delta + \beta_1' N_2 - \gamma_1'\right) \end{aligned}.$$

Where $M_2 = 2a\frac{x_Q}{z_Q-1}$ and $N_2 = a\frac{y_Q}{z_Q-1}$. $\alpha_1' = X_1(-a(Y_1-T_1)+Z_1-T_1)$; $\beta_1' = Y_1(Z_1-T_1)$; $\gamma_1' = Z_1(Y_1-T_1)$.

We can also verify that $\frac{T_3(z_Q-1)}{T_1^3[(Z_3-T_3)y_Q+(T_3-Y_3)z_Q+(Y_3-Z_3)]} \in \mathbb{F}_{q^{k/2}}$ such that it can be discarded thanks to the final exponentiation. Thus we only have to evaluate

$$(\alpha_1' M_2)\delta + \beta_1' N_2 - \gamma_1'$$

Again the quantities $M_2 = 2a\frac{x_Q}{z_Q-1}$ and $N_2 = a\frac{y_Q}{z_Q-1}$ are precomputed in $\mathbb{F}_{q^{k/2}}$. Note that each of the multiplications $\alpha_1' M_2$ and $\beta_1' N_2$ costs $\frac{k}{2}m$. Computing α_1',β_1' and γ_1' requires $3m$ and the point doubling from Subsection 2.2 requires $2m + 5s + 1m_a$. Thus the point doubling and Miller value computation require a total of $1M + 1S + (k+5)m + 5s + 1m_a$.

3.3 Comparison of Results

The comparison of results is given in Table 1. These comparisons are made for the Tate pairing and curves with a quadratic twist.

Table 1. Comparisons of our pairing formulas with the previous fastest formulas

Curves	Doubling	Mixed Addition
Weierstrass(a=0) [6]	$1M + 1S + (k+2)m + 7s + 1m_b$	$1M + (k+10)m + 2s$
Twisted Edwards [1]	$1M + 1S + (k+6)m + 5s + 2m_a$	$1M + (k+12)m + 1m_a$
Jacobi quartic [23]	$1M + 1S + (k+4)m + 8s + 1m_a$	$1M + (k+16)m + 1s + 4m_{a,d}$
This work	$1M + 1S + (k+5)m + 5s + 1m_a$	$1M + (k+16)m + 1s + 2m_a$

4 Tate Pairing Computation on $E_d : Y^2 = dX^4 + Z^4$

Wang et al. in [23] considered pairings on Jacobi quartics and gave the geometric interpretation of the group law. We use a different way, namely birational equivalence between Jacobi quartic curves and Weierstrass curves, of obtaining the

formulas. We specialize to the particular curves $E_d : Y^2 = dX^4 + Z^4$ to obtain better results for these up to 26% improvement compared to the result in [23]. To derive the Miller function $H(X, Y, Z)$ for E_d, we first write the Miller function $h(x, y)$ on the Weierstrass curve E. Then by using the birational equivalence we have $H(X, Y, Z) = h(\varphi(X, Y, Z))$.

4.1 The Miller Function

The Jacobi quartic curve $E_d : Y^2 = dX^4 + Z^4$ is birationally equivalent to the Weierstrass curve $E : y^2 = x^3 - 4dx$. Given two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ such that $P_3(x_3, y_3) = P_1 + P_2$, then the Miller function $h(x, y)$ for this Weierstrass curve such that a relation $\mathrm{Div}(h) = (P_1) + (P_2) - (P_3) - (O)$ holds is given by:

$$h(x, y) = \frac{y - \lambda x - \alpha}{x - x_3}$$

Where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ if $P_1 \neq P_2$ and $\lambda = \frac{3x_1^2 - 4d}{2y_1}$ if $P_1 = P_2$ and $\alpha = y_1 - \lambda x_1$.
As explained at the beginning of this section, the Miller function for the Jacobi quartic $E_d : Y^2 = dX^4 + Z^4$ is given by $H(X, Y, Z) = h(\varphi(X, Y, Z))$. A simple calculation gives:

$$H(X, Y, Z) = \frac{4X_3^2 X^2}{2X_3^2(Y + Z^2) - 2X^2(Y_3 + Z_3^2)} \left(\frac{ZY + Z^3}{X^3} - \frac{1}{2}\lambda \left(\frac{Y + Z^2}{X^2} \right) - \frac{\alpha}{4} \right)$$

where

$$\lambda = \begin{cases} \frac{-2X_1^3 Z_2(Y_2 + Z_2^2) + 2X_2^3 Z_1(Y_1 + Z_1^2)}{X_1 X_2[-X_1^2(Y_2 + Z_2^2) + X_2^2(Y_1 + Z_1^2)]} & \text{if } P_1 \neq P_2, \\ \frac{Y_1 + 2Z_1^2}{X_1 Z_1} & \text{if } P_1 = P_2. \end{cases}$$

and

$$\alpha = \begin{cases} \frac{-4(Y_1 + Z_1^2)(Y_2 + Z_2^2)(Z_2 X_1 - Z_1 X_2)}{X_1 X_2[-X_1^2(Y_2 + Z_2^2) + X_2^2(Y_1 + Z_1^2)]} & \text{if } P_1 \neq P_2, \\ \frac{-2Y_1(Y_1 + Z_1^2)}{X_1^3 Z_1} & \text{if } P_1 = P_2. \end{cases}$$

Remark 1. It is simple to verify that our formula obtained by change of variables is exactly the same result obtained by Wang et al. in [23] using the geometric interpretation of the group law.

Indeed, by setting $x_1 = \frac{X_1}{Z_1}$, $x_2 = \frac{X_2}{Z_2}$, $y_1 = \frac{Y_1}{Z_1^2}$ and $y_2 = \frac{Y_2}{Z_2^2}$ in their Miller function obtained for the curve $E_{d,a} : y^2 = dx^4 + 2ax + 1$ (by taking $a = 0$), we get exactly the same result that we found above.
The correctness of the formulas in this work can be checked at
`http://www.prmais.org/Jacobi-Formulas.txt`.

4.2 Simplification of the Miller Function

By using twist technique as explained earlier, the point Q in the Tate pairing computation can be chosen to be $\left(\frac{X_Q}{\omega^2} : \frac{Y_Q}{\omega^6} : \frac{Z_Q}{\omega^3} \right)$ or $(x_Q\omega, y_Q, 1)$ in affine coordinates where X_Q, Y_Q, Z_Q, x_Q and y_Q are in $\mathbb{F}_{q^{k/4}}$. Thus

$$H(x_Q\omega, y_Q, 1) = \frac{2X_3^2x_Q^2\omega^2}{X_3^2(y_Q+1)-x_Q^2\omega^2(Y_3+Z_3^2)}\left(-\frac{1}{2}\lambda\left(\frac{y_Q+1}{x_Q^2\omega^4}\right)\omega^2 + \left(\frac{y_Q+1}{x_Q^3\omega^4}\right)\omega - \frac{\alpha}{4}\right).$$

Write $-\frac{\alpha}{4} = \frac{A}{D}$ and $-\frac{1}{2}\lambda = \frac{B}{D}$ then

$$H(x_Q\omega, y_Q, 1) = \frac{2X_3^2x_Q^2\omega^2D^{-1}}{X_3^2(y_Q+1)-x_Q^2\omega^2(Y_3+Z_3^2)}\left(B\left(\frac{y_Q+1}{x_Q^2\omega^4}\right)\omega^2 + D\left(\frac{y_Q+1}{x_Q^3\omega^4}\right)\omega + A\right)$$

We can easily see that $\frac{2X_3^2x_Q^2\omega^2}{D(X_3^2(y_Q+1)-x_Q^2\omega^2(Y_3+Z_3^2))} \in \mathbb{F}_{q^{k/2}}$ so it can be discarded in pairing computation thanks to the final exponentiation. Thus we only have to evaluate

$$H = B\left(\frac{y_Q+1}{x_Q^2\omega^4}\right)\omega^2 + D\left(\frac{y_Q+1}{x_Q^3\omega^4}\right)\omega + A$$

Since $Q = (x_Q\omega, y_Q, 1)$ is fixed during pairing computation, the quantities $\frac{y_Q+1}{x_Q^3\omega^4}$ and $\frac{y_Q+1}{x_Q^2\omega^4}$ can be precomputed in $\mathbb{F}_{q^{k/4}}$. Note that each of the multiplications $D\left(\frac{y_Q+1}{x_Q^3\omega^4}\right)$ and $B\left(\frac{y_Q+1}{x_Q^2\omega^4}\right)$ costs $\frac{k}{4}m$.

Remark 2. We can use the fact that in the expression of H the term ω^3 is absent and $A \in \mathbb{F}_q$. Thus in Miller's algorithm, the cost of the main multiplication in $\mathbb{F}_{q^k}$ is not $1M$ but $\left(\frac{1}{k}+\frac{1}{2}\right)M$ assuming that schoolbook multiplication is used. But if we are using pairing friendly fields the embedding degree will be of the form $k = 2^i3^j$. Then we follow [19] and the cost of a multiplication or a squaring in the field $\mathbb{F}_{q^k}$ is 3^i5^j multiplications or squaring in $\mathbb{F}_q$ using Karatsuba and (or) Toom-Cook multiplication method. In this case, in Miller's algorithm, the cost of the main multiplication in $\mathbb{F}_{q^k}$ is $\left(\frac{7\cdot3^{i-2}5^j+2^{i-2}3^j}{3^i5^j}\right)M$. In the next sections ε stands for $\frac{1}{k}+\frac{1}{2}$ or $\frac{7\cdot3^{i-2}5^j+2^{i-2}3^j}{3^i5^j}$. A summary of how to obtain these costs is given in appendix.

In the next sections, we will compute A, B and D. In the work of Hisil et al. [16], there are different formulas in affine version for scalar multiplication. They used one of them to improve points addition and point doubling. These improved formulas have been used by Wang et al. to compute pairings. But in our case we obtained our formulas from a different affine version. For efficiency the point is represented by $(X : Y : Z : X^2 : Z^2)$ with $Z \neq 0$. We present the first time that this representation is used when $d \neq 1$. Thus we will use the points $P_1 = (X_1 : Y_1 : Z_1 : U_1 : V_1)$ and $P_2 = (X_2 : Y_2 : Z_2 : U_2 : V_2)$ where $U_i = X_i^2$, $V_i = Z_i^2$, $i = 1, 2$.

Remark 3. Note that if X^2 and Z^2 are known then expressions of the form XZ can be computed using the formula $((X+Z)^2 - X^2 - Z^2)/2$. This allows the replacement of a multiplication by a squaring presuming a squaring and three additions are more efficient. The operations concerned with this remark are followed by $*$ in the Tables 2 and 3.

4.3 Point Addition and Miller Iteration

When $P_1 \neq P_2$ we have $A = (Y_1 + Z_1^2)(Y_2 + Z_2^2)(Z_1X_2 - Z_2X_1)$, $D = X_1X_2[-X_1^2(Y_2 + Z_2^2) + X_2^2(Y_1 + Z_1^2)]$ and $B = X_1^3Z_2(Y_2 + Z_2^2) - X_2^3Z_1(Y_1 + Z_1^2)$.

Using the algorithm in Table 2 the computation of A, B, D and the point addition can be done in $18m+5s+1m_d$ or $12m+11s+1m_d$ according to Remark 3. Applying mixed addition($Z_2 = 1$), this cost is reduced to $15m + 4s + 1m_d$ or $12m + 7s + 1m_d$. Thus the point addition and Miller value computation require a total of $\varepsilon M + 1S + \left(\frac{k}{2} + 15\right) m + 4s + 1m_d$ or $\varepsilon M + 1S \left(\frac{k}{2} + 12\right) m + 7s + 1m_d$.

Table 2. Combined formulas for addition and Miller value computation

Operations		*Values*
$U := Y_1 + V_1$		$U = Y_1 + Z_1^2$
$V := Y_2 + V_2$		$V = Y_2 + Z_2^2$
$R := Z_2X_1$	*	$R = Z_2X_1$
$S := Z_1X_2$	*	$S = Z_1X_2$
$A := S - R$		$A = Z_1X_2 - Z_2X_1$
$A := AV$		$A = (Y_2 + Z_2^2)(Z_1X_2 - Z_2X_1)$
$A := AU$		$A = (Y_1 + Z_1^2)(Y_2 + Z_2^2)(Z_1X_2 - Z_2X_1)$
$U := U_2U$		$U = X_2^2(Y_1 + Z_1^2)$
$V := U_1V$		$V = X_1^2(Y_2 + Z_2^2)$
$B := RV - SU$		$B = X_1^3Z_2(Y_2 + Z_2^2) - X_2^3Z_1(Y_1 + Z_1^2)$
$D := X_1X_2$	*	$D = X_1X_2$
$E := dD^2$		$E = d(X_1X_2)^2$
$D := D(U - V)$		$D = X_1X_2[-X_1^2(Y_2 + Z_2^2) + X_2^2(Y_1 + Z_1^2)]$
$X_3 := (R + S)(R - S)$		$X_3 = X_1^2Z_2^2 - Z_1^2X_2^2$
$W_1 := X_1Z_1$	*	$W_1 = X_1Z_1$
$W_2 := X_2Z_2$	*	$W_2 = X_2Z_2$
$Z_3 := W_1Y_2 - W_2Y_1$		$Z_3 = X_1Z_1Y_2 - X_2Z_2Y_1$
$U := Y_1Y_2$		$U = Y_1Y_2$
$V := Z_1Z_2$	*	$V = Z_1Z_2$
$V := V^2 + E$		$V = (Z_1Z_2)^2 + d(X_1X_2)^2$
$E := (R - S)^2$		$E = (X_1Z_2 - X_2Z_1)^2$
$U_3 := X_3^2$		$U_3 = X_3^2$
$V_3 := Z_3^2$		$V_3 = Z_3^2$
$Y_3 := E(U + V) - V_3$		$Y_3 = (X_1Z_2 - X_2Z_1)^2(Y_1Y_2 + (Z_1Z_2)^2 + d(X_1X_2)^2) - Z_3^2$

4.4 Point Doubling and Miller Iteration

When $P_1 = P_2$ we have $A = Y_1(Y_1 + Z_1^2)$, $D = 2X_1^3Z_1$ and $B = -X_1^2(Y_1 + 2Z_1^2)$. The computation of A, B, D and the point doubling can be done using the algorithm in Table 3 with $4m + 6s + 1m_d$ or $3m + 7s + 1m_d$ according to the Remark 3.

Table 3. Combined formulas for doubling and Miller value computation

Operations		*Values*
$U := U_1^2$		$U = X_1^4$
$V := V_1^2$		$V = Z_1^4$
$Z_3 := V - dU$		$Z_3 = Z_1^4 - dX_1^4$
$E := X_1 Z_1$	$*$	$E = X_1 Z_1$
$D := 2U_1 E$		$D = 2X_1^3 Z_1$
$A := (2Y_1 + V_1)^2/4 - U$		$A = Y_1(Y_1 + Z_1^2)$
$B := -U_1(Y_1 + 2V_1)$		$B = -X_1^2(Y_1 + 2Z_1^2)$
$X_3 := 2EY_1$		$X_3 = 2X_1 Y_1 Z_1$
$V_3 := Z_3^2$		$V_3 = Z_3^2$
$Y_3 := 2V - Z_3$		$Y_3 = dX_1^4 + Z_1^4 = Y_1^2$
$Y_3 := 2Y_3^2 - V_3$		$Y_3 = 2Y_1^4 - Z_3^2$
$U_3 := X_3^2$		$U_3 = X_3^2$

Thus the point doubling and Miller value computation require a total of $\varepsilon M + 1S + (\frac{k}{2} + 4)m + 6s + 1m_d$ or $\varepsilon M + 1S + (\frac{k}{2} + 3)m + 7s + 1m_d$.

4.5 Comparison

The comparison of results is summarized in Table 4 and Table 5. These comparisons are made for the Tate pairing and curves with a quartic twist. In Table 4 we assume that Schoolbook multiplication method is used whereas the comparisons in Table 5 are made using Karatsuba and Toom-Cook method for curves with $k = 2^i 3^j$. We also present an example of comparison in the cases $k = 8$ and $k = 16$ since these values are the most appropriate for cryptographic applications when a quartic twist is used.

Table 4. Comparison of our pairing formulas with the previous fastest formulas with an example using Schoolbook multiplication method

Curves	Doubling	Mixed Addition
Weierstrass(b=0) [6]	$1M + 1S + (\frac{k}{2} + 2)m + 8s + 1m_a$	$1M + (\frac{k}{2} + 9)m + 5s$
Jacobi quartic(a=0) [23]	$1M + 1S + (\frac{k}{2} + 5)m + 6s$	$1M + (\frac{k}{2} + 16)m + 1s + 1m_d$
This work	$(\frac{1}{k} + \frac{1}{2})M + 1S + (\frac{k}{2} + 3)m + 7s + 1m_d$	$(\frac{1}{k} + \frac{1}{2})M + (\frac{k}{2} + 12)m + 7s + 1m_d$
Example: $k = 8$		
Weierstrass(b=0) [6]	$98m + 16s + 1m_a$	$77m + 5s$
Jacobi quartic (a=0) [23]	$101m + 14s$	$84m + 1s + 1m_d$
This work	$75m + 15s + 1m_d$	$57m + 6s + 1m_d$

Remark 4. If we assume that $m = s = m_c$ and $k = 8$ then for the doubling step the total costs are $115m$, $115m$ and $91m$ for Weierstrass curve, Jacobi quartic curve (a=0) [23] and *this work* respectively. Hence we obtain in this work

a theoretical gain of 21% with respect to Weierstrass curves and Jacobi quartic curves. Similarly for the addition step we obtain a theoretical gain of 22% and 26% over Weierstrass and Jacobi quartic curves respectively. This theoretical gain increases together with the value of k.

Table 5. Comparison of our pairing formulas with the previous fastest formulas with an example on pairing friendly fields

Curves	Doubling	Mixed Addition
Weierstrass(b=0) [6]	$1M + 1S + (\frac{k}{2} + 2)m + 8s + 1m_a$	$1M + (\frac{k}{2} + 9)m + 5s$
Jacobi quartic(a=0) [23]	$1M + 1S + (\frac{k}{2} + 5)m + 6s$	$1M + (\frac{k}{2} + 16)m + 1s +$ $1m_d$
This work	$\left(\frac{7\cdot 3^{i-2}5^j + 2^{i-2}3^j}{3^i 5^j}\right) M + 1S +$ $(\frac{k}{2} + 3)m + 7s + 1m_d$	$\left(\frac{7\cdot 3^{i-2}5^j + 2^{i-2}3^j}{3^i 5^j}\right) M +$ $(\frac{k}{2} + 12)m + 7s + 1m_d$
Example 1: $k = 8$		
Weierstrass(b=0) [6]	$33m + 35s + 1m_a$	$40m + 5s$
Jacobi quartic (a=0) [23]	$36m + 33s$	$84m + 1s + 1m_d$
This work	$30m + 34s + 1m_d$	$39m + 7s + 1m_d$
Example 2: $k = 16$		
Weierstrass(b=0) [6]	$91m + 89s + 1m_a$	$98m + 5s$
Jacobi quartic (a=0) [23]	$94m + 87s$	$105m + 1s + 1m_d$
This work	$78m + 88s + 1m_d$	$87m + 7s + 1m_d$

Remark 5. We assume again that $m = s = m_c$. For $k = 8$ and for the doubling step we obtain a theoretical gain of 6% over Weierstrass curves and Jacobi quartic curves (a=0) [23]. This theoretical gain increases together with the value of k. When $k = 16$ the gain is 8% both for the addition and doubling step over Weierstrass curves. The improvement is 13% in addition step over Jacobi quartic curves.

Remark 6. The security and the efficiency of pairing-based systems requires using pairing-friendly curves. The Jacobi models of elliptic curves studied in this work are isomorphic to Weierstrass curves. Thus we can obtain pairing friendly curves of such models using the construction given by Galbraith et al. [14] or by Freeman et al. [11]. Some examples of pairing friendly curves of Jacobi quartic form can be found in [23].

5 Conclusion

In this work we have computed the Tate pairing on Jacobi intersection curves using the geometric interpretation of the group law. Our results show that the doubling step is efficient but not competitive compared to the results using other elliptic curves. The addition step may require further improvements. Furthermore we significantly improved the doubling and the addition step in Miller's algorithm to compute the Tate pairing on the special Jacobi quartic elliptic curve $E_d : Y^2 = dX^4 + Z^4$. Our result is the best to date among all the curves with a quartic twist.

Acknowledgements. The authors thank Nadia El Mrabet and Hongfeng Wu for helpful discussions. The authors also thank the anonymous referees and the program committee for their useful comments.

References

1. Arene, C., Lange, T., Naehrig, M., Ritzenthaler, C.: Faster computation of the Tate pairing. Journal of Number Theory 131(5), 842–857 (2011)
2. Billet, O., Joye, M.: The Jacobi Model of an Elliptic Curve and Side-Channel Analysis. In: Fossorier, M.P.C., Høholdt, T., Poli, A. (eds.) AAECC 2003. LNCS, vol. 2643, pp. 34–42. Springer, Heidelberg (2003)
3. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)
4. Chudnovsky, D.V., Chudnovky, G.V.: Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Advances in Applied Mathematics 7(2), 385–434 (1986)
5. Costello, C., Hisil, H., Boyd, C., Nieto, J.G., Wong, K.K.-H.: Faster Pairings on Special Weierstrass Curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 89–101. Springer, Heidelberg (2009)
6. Costello, C., Lange, T., Naehrig, M.: Faster Pairing Computations on Curves with High-Degree Twists. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 224–242. Springer, Heidelberg (2010)
7. Das, M.P.L., Sarkar, P.: Pairing Computation on Twisted Edwards Form Elliptic Curves. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 192–210. Springer, Heidelberg (2008)
8. Duquesne, S., Frey, G.: Background on pairings. In: Cohen, H., Frey, G. (eds.) Handbook of Elliptic and Hyperelliptic Curves Cryptography, pp. 115–124. Chapman and Hall/CRC (2005)
9. Dutta, R., Barua, R., Sarkar, P.: Pairing-based cryptography: A survey. Cryptology ePrint Archive, Report 2004/064 (2004)
10. Feng, R., Nie, M., Wu, H.: Twisted Jacobi Intersections Curves. In: Kratochvíl, J., Li, A., Fiala, J., Kolman, P. (eds.) TAMC 2010. LNCS, vol. 6108, pp. 199–210. Springer, Heidelberg (2010)
11. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology 23(2), 224–280 (2010)
12. Frey, G., Müller, M., Rück, H.: The Tate Pairing and the Discrete Logarithm applied to Elliptic Curve Cryptosystems. IEEE Transactions on Information Theory 45(5), 1717–1719 (1999)
13. Galbraith, S.D.: Pairings. In: Seroussi, G., Blake, I., Smart, N. (eds.) Advances in Elliptic Curve Cryptography, pp. 193–213. Cambridge University Press (2005)
14. Galbraith, S.D., McKee, J.F., Valenca, P.C.: Ordinary abelian varieties having small embedding degree. Finite Fields Applications 13, 800–814 (2007)
15. Hisil, H., Wong, K.K., Carter, G., Dawson, E.: Faster group operations on elliptic curves. In: Australasian Information Security Conference (AISC), Wellington, New Zealand, vol. 98, pp. 7–19 (2009)
16. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Jacobi Quartic Curves Revisited. In: Boyd, C., Nieto, J.G. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 452–468. Springer, Heidelberg (2009)

17. Ionica, S., Joux, A.: Another Approach to Pairing Computation in Edwards Coordinates. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 400–413. Springer, Heidelberg (2008)
18. Joux, A.: A One-Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
19. Koblitz, N., Menezes, A.: Pairing-Based Cryptography at High Security Levels. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)
20. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory 39(5), 1639–1646 (1993)
21. Merriman, J.R., Siksek, S., Smart, N.P.: Explicit 4-descents on an elliptic curve. Acta Arithmetica 77, 385–404 (1996)
22. Miller, S.V.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17(4), 235–261 (2004)
23. Wang, H., Wang, K., Zhang, L., Li, B.: Pairing Computation on Elliptic Curves of Jacobi Quartic Form. Chinese Journal of Electronics 20(4), 655–661 (2011)

A Appendix: Cost of the Main Multiplication in Miller's Algorithm

The main multiplication in Miller's algorithm is of the form $f \cdot h$ where f and h are in $\mathbb{F}_{q^k}$. Since $\mathbb{F}_{q^k}$ is a $\mathbb{F}_{q^{k/4}}$-vector space with basis $\{1, \omega, \omega^2, \omega^3\}$, f and h can be written as : $f = f_0 + f_1\omega + f_2\omega^2 + f_3\omega^3$ and $h = h_0 + h_1\omega + h_2\omega^2 + h_3\omega^3$ with f_i and h_i in $\mathbb{F}_{q^{k/4}}$, $i = 0, 1, 2, 3$. However in our case $h_3 = 0$, $h_0 \in \mathbb{F}_q$ and $k = 2^i 3^j$.

Schoolbook Method: A full multiplication $f.h$ costs k^2 multiplications in the base field $\mathbb{F}_q$ using schoolbook method. But thanks to the particular form of h_0 and h_3, each of the multiplications $f_i \cdot h_0$ costs $\frac{k}{4}$ and each of the multiplications $f_i \cdot h_1$, $f_i \cdot h_2$ costs $\frac{k^2}{16}$, $i = 0, 1, 2, 3$. Then final cost of the product $f \cdot h$ in the base field $\mathbb{F}_q$ is $8\frac{k^2}{16} + 4\frac{k}{4} = \frac{k^2}{2} + k$. Finally the ratio of the cost in this case by the cost of the general multiplication is $\frac{\frac{k^2}{2}+k}{k^2} = \frac{1}{2} + \frac{1}{k}$.

Karatsuba Method: The computation of $f \cdot h$ is done by computing the three products: $u = (f_0 + f_1\omega)(h_0 + h_1\omega)$ which costs $2^{i-2}3^j + 2(3^{i-2}5^j)$, $v = f_2(h_2 + h_3\omega)$ which costs $2(3^{i-2}5^j)$ and $w = (f_0 + f_2 + (f_1 + f_3)\omega)(h_0 + h_2 + (h_1 + h_3)\omega)$ which costs $3(3^{i-2}5^j)$. The final cost is then $7 \cdot 3^{i-2}5^j + 2^{i-2}3^j$.

Group Signatures with Message-Dependent Opening

Yusuke Sakai[1,⋆], Keita Emura[2,⋆⋆], Goichiro Hanaoka[3], Yutaka Kawai[4,⋆⋆⋆], Takahiro Matsuda[3,†], and Kazumasa Omote[5]

[1] The University of Electro-Communications, Japan
yusuke.sakai@uec.ac.jp
[2] National Institute of Information and Communications Technology, Japan
[3] National Institute of Advanced Industrial Science and Technology, Japan
[4] Mitsubishi Electric, Japan
[5] Japan Advanced Institute of Science and Technology, Japan

Abstract. This paper introduces a new capability of the group signature, called *message-dependent opening*. It is intended to weaken the higher trust put on an opener, that is, no anonymity against an opener is provided by ordinary group signature. In a group signature system with message-dependent opening (GS-MDO), in addition to the opener, we set up the *admitter* which is not able to open any user's identity but *admits* the opener to open signatures by specifying messages whose signatures should be opened. For any signature whose corresponding message is not specified by the admitter, the opener cannot extract the signer's identity from it. In this paper, we present formal definitions and constructions of GS-MDO. Furthermore, we also show that GS-MDO implies identity-based encryption, and thus for designing a GS-MDO scheme, identity-based encryption is crucial. Actually, we propose a generic construction of GS-MDO from identity-based encryption and adaptive NIZK proofs, and its specific instantiation from the Groth-Sahai proof system by constructing a new (k-resilient) identity-based encryption scheme which is compatible to the Groth-Sahai proof.

1 Introduction

Group signature [20] is a kind of anonymous signatures, which allows members of a group to sign a message anonymously. Signatures are verified with a single group public key, but the verification process does not reveal the identity of the signer. In some exceptional case, a designated authority, called the opener, identifies the actual signer. However, ordinary group signature puts extremely strong

⋆ The first author is supported by a JSPS Fellowship for Young Scientists.
⋆⋆ This work was done when the second author was a postdoctoral researcher at Center for Highly Dependable Embedded Systems Technology, Japan Advanced Institute of Science and Technology (JAIST).
⋆⋆⋆ This work was done when the fourth author was a doctoral student in The University of Tokyo, Japan.
† The fifth author is supported by a JSPS Fellowship for Young Scientists.

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 270–294, 2013.

privilege on the opener, i.e., the opener can freely identify the originator of any signature of his choice. In other words, ordinary group signature schemes provide no assurance on privacy against the opener at all. For example, in anonymous auction (which will later be explained in more detail), the opener can extract all bidders' identities.

This paper investigates a way of decentralizing this strong power of the opener. To this end, we propose a new kind of group signatures, group signature with a message-dependent opening capability. It divides (or "decentralizes") the strong power of the opener by introducing another authority, called the *admitter*. In an exceptional case in which, for example, a signature on a problematic message is found, the admitter issues a *token* which corresponds to the message (not the whole signed message). By using this token, the opener extracts the signer's identity from the signature while without the token, he is not able to do so. For instance, if the admitter decides that a message "Mr. XXX is fool!" should not be publicized as a signed message by an anonymous group member, he issues a token on this message. Then, by using it, the opener can immediately open the signer's identity of any signature if it corresponds to the above message.

At a first glance, one may think that for achieving the above functionality, the popular thresholding technique (i.e. thresholding the opener into multiple less-trusted openers) would be already sufficient. However, this is not true. Namely, in our context, the token is generated based on *the message which the admitter chooses* but not the signature for such messages. Therefore, once a token under a message (which is chosen by the admitter) is issued, for all signatures of this message, the signer's identity can be immediately extracted by the opener without interacting with any other party. Consequently, for a message which has already been specified as problematic, the opener can non-interactively open the signer's identity, and furthermore, if the admitter considers that there is no need to specify further messages which should be opened anymore, then he can erase his memory for avoiding leaking his secret. Notice that even when the admitter erases his secret, the opener can still open the signer's identity of any signature provided that its corresponding message was specified by the admitter before.

Contributions. In this paper, we propose group signature with a new additional capability, called *group signature with message-dependent opening* (GS-MDO). In GS-MDO, as mentioned above, we introduce the *admitter* which issues tokens for specific messages, and by using these tokens, the opener can extract signers' identities from signatures only if their corresponding messages are those specific ones. Due to this functionality, we can flexibly restrict the ability of the opener without any complicated interactive procedure (e.g. threshold decryption).

We first give a security definition of GS-MDO. Our security definition is an extension of the Bellare-Micciancio-Warinschi model [7] which is considered as the basic security definition for group signatures in the static setting, and more specifically, our security model is a natural modification of this model according to the difference between the standard group signature and ours which introduces the functionality of the message-dependent opening. Next, we discuss technical

hurdles for constructing GS-MDO which satisfies the above security requirement. Especially, we show that *it is possible to derive identity-based encryption (IBE) from any GS-MDO scheme in a black-box manner* if the underlying GS-MDO is secure in the above sense. In other words, IBE is crucial for constructing GS-MDO, and thus, it is impossible to construct GS-MDO without using IBE as a building block. Then, based on this observation, we present a generic construction of GS-MDO from IBE and adaptive non-interactive zero-knowledge (NIZK) proofs. Notice that in our generic construction, simulation-soundness [39] for NIZK is not required while the generic construction of the (standard) group signature [7] requires this strong property. Lastly, we propose an efficient instantiation of GS-MDO by applying the Groth-Sahai proof [29] to our generic construction. For utilizing the Groth-Sahai proof in our generic construction, we see that an IBE scheme which is compatible to the Groth-Sahai proof (like "structure preserving signatures" [4]) is necessary since our generic construction requires IBE. Unfortunately, there is no known such primitive, and thus we also construct a new IBE scheme which satisfies this requirement. By using our new IBE together with the Groth-Sahai proof, a fairly practical GS-MDO can be constructed. Specifically, the size of a signature is approximately 16 kilobytes when 256-bit prime order group is used. However, we should also honestly mention that our IBE has only k-resilient security [30], and consequently, the resulting GS-MDO scheme inherits this restriction (i.e. the admitter can issues at most k tokens).

Applications. As mentioned before, a straightforward application of GS-MDO schemes is detecting the originator of inappropriate messages in an anonymous bulletin board system. We further discuss more other potential applications of message-dependent opening systems in the following.

The first application we discuss is *anonymous auction*. In this application, the bidders form the group of anonymous signers. Each bidder produces a group signature on his bidding price. To detect the winner(s), the admitter issues the token for opening signatures on the highest price. Then the opener is only able to open the signatures on the highest price.

The advantage (of the message-dependent opening approach) over the threshold approach becomes clear in this application. Suppose that there are *many* winners who all bid the highest price in a tie. In the threshold approach, an interaction will be needed for each winner, hence the total communication cost will be proportional to the number of winners. In contrast, if one takes the message-dependent opening approach, only a small communication from the admitter to the opener will be needed. The communication cost does not depend on the number of winners.

Another application in which the message-dependent opening capability is useful is *identity escrow*. Let us consider an automated parking garage [34], in which when a customer enters the garage, he generates a group signature on a

message which encodes the date when he enters the garage (say, the string "2012-02-20"). Suppose a case in which there is an accident (a person is murdered, for example) in the garage. In this case the opener will open the signatures on the date when the accident occurs, in order to identify who is there at that day.

In this application, the opener needs to open *many* signatures on the *same* message. If one adopts the threshold technique to decentralize the authority, a large amount of interactions is required to open all the signatures. The message-dependent opening capability removes interactions between authorities, that is, the admitter issues a token for the day, and the opener opens all the signatures without interaction.

Related Works. Since the first proposal of group signature by Chaum and van Heyst [20], many efficient constructions have been proposed, most of which are relying on the random oracle model [6,11,18,33,25,23,10]. Many initial schemes were based on the strong-RSA assumption. Group signature schemes based on assumptions of the discrete-logarithm type were achieved, to name a few, by Camenisch and Lysyanskaya [18] and by Boneh, Boyen, and Shacham [11]. The former scheme is based on the LRSW assumption, while the latter is based on the q-strong Diffie-Hellman assumption.

Except generic constructions from general NIZK techniques, group signature schemes without relying on the random oracles are only very recently achieved. Ateniese, Camenisch, Hohenberger, and de Medeiros first proposed a group signature scheme from interactive assumptions avoiding random oracles [5]. Following to this scheme, Groth proposed a group signature scheme which avoids random oracles and interactive assumptions [27], but the scheme has a very large signature size. Boyen and Waters proposed highly efficient constructions [14,15], although the security guarantee of their schemes are not very strong, i.e. they only achieve so-called CPA-anonymity. Groth proposed another group signature scheme [28], which is almost as efficient as the Boyen-Waters schemes and satisfies higher security guarantee of the Bellare-Shi-Zhang model [8].

As for decentralizing and distributing the power of the group manager, separability of a cryptographic protocol was introduced by Kilian and Petrank [34] in the context of identity escrow. Lately, this notion was refined and adopted to the context of group signature by Camenisch and Michels [19]. The separability notion demands that keys of several entities involved in the cryptographic primitive need to be generated independently each other. In their setting, the power of a group manager is separated into two authorities. The first authority is able to allow a new group member to join the group, but not able to identify the originator of a group signature, and the other authority is vice versa. More formal modeling of these separated authorities is put forward by Bellare, Shi, and Zhang [8] and Kiayias and Yung [32].

Traceable signature is an extended notion of group signature, introduced by Kiayias, Tsiounis, and Yung [31]. This primitive allows the group manager to specify a group member as "misbehaving". Once a member was specified by the manager, anyone becomes able to detect the signatures of the specified user without interacting with the manager. In this time signatures of other group

members continue to be anonymous. In our terminology, this primitive achieves somewhat "signer-dependent opening" capability, but no message-dependent opening is achieved. A contractual anonymity system [40] has been proposed based on group signatures with verifier-local revocation [13]. In this system, when a user breaks a contract, an accountability server revokes anonymity of the user and notices the identity of the user to the service provider (In the contractual anonymity system, a user is said to *break the contract* when the user sends a message specified by the contract policy of the service provider). Since this scheme uses the conventional open algorithm, this system also differs from message-dependent opening.

Paper Organization. The rest of the paper is structured as follows. Sect. 2 describes definitions and security notions of several building blocks briefly. Sect. 3 presents the notion of GS-MDO and its syntax and security definitions. Sect. 4 discusses difficulties behind constructing efficient GS-MDO schemes. Specifically we argue that use of IBE in a construction of GS-MDO is essential by showing a generic construction of IBE from GS-MDO. In Sect. 5 and 6, we propose a generic construction of GS-MDO and its fairly efficient instantiation.

2 Preliminaries

Signatures. A signature scheme consists of the following three algorithms: A key generation algorithm $\mathsf{SigKg}(1^\lambda)$ outputs a pair (vk, sk). A signing algorithm $\mathsf{Sign}_{sk}(M)$ generates a signature s for a message M. A verification algorithm $\mathsf{Verify}_{vk}(M, s)$ outputs $\top$ or $\bot$, which respectively indicate "accept" or "reject". As a correctness, for all $\lambda \in \mathbb{N}$, all pairs (vk, sk) in the range of $\mathsf{SigKg}(1^\lambda)$, and all messages M, it is required to be satisfied that $\Pr[\mathsf{Verify}_{vk}(M, \mathsf{Sign}_{sk}(M)) = \top] = 1$. A signature scheme is *existentially unforgeable under chosen-message attack (EUF-CMA)* if all PPT adversaries, given vk generated from $\mathsf{SigKg}(1^\lambda)$ and an access to a signing oracle, which gives the adversary a signature of his choice, have negligible probability of outputting a pair (M, s) where M was never queried to the signing oracle and $\mathsf{Verify}_{vk}(M, s) = \top$. A signature scheme is said to be *strongly unforgeable one-time signature* when no adversary, given vk and allowed to access to a signing oracle *only at most once*, can output a valid message-signature pair (M, s) (i.e. $\mathsf{Verify}_{vk}(M, s) = \top$) which is different from the message-signature pair obtained from the signing oracle.

Tag-Based Key Encapsulation Mechanism. A tag-based key encapsulation mechanism (tag-based KEM)[1] [36,35] consists of the following three algorithms: A key generation algorithm $\mathsf{TKg}(1^\lambda)$ outputs a pair (pk, dk). An encapsulation

[1] Tag-based encryption, an encryption analogue of tag-based KEM, is originally introduced as "encryption with labels" by Shoup and Gennaro [42]. Tag-based KEM is different from "tag-KEM", introduced by Abe, Gennaro, Kurosawa, and Shoup [3]. However, any CCA-secure tag-KEM scheme can be immediately converted to a tag-based KEM scheme which is sufficiently secure for our purpose.

algorithm $\mathsf{TEnc}_{pk}(t)$ outputs (C, K) where a ciphertext C for a tag t encapsulates a session key $K \in \mathcal{K}_{\mathrm{PKE}}$, where $\mathcal{K}_{\mathrm{PKE}}$ is the session key space associated with the scheme. A decapsulation algorithm $\mathsf{TDec}_{dk}(t, C)$ outputs a decapsulated session key K or a special symbol $\perp$ indicating an invalid ciphertext. A tag-based KEM is said to be *selective-tag weakly chosen-ciphertext secure* when no PPT adversary has non-negligible advantage in the following game: The adversary is given a security parameter 1^λ and output a target tag t^*, then the challenger gives a public key pk. After receiving the public key, the adversary, in an arbitrary order, issues decryption queries (t, C), to which the challenger responds with the decryption result of C under the tag t. Here the adversary is not allowed to issue queries with $t = t^*$. At some point the adversary requests a challenge. The challenger flips a fair coin b' and sends (C^*, K^*) where C^* is a ciphertext generated under the tag t^* and K^* is either the session key encapsulated in C^* when $b' = 0$ or a random session key when $b' = 1$. After receiving the challenge the adversary is again allowed to issue decryption queries. The same restriction for queries is applied as before. Finally the adversary outputs a bit b. The advantage of the adversary is defined by the probability that $b = b'$ minus $1/2$.

Identity-Based KEM and Its k-resilient Variant. A k-resilient identity-based KEM [30] consists of the following four algorithms: A setup algorithm $\mathsf{ISetup}(1^\lambda, 1^k)$ outputs a pair (par, mk). A key extraction algorithm $\mathsf{IExt}_{mk}(ID)$ outputs a user decapsulation key dk_{ID}. An encapsulation algorithm $\mathsf{IEnc}_{par}(ID)$ outputs (C, K) where a ciphertext C for an identity ID encapsulates a session key $K \in \mathcal{K}_{\mathrm{IBE}}$, where $\mathcal{K}_{\mathrm{IBE}}$ is the session key space associated with the scheme. A decapsulation algorithm $\mathsf{IDec}_{dk_{ID}}(C)$ outputs a decapsulated session key K or a special symbol $\perp$ indicating an invalid ciphertext. A k-resilient identity-based KEM is said to be *k-resilient* if no PPT adversary has non-negligible (in λ) advantage in the following game: The adversary first receives a public parameter par. After receiving the parameter the adversary, in an arbitrary order, issues extraction queries ID, to which the challenger responds with the user decapsulation key for the user ID. At some point the adversary requests a challenge with an identity ID^*. The challenger flips a fair coin b' and sends a pair (C^*, K^*) where C^* is a ciphertext for the user ID^* and K^* is either the session key encapsulated in C^* when $b' = 0$ or a random session key when $b' = 1$. The adversary is not allowed to request a challenge with an identity whose user decapsulation key is queried before. After receiving the challenge the adversary is again allowed to issue extraction queries. This time querying the user decapsulation key for ID^* is disallowed. The adversary is also restricted that the total number of queries before and after the challenge is at most k. Finally the adversary outputs a bit b. The advantage of the adversary is defined by the probability that $b = b'$ minus $1/2$. We also say that an identity-based KEM is *fully secure* when any PPT adversary has non-negligible advantage in the same game even when the number of extraction queries is unbounded.

Non-Interactive Zero-Knowledge Proofs. A non-interactive proof system for a polynomial-time computable relation R consists of three probabilistic algorithms K, P, and V. The common reference string generation algorithm K produces a common reference string Σ. The proof algorithm P takes a common reference string Σ, a theorem x, and a witness w, where $R(x,w) = \top$, and produces a proof π. The verification algorithm V takes as input (Σ, x, π), and outputs either $\top$ or $\bot$. We say that a non-interactive proof system (K,P,V) has perfect completeness, when we have $\Pr[\Sigma \leftarrow K(1^\lambda); (x,w) \leftarrow \mathcal{A}(\Sigma); \pi \leftarrow P(\Sigma,x,w) : V(\Sigma,x,\pi) = \top \vee R(x,w) = \bot] = 1$ for any adversary $\mathcal{A}$. We say that a non-interactive proof system (K,P,V) has perfect soundness, when we have $\Pr[\Sigma \leftarrow K(1^\lambda); (x,\pi) \leftarrow \mathcal{A}(\Sigma) : V(\Sigma,x,\pi) = \bot \vee x \in L] = 1$ for all adversary $\mathcal{A}$, where L denotes the set of all x that has at least one w such that $R(x,w) = \top$. We say that a non-interactive proof system (K,P,V) is zero-knowledge when there exists a pair of probabilistic algorithms (S_1, S_2) such that we have $\Pr[\Sigma \leftarrow K(1^\lambda); (x,w) \leftarrow \mathcal{A}(\Sigma); \pi \leftarrow P(\Sigma,x,w) : \mathcal{A}(\pi) = 1] - \Pr[(\Sigma,\tau) \leftarrow S_1(1^\lambda); (x,w) \leftarrow \mathcal{A}(\Sigma); \pi \leftarrow S_2(\Sigma,\tau,x) : \mathcal{A}(\pi) = 1]$ is negligible for all PPT adversaries $\mathcal{A}$ that do not output (x,w) with $R(x,w) = \bot$.

3 Group Signatures with Message-Dependent Opening

Firstly we give an explanation of the scenario in which group signature with message-dependent opening is used. As ordinary group signatures, a GS-MDO scheme allows group members to sign a message anonymously, that is, without revealing their identities but only showing that one of the group members actually signed. In exceptional cases, a designated third party, called the opener, can "open" exceptional signatures, to identify the originator of signatures. In contrast to ordinary group signature schemes, a GS-MDO scheme requires the opener to cooperate with another authority, called the admitter, to open signatures. The admitter issues a message-specific token, and the opener is able to open signature on some message *only when a token for the message is issued from the admitter*.

A formal model of this scenario is given by the following definition. A GS-MDO scheme consists of the following five algorithms:

GKg: This algorithm takes as an input $(1^\lambda, 1^n, 1^k)$ where λ is a security parameter, n is the number of group members, and k is the maximum number of message-specific tokens that can be issued, and returns a group public key gpk, a message specification key msk, an opening key ok, and n group signing keys $\{gsk_i\}_{i\in[n]}$.

GSig: This algorithm takes as inputs gpk, gsk_i, and a message M, and returns a group signature σ.

Td: This algorithm takes as inputs gpk, msk, and M, and returns the token t_M for M.

GVf: This algorithm takes as inputs gpk, σ, and M, and returns $\top$ or $\bot$.

Open: This algorithm takes as inputs gpk, ok, M, σ, and t_M, and returns $i \in \mathbb{N}$ or $\bot$.

As a correctness, it is required that for all λ, n, k and for all $(gpk, msk, ok, \{gsk_i\}_{i\in[n]})$ in the range of $\mathsf{GKg}(1^\lambda, 1^n, 1^k)$, $\mathsf{GVf}(gpk, M, \mathsf{GSig}(gpk, gsk_i, M)) = \top$ for all $M \in \{0,1\}^*$ and $i \in [n]$, and $\mathsf{Open}(gpk, ok, M, \mathsf{GSig}(gpk, gsk_i, M), \mathsf{Td}(gpk, msk, M)) = i$ for all $M \in \{0,1\}^*$ and $i \in [n]$.

As in ordinary group signature, we need to ensure anonymity and traceability. However, in contrast to ordinary group signature, we have to further ensure two types of anonymity. It is related to the original motivation of the introduction of the admitter. The introduction of the admitter is intended to strengthen signers' anonymity against the authorities as strong as possible. To capture this intention, we define the indistinguishability of the originator of the signature in the strong setting that the opening key is given to the adversary. As a counterpart of this, we also define the indistinguishability in the setting that the message-specification key is given to the adversary.

For traceability, we just use the same definition to the ordinary group signature, in which the authorities are entirely corrupted by the adversary, since even ordinary group signature schemes has ensured that traceability against entirely corrupted openers.

Opener Anonymity. Here we give a formal definition of anonymity against the opener, called *opener anonymity*. It is formalized as the indistinguishability of signatures of two different signers of the adversary's choice. In the indistinguishability game, the adversary is given the opening key, and is asked to distinguish signatures of two different signers of its own choice. Opener anonymity is defined by requiring that no adversary has non-negligible advantage in distinguishing signatures.

We again remark that contrary to ordinary group signatures, the adversary is allowed to have the opening key. This is intended for modeling "anonymity against the opener."

Definition 1. *A GS-MDO scheme has* opener anonymity *if the advantage of any PPT adversary $\mathcal{A}$ in the following game between a challenger and the adversary is negligible in the security parameter λ:*

Setup. *The challenger runs $\mathsf{GKg}(1^\lambda, 1^n, 1^k)$ to obtain $(gpk, ok, msk, \{gsk_i\}_{i\in[n]})$ and sends $(gpk, ok, \{gsk_i\}_{i\in[n]})$ to $\mathcal{A}$.*

Token Query (Phase I). *$\mathcal{A}$ adaptively issues token queries. For a token query for a message M, the challenger responds with t_M which is obtained by running $\mathsf{Td}(gpk, msk, M)$.*

Challenge. *At some point $\mathcal{A}$ requests a challenge for i_0, $i_1 \in [n]$ and a message M^*. The challenger chooses a random bit b, and responds with $\mathsf{GSig}(gpk, gsk_{i_b}, M^*)$. In this phase $\mathcal{A}$ is forbidden to submit M^* whose token is previously queried in Phase I.*

Token Query (Phase II). *$\mathcal{A}$ continues to query tokens. In this phase $\mathcal{A}$ is forbidden to query M^*, which is submitted in Challenge phase.*

Guess. *Finally $\mathcal{A}$ outputs a bit b'. The advantage of $\mathcal{A}$ is defined by the absolute difference between the probability that b' is equal to b and $1/2$.*

We also say that a GS-MDO scheme has opener anonymity with k-bounded tokens *if any PPT adversary $\mathcal{A}$ which issues at most k token queries in total has negligible advantage.*

Admitter Anonymity. We then give a definition of anonymity against the admitter, called *admitter anonymity*. It is formalized in a similar manner to opener anonymity. That is, admitter anonymity requires signatures of two different signers are indistinguishable even when the adversary is given the message-specification key. The formal definition is as follows:

Definition 2. *A GS-MDO scheme has* admitter anonymity *if the advantage of any PPT adversary $\mathcal{A}$ in the following game between a challenger and the adversary is negligible in the security parameter λ:*

Setup. *The challenger runs* $\mathsf{GKg}(1^\lambda, 1^n, 1^k)$ *to obtain* $(gpk, ok, msk, \{gsk_i\}_{i\in[n]})$ *and sends* $(gpk, msk, \{gsk_i\}_{i\in[n]})$ *to* $\mathcal{A}$.

Open Query (Phase I). $\mathcal{A}$ *adaptively issues open queries. For an open query for a message-signature pair* (M, σ)*, the challenger generates* t_M *by running* $\mathsf{Td}(gpk, msk, M)$ *and responds with* $\mathsf{Open}(gpk, ok, M, \sigma, t_M)$.

Challenge. *At some point* $\mathcal{A}$ *requests a challenge for* i_0, $i_1 \in [n]$ *and a message* M^**. The challenger chooses a random bit* b*, and responds with* $\sigma^* \leftarrow \mathsf{GSig}(gpk, gsk_{i_b}, M^*)$.

Open Query (Phase II). $\mathcal{A}$ *continues to submit open queries. In this phase* $\mathcal{A}$ *is forbidden to query* σ^**, which is same as the signature produced in Challenge phase.*

Guess. *Finally* $\mathcal{A}$ *outputs a bit* b'*. The advantage of* $\mathcal{A}$ *is defined by the absolute difference between the probability that* b' *is equal to* b *and* $1/2$.

Notice that the number of opening queries the adversary issues is unbounded (but of course polynomially many).

Traceability. The last notion is *traceability*, which requires that even if the opener and the admitter collude and they further adaptively corrupt some group members, the corrupted parties can produce neither forged signatures nor untraceable signatures. In contrast to the case of the anonymity notions, this case considers a collusion of two authorities.

Definition 3. *A GS-MDO scheme has* traceability *if the advantage of any PPT adversary $\mathcal{A}$ in the following game between a challenger and the adversary is negligible in the security parameter λ:*

Setup. *The challenger runs* $\mathsf{GKg}(1^\lambda, 1^n, 1^k)$ *to obtain* $(gpk, ok, msk, \{gsk_i\}_{i\in[n]})$ *and sends* (gpk, ok, msk) *to* $\mathcal{A}$.

Query. $\mathcal{A}$ *adaptively issues following two types of queries:*

1. *The first type of queries is* key revealing query*, in which $\mathcal{A}$ requests for revealing the group signing key of the group member i. For this type of queries the challenger responds with* gsk_i.

$\mathsf{ISetup}(1^\lambda)$:
$(gpk, ok, msk, \{gsk_1, gsk_2\}) \leftarrow \mathsf{GKg}(1^\lambda, 1^2)$;
$par \leftarrow (gpk, ok, gsk_1, gsk_2)$; $mk \leftarrow msk$;
Output (par, mk).

$\mathsf{IExt}_{mk}(ID)$:
$dk_{ID} \leftarrow \mathsf{Td}(gpk, mk, ID)$;
Output dk_{ID}.

$\mathsf{IEnc}_{par}(ID)$:
For $i \in \{1, \ldots, \lambda\}$:
  $K_i \leftarrow \{0,1\}$;
  $\sigma_i \leftarrow \mathsf{GSig}(gpk, gsk_{K_i+1}, ID)$;
$C \leftarrow (\sigma_1, \ldots, \sigma_\lambda)$;
$K \leftarrow K_1 \cdots K_\lambda$;
Output (C, K).

$\mathsf{IDec}_{dk_{ID}}(C)$:
Parse C as $(\sigma_1, \ldots, \sigma_\lambda)$;
For $i \in \{1, \ldots, \lambda\}$:
  $K_i \leftarrow \mathsf{Open}(gpk, ok, ID, \sigma_i, dk_{ID})$;
If $K_i = \perp$ for some i
  then Output $\perp$;
Else Output $K_1 \ldots K_\lambda$.

Fig. 1. The black-box construction of identity-based KEM from group signature with message-dependent opening

2. *The second type of queries is* signing query*, in which $\mathcal{A}$ requests for a signature on some message by some group member. For a query (i, M) of this type, the challenger responds with $\mathsf{GSig}(gpk, gsk_i, M)$.*

Forge. *Finally the challenger outputs a forgery (M^*, σ^*). $\mathcal{A}$ wins if $\mathsf{GVf}(gpk, M^*, \sigma^*) = \top$ and one of the following two conditions holds: (1) $\mathsf{Open}(gpk, ok, M^*, \sigma^*, \mathsf{Td}(gpk, msk, M^*)) = \perp$, or (2) $\mathsf{Open}(gpk, ok, M^*, \sigma^*, \mathsf{Td}(gpk, msk, M^*)) = i^* \neq \perp$, and neither a key revealing query for the user i^* nor a signing query for (i^*, M^*) is submitted. The advantage of $\mathcal{A}$ is defined by the probability that $\mathcal{A}$ wins.*

4 Difficulty in Having Efficient Constructions

In this section we discuss several difficulties in designing efficient GS-MDO schemes. We firstly investigate relationships between GS-MDO and other cryptographic primitives, and then we discuss the difficulty that lies in designing efficient constructions.

As for the relationship to other primitives, we show that the existence of a GS-MDO scheme implies that of an IBE scheme. In other words, we will present a black-box construction of IBE from any GS-MDO scheme. The same holds for the k-resilient versions.

The formal theorems are as follows:

Theorem 1. *If the underlying GS-MDO scheme satisfies opener anonymity, the identity-based KEM in Fig. 1 is fully secure.*

Theorem 2. *If the underlying GS-MDO scheme satisfies opener anonymity with k-bounded tokens, the identity-based KEM in Fig. 1 is k-resilient.*

Formal proofs can be given by a straightforward modification from the proof by Abdalla and Warinschi [1] or the similar technique used by Ohtake, Fujii, Hanaoka, and Ogawa [37], hence we omit detailed proofs.

We also note that Fig. 1 only shows a construction of identity-based *key encapsulation mechanism* rather than identity-based *encryption*. However, it suffices for the theorems since we can obtain a secure encryption scheme by combining the construction with an appropriate data encapsulation mechanism.

These theorems suggest that to use IBE is crucial for constructing a GS-MDO scheme. Considering the fact that a black-box construction of IBE from trapdoor permutation is impossible [12], we should conclude that it is almost unavoidable for a GS-MDO scheme to relying on an IBE scheme or its equivalence, not only on trapdoor permutation and NIZK proof. Otherwise one would construct an IBE scheme from surprisingly weaker primitives.

Another important aspect to establish an *efficient* GS-MDO scheme is realizing a "Groth-Sahai compatible" IBE scheme. This is because the only known construction of non-interactive zero-knowledge proof with reasonable efficiency is limited to the Groth-Sahai proof system. Also note that a non-interactive zero-knowledge proof system has been an important building block for almost all group signature schemes ever.

However, no currently known IBE scheme is Groth-Sahai compatible in the sense that the Groth-Sahai proof system cannot prove a kind of well-formedness of an IBE ciphertext in a zero-knowledge manner.

To overcome this gap, we adopt k-resilient IBE instead of fully secure IBE. In particular we design a k-resilient IBE scheme from the decision linear assumption by modifying the Heng-Kurosawa scheme [30] for this purpose (We also note that a similar construction can be obtained from a key-insulated encryption scheme by Dodis, Katz, Xu, and Yung [24]). The modification is needed since the original Heng-Kurosawa scheme is based on the decision Diffie-Hellman (DDH) assumption, which does not hold in groups with a bilinear map, and the Groth-Sahai proof system relies on a bilinear map in an essential way.

5 Generic Construction

In this section, we give a construction of a GS-MDO scheme. The construction is built on an EUF-CMA secure signature scheme, a strongly unforgeable one-time signature scheme, a selective-tag weakly chosen-ciphertext secure tag-based KEM, a k-resilient identity-based KEM, and an adaptive NIZK proof system.

At a first glance there are various building blocks. However, our generic construction is only relying on the existence of an IBE scheme and that of an NIZK proof system. Indeed signature schemes and a chosen-ciphertext secure tag-based encryption scheme can be constructed from a fully secure IBE.

The proposed construction shares an underlying idea with the generic construction by Bellare, Micciancio, and Warinschi (the BMW construction) [7] except the use of "simulation-sound" NIZK proofs. The proposed construction no longer relies on such a strong security requirement of simulation-soundness, which was exploited by the BMW construction [7]. Instead of the strong security requirement of simulation-soundness, we combine (ordinary) NIZK proofs with a strongly unforgeable one-time signature scheme. We remark that essentially same techniques have been used in a variety of contexts. To name a few,

Groth [28] used this technique for an efficient group signature scheme in a very similar manner. Camenisch, Chandran, and Shoup [17,16] used this to construct simulation-sound NIZK proofs, improving the result of Groth [27].

$\mathsf{GKg}(1^\lambda, 1^n, 1^k)$:
$(vk_{\mathrm{issue}}, sk_{\mathrm{issue}}) \leftarrow \mathsf{SigKg}(1^\lambda)$;
$(pk, dk) \leftarrow \mathsf{TKg}(1^\lambda)$;
$(par, mk) \leftarrow \mathsf{ISetup}(1^\lambda, 1^k)$;
$\Sigma \leftarrow K(1^\lambda)$;
$gpk \leftarrow (vk_{\mathrm{issue}}, pk, par, \Sigma)$;
$ok \leftarrow dk$;
$msk \leftarrow mk$;
For all $i \in [1, n]$:
  $(vk_i, sk_i) \leftarrow \mathsf{SigKg}(1^\lambda)$;
  $cert_i \leftarrow \mathsf{Sign}_{sk_{\mathrm{issue}}}(\langle i, sk_i \rangle)$;
  $gsk_i \leftarrow (i, vk_i, cert_i, sk_i)$;
Output $(gpk, ok, msk, \{gsk_i\}_i)$.

$\mathsf{GSig}(gpk, gsk_i, M)$:
Parse gpk as $(vk_{\mathrm{issue}}, pk, par, \Sigma)$;
Parse gsk_i as $(i, vk_i, cert_i, sk_i)$;
$s \leftarrow \mathsf{Sign}_{sk_i}(M)$;
$(vk_{\mathrm{OT}}, sk_{\mathrm{OT}}) \leftarrow \mathsf{SigKg}^{\mathrm{OT}}(1^\lambda)$;
$(C_{\mathrm{PKE}}, K_{\mathrm{PKE}}) \leftarrow \mathsf{TEnc}_{pk}(vk_{\mathrm{OT}})$;
$(C_{\mathrm{IBE}}, K_{\mathrm{IBE}}) \leftarrow \mathsf{IEnc}_{par}(M)$;
$\chi \leftarrow \langle i, vk_i, cert_i, s \rangle \odot K_{\mathrm{PKE}} \odot K_{\mathrm{IBE}}$;
$\pi \leftarrow P_{\mathrm{NIZK}}(\cdots)$;
$\sigma_{\mathrm{OT}} \leftarrow \mathsf{Sign}^{\mathrm{OT}}_{sk_{\mathrm{OT}}}(\langle C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi \rangle)$;
$\sigma \leftarrow (vk_{\mathrm{OT}}, C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi, \sigma_{\mathrm{OT}})$;
Output σ.

$\mathsf{GVf}(gpk, M, \sigma)$:
Parse gpk as $(vk_{\mathrm{issue}}, pk, par, \Sigma)$;
Parse σ as $(vk_{\mathrm{OT}}, C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi, \sigma_{\mathrm{OT}})$;
If $\mathsf{Verify}^{\mathrm{OT}}_{vk_{\mathrm{OT}}}(\langle C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi \rangle, \sigma_{\mathrm{OT}}) = 1$
and $V_{\mathrm{NIZK}}(\ldots) = 1$ then Output $\top$;
Else Output $\bot$.

$\mathsf{Td}(gpk, msk, M)$:
Parse gpk as $(vk_{\mathrm{issue}}, pk, par, \Sigma)$;
$t_M \leftarrow \mathsf{IExt}(par, msk, M)$;
Output t_M.

$\mathsf{Open}(gpk, ok, M, \sigma, t_M)$:
Parse gpk as $(vk_{\mathrm{issue}}, pk, par, \Sigma)$;
Parse σ as $(vk_{\mathrm{OT}}, C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi, \sigma_{\mathrm{OT}})$;
$K_{\mathrm{PKE}} \leftarrow \mathsf{TDec}_{ok}(vk_{\mathrm{OT}}, C_{\mathrm{PKE}})$;
$K_{\mathrm{IBE}} \leftarrow \mathsf{IDec}_{t_M}(M, C_{\mathrm{IBE}})$;
$\langle i, vk_i, cert_i, s \rangle \leftarrow \chi \odot K_{\mathrm{IBE}}^{-1} \odot K_{\mathrm{PKE}}^{-1}$;
If $\mathsf{Verify}^{\mathrm{OT}}_{vk_{\mathrm{OT}}}(\langle C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi \rangle, \sigma_{\mathrm{OT}}) = 1$
and $V_{\mathrm{NIZK}}(\ldots) = 1$
then Output i;
else Output $\bot$;

Fig. 2. The brief overview of the proposed GS-MDO scheme. The operator $\odot$ denotes some group operation. In the concrete instantiation, $\langle \cdots \rangle$ denotes a tuple consisting of all group elements that appear in the bracket, and the operator $\odot$ is the component-wise group multiplication. The non-interactive proof system $(K, P_{\mathrm{NIZK}}, V_{\mathrm{NIZK}})$ is for demonstrating the existence of a satisfying assignment of Eq. (1).

The Construction. In the construction, a group member has a key pair (vk_i, sk_i) of the signature scheme in which vk_i is authorized by another verification key vk_{issue} at the setup time. When a member makes a group signature, the member simply signs a message by sk_i. To be anonymous, the member further encrypts the signature together with the certificate (of the member), which authorizes the verification key vk_i, and attaches a non-interactive proof that demonstrates that a signature of an authorized member is encrypted. To encrypt a signature, the member uses a multiple encryption technique to ensure

neither the opener nor the admitter can reveal the identity as long as the admitter does not issue a token to the opener. The complete description of the scheme is shown in Fig. 2.

Let us explain the non-interactive proof that appears in the construction. The signature of the proposed scheme is of the form as $(vk_{\mathrm{OT}}, C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi, \sigma_{\mathrm{OT}})$, and, as mentioned above, the proof π demonstrates a valid signature of an authorized group member is encrypted into $(C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi)$ in a kind of a "multiple encryption" manner. In detail, the proof π proves that there exists a randomness r (for tag-based KEM), another randomness ρ (for identity-based KEM), a group member i, and the verification key vk_i, the certificate $cert_i$, and the signature s on a message M, such that

$$\begin{aligned}
&(C_{\mathrm{PKE}}, K_{\mathrm{PKE}}) = \mathsf{TEnc}_{pk}(vk_{\mathrm{OT}}; r),\\
&(C_{\mathrm{IBE}}, K_{\mathrm{IBE}}) = \mathsf{IEnc}_{par}(M; \rho),\\
&\chi = \langle i, vk_i, cert_i, s\rangle \odot K_{\mathrm{PKE}} \odot K_{\mathrm{IBE}},\\
&\mathsf{Verify}_{vk_{\mathrm{issue}}}(\langle i, vk_i\rangle, cert_i) = \top,\\
&\mathsf{Verify}_{vk_i}(M, s) = \top.
\end{aligned} \tag{1}$$

Technically speaking, we need several requirements on the session key spaces of tag-based KEM and k-resilient IBE. The requirements are: (i) The tag-based KEM scheme and the k-resilient IBE scheme share the same session key space $\mathcal{K}_{\mathrm{PKE}} = \mathcal{K}_{\mathrm{IBE}}$ and (ii) this session key space forms a finite group. These requirements are needed because we do a one-time pad to encrypt a signature of the group member. This group operation also needs to fall into the class of relations that the used non-interactive proof system can represent.

Finally, there are two encoding functions needed for completing the generic construction. The first is used to encode the identity of a group member and his verification key into the message space of the signature scheme when generating certificates of group members. The second one is used to encode $(i, vk_i, cert_i, s)$ into $\mathcal{K}_{\mathrm{PKE}}$, where i is the identity of a group member and vk_i, $cert_i$, and s are his verification key, certificate, and signature, respectively. It is used when issuing group signatures, especially encrypting his signature in order to hide his identity.

As below, the generic construction will have desirable security properties when all building blocks satisfy appropriate security properties.

Theorem 3. *The proposed scheme satisfies opener anonymity with k-bounded tokens if the identity-based KEM is k-resilient and the non-interactive proof system is zero-knowledge.*

Theorem 4. *The proposed scheme satisfies admitter anonymity when the tag-based KEM is selective-tag weakly chosen-ciphertext secure, the non-interactive proof system is zero-knowledge, and the one-time signature scheme is strongly unforgeable.*

Theorem 5. *The proposed scheme satisfies traceability when the non-interactive proof system is sound and the signature scheme is EUF-CMA secure.*

All the proofs of the theorems will appear in Appendix B.

6 Efficient Instantiation

Toward an efficient scheme, we will discuss how to instantiate the building blocks used in the generic constructions of the previous section.

As for the non-interactive proof, an obvious choice is the Groth-Sahai proof system, since there is no known fairly practical construction of a NIZK proof system except the Groth-Sahai proof system. However, to adopt the Groth-Sahai proof system, other building blocks are subjected to restrictions, due to the limitation of the type of theorems that the Groth-Sahai proof system can prove. In other words, other building blocks need to be *structure preserving* [2], and especially, the theorem should not involve elements of $\mathbb{G}_T$, where $\mathbb{G}_T$ is the target group of the underlying bilinear mapping. Hence, we have to choose an IBE scheme which fulfills this requirement as a building block, but unfortunately, there is no known such scheme. This means that *it is not straightforward to construct an efficient instantiation of our generic construction from the Groth-Sahai proof.*

In this section, we give an efficient instantiation by constructing a *structure preserving* IBE scheme and choosing other appropriate building blocks. However, we must also honestly mention that our IBE does not provide full security but only k-resilience [30]. It is also worth noting that constructing a structure-preserving IBE scheme is already an important open problem.

Our structure-preserving k-resilient IBE scheme is obtained by means of modifying the Heng-Kurosawa scheme [30] which is secure under the decision Diffie-Hellman (DDH) assumption in the sense of k-resilient security. Since the DDH assumption does not hold in a bilinear group, it is not possible to utilize it as it is, and thus, we construct a modified version of this scheme which is secure under the decision linear (DLIN) assumption.

6.1 k-Resilient IBE from the Decision Linear Assumption

As mentioned above, our proposed k-resilient IBE scheme can be obtained by applying several modifications to the original Heng-Kurosawa scheme [30, Sect. 3.2] which are as follows: (1) Basing on the DLIN assumption instead of the DDH assumption[2], (2) designing it as a key encapsulation mechanism instead of an encryption scheme, and (3) modifying it to encapsulate a sufficiently long session key in a constant size ciphertext (Indeed our proposed scheme encapsulates a session key of l group elements in a ciphertext of three group elements). Our proposed scheme is as follows:

[2] If we adopt the SXDH assumption, we can plug in the original Heng-Kurosawa scheme to the generic construction. However, in this case we need to set up two instances of the original Heng-Kurosawa scheme for two different groups $\mathbb{G}_1$ and $\mathbb{G}_2$, over which the bilinear map is defined. This is because the Abe-Haralambiev-Ohkubo signature scheme contains elements of both groups in its signature value. The same thing holds for the tag-based KEM.

Setup. Let $par = (u, v, h, \{D_{i,j} = u^{d_{i,j}} h^{d''_{i,j}}, \tilde{D}_{i,j} = v^{d'_{i,j}} h^{d''_{i,j}}\}_{i\in[l],j\in[k]})$ and $mk = (d_i(X), d'_i(X), d''_i(X))_{i\in[l]}$ where d_i, d'_i, and d''_i are the polynomials defined as follows: $d_i(X) = \sum_{j=0}^{k} d_{i,j}X^j$, $d'_i(X) = \sum_{j=0}^{k} d'_{i,j}X^j$, and $d''_i(X) = \sum_{j=0}^{k} d''_{i,j}X^j$ for all $i \in [l]$.

Key Extract. The decryption key for the user ID is derived as $dk_{ID} = \{d_i(ID), d'_i(ID), d''_i(ID)\}_{i\in[l]}$.

Encrypt. To encapsulate a session key, choose ρ and $\tilde{\rho}$ from $\mathbb{Z}_p$ randomly and compute $C_{\mathrm{IBE}} = (u^{\rho}, v^{\tilde{\rho}}, h^{\rho+\tilde{\rho}})$, which encapsulates the session key $K_{\mathrm{IBE}} = ((\prod_{j=0}^{k} D_{1,j}^{ID^j})^{\rho}(\prod_{j=0}^{k} \tilde{D}_{1,j}^{ID^j})^{\tilde{\rho}}, \ldots, (\prod_{j=0}^{k} D_{l,j}^{ID^j})^{\rho}(\prod_{j=0}^{k} \tilde{D}_{l,j}^{ID^j})^{\tilde{\rho}})$.

Decrypt. To decapsulate a session key from a ciphertext $C_{\mathrm{IBE}} = (C_1, C_2, C_3)$, compute $(C_1^{d_1(ID)} C_2^{d'_1(ID)} C_3^{d''_1(ID)}, \ldots, C_1^{d_l(ID)} C_2^{d'_l(ID)} C_3^{d''_l(ID)})$.

The security of this scheme is proved under the DLIN assumption, which says that given a tuple $(u, v, h, u^r, v^{\tilde{r}}, h^{\tilde{\tilde{r}}})$ it is hard to efficiently decide $r + \tilde{r} = \tilde{\tilde{r}}$ or not. Formal statements of the assumption and the theorem are as follows.

Definition 4. *We say that the* decision linear assumption *on* $\mathbb{G}$ *holds if for any polynomial-time algorithm* $\mathcal{D}$, $|\Pr[\mathcal{D}(u, v, h, u^r, v^{\tilde{r}}, h^{\tilde{\tilde{r}}}) \to 1 | r + \tilde{r} = \tilde{\tilde{r}}] - \Pr[\mathcal{D}(u, v, h, u^r, v^{\tilde{r}}, h^{\tilde{\tilde{r}}}) \to 1 | r + \tilde{r} \neq \tilde{\tilde{r}}]|$ *is negligible.*

Theorem 6. *The above construction is an adaptively secure* k*-resilient identity-based KEM if the decision linear assumption on* $\mathbb{G}$ *holds.*

Proof. Given an adversary $\mathcal{A}$ which attacks adaptive security against the above scheme, we bound its advantage by constructing the simulator below:

Setup. The simulator $\mathcal{B}$ receives an instance $(u, v, h, u^r, v^{\tilde{r}}, h^{\tilde{\tilde{r}}})$ of the decision linear problem, where $\tilde{\tilde{r}}$ is either $r + \tilde{r}$ or an independently random element of $\mathbb{Z}_p$. The simulator generates random polynomials $\{d_i(x) = d_{i,0} + \cdots + \alpha_{i,k}x^k, d'_i(x) = d'_{i,0} + \cdots + d'_{i,k}x^k, d''_i(x) = d''_{i,0} + \cdots + d''_{i,k}x^k\}_{i\in[l]}$ of degree k, sets $D_{i,j} \leftarrow u^{d_{i,j}} h^{d''_{i,j}}$ and $\tilde{D}_{i,j} \leftarrow v^{d'_{i,j}} h^{d''_{i,j}}$ for all $i \in [1, l]$ and $j \in [0, k]$, and runs $\mathcal{A}$ with input $par = (u, v, h, \{D_{i,j}, \tilde{D}_{i,j}\}_{i\in[1,l],j\in[0,k]})$.

Key Extraction Query (Phase I). When $\mathcal{A}$ queries an identity ID, $\mathcal{B}$ returns $dk_{ID} = \{d_i(ID), d'_i(ID), d''_i(ID)\}_{i\in[l]}$.

Challenge. When $\mathcal{A}$ requests a challenge for an identity ID^*, $\mathcal{B}$ computes $C^* = (u^r, v^{\tilde{r}}, h^{\tilde{\tilde{r}}})$ and $K^* = (K_1^*, \ldots, K_l^*) = ((u^r)^{d_1(ID^*)}(v^{\tilde{r}})^{d'_1(ID^*)}(h^{\tilde{\tilde{r}}})^{d''_1(ID^*)}, \ldots, (u^r)^{d_l(ID^*)}(v^{\tilde{r}})^{d'_l(ID^*)}(h^{\tilde{\tilde{r}}})^{d''_l(ID^*)})$. This C^* and K^* are given to $\mathcal{A}$ as a challenge.

Key Extraction Query (Phase II). Again, $\mathcal{A}$ may request a decryption key for ID and $\mathcal{B}$ responds as before.

Guess. Finally $\mathcal{A}$ outputs a bit b' and $\mathcal{B}$ outputs the same bit.

When $\tilde{\tilde{r}} = r + \tilde{r}$, a simple calculation shows that K^* is the real session key encapsulated in C^*. Otherwise when $\tilde{\tilde{r}} \neq r + \tilde{r}$, we will show that K^* distributes independently from all other values seen by $\mathcal{A}$. To see this, let ID_1, …, ID_k be the decapsulation key queries issued by $\mathcal{A}$ during the simulation, and observe

that queries reveal function values $d_i(ID_j)$, $d'_i(ID_j)$, $d''_i(ID_j)$ to $\mathcal{A}$, but $d_i(ID^*)$, $d'_i(ID^*)$, and $d''_i(ID^*)$ are not revealed. However, *par* further reveals the value $d_i(ID^*) + \alpha d''_i(ID^*)$ and $d'_i(ID^*) + \beta d''_i(ID_*)$, where $u = g^\alpha$ and $v = g^\beta$. The equations $\mathcal{A}$ can observe is represented as

$$\begin{pmatrix} \begin{pmatrix} dk_{ID_1} \end{pmatrix} \\ \vdots \\ \begin{pmatrix} dk_{ID_k} \end{pmatrix} \\ \log_g u^{x_i(ID^*)} h^{x''_i(ID^*)} \\ \log_g v^{x'_i(ID^*)} h^{x''_i(ID^*)} \\ \log_g K^*_i \end{pmatrix} = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & \ddots & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & \alpha & & 1 \\ & & & & & & & & \beta & 1 \\ & & & & & & & r\alpha & \tilde{r}\beta & \tilde{\tilde{r}} \end{pmatrix} \begin{pmatrix} d_i(ID_1) \\ d'_i(ID_1) \\ d''_i(ID_1) \\ \vdots \\ d_i(ID_k) \\ d'_i(ID_k) \\ d''_i(ID_k) \\ d_i(ID^*) \\ d'_i(ID^*) \\ d''_i(ID^*) \end{pmatrix},$$

where this matrix is non-singular, and hence K^* is uniformly distributed. All these facts justify the fact that the quantity

$$\Pr[\mathcal{A} \to 1 \mid K^* \text{ is real}] - \Pr[\mathcal{A} \to 1 \mid K^* \text{ is random}]$$

is equal to

$$\Pr[\mathcal{B}(u, v, h, u^r, v^{\tilde{r}}, h^{\tilde{\tilde{r}}}) \to 1 \mid r + \tilde{r} = \tilde{\tilde{r}}] \\ - \Pr[\mathcal{B}(u, v, h, u^r, v^{\tilde{r}}, h^{\tilde{\tilde{r}}}) \to 1 \mid r + \tilde{r} \neq \tilde{\tilde{r}}].$$

The decision linear assumption says that the latter is negligible, and so is the former, which is what we wanted. □

6.2 Other Building Blocks

Other building blocks are instantiated as follows.

Groth-Sahai Proofs [29,26]. This is an efficient non-interactive proof system for groups with a bilinear map. This proof system is able to demonstrate quite broad types of algebraic equations hold in a zero-knowledge manner, and is useful to avoid an expensive blowup from general NIZK techniques.

Abe-Haralambiev-Ohkubo Signature [4,2]. This is a structure-preserving signature, in the sense that the signing and verification procedure has no use of non-algebraic operation. This property is essential when the scheme is used together with Groth-Sahai proofs, due to the restriction that Groth-Sahai proofs are unable to treat a kind of non-algebraic relation such as hashing.

The Decision Linear Variant of Cramer-Shoup [41]. Groth-Sahai proofs are highly relying on its use of pairing, and thus we can no longer expect the DDH assumption to hold in our setting. This is why we need to modify the Cramer-Shoup encryption to use the DLIN assumption instead of the classical DDH assumption. Such a DLIN variant of the Cramer-Shoup encryption was proposed by Shacham [41], but we further modify the Shacham's scheme to be tag-based for adopting the one-time signature technique and to be a key encapsulation mechanism for further efficiency than in a direct use of public-key encryption[3].

Encoding Functions. The first encoding function has to encode (i, vk_i) into the message space of the Abe-Haralambiev-Ohkubo scheme. The verification key vk_i is already represented by sixteen elements of $\mathbb{G}$. The identity i of a signer is an integer, but it can be efficiently encoded as g^i. Notice that decoding is also efficient, because the number of group members is polynomial, and so is i. The same thing holds for the second encoding function. In this case, $(i, vk_i, cert_i, s)$ can be encoded as thirty-one group elements of $\mathbb{G}$.[4] Because the Shacham PKE, as well as the Heng-Kurosawa IBE, can be modified to have the session key space $\mathbb{G}^{31}$, the identity encoding function suffices for this purpose. Another important point is that $\langle i, vk_i, cert_i, s\rangle$ is masked by a session key via the group operation of $\mathbb{G}$ for keeping the structure-preserving property, which enables us to adopt Groth-Sahai proofs.

Theorem 7. *When instantiating our construction in Fig. 2 with our decision linear variant of the Heng-Kurosawa k-resilient IBE, the Groth-Sahai proof, the decision linear variant of the Cramer-Shoup encryption, the Abe-Haralambiev-Ohkubo signature scheme, and the one-time signature scheme from the Okamoto identification scheme [38] via the transformation due to Bellare and Shoup [9], the resulting scheme satisfies opener anonymity with k-bounded tokens, admitter anonymity, and traceability.*

6.3 Efficiency

Finally we give a brief efficiency comparison between the proposed scheme and previous group signatures (without message-dependent opening capability).

In the instantiation in Theorem 7, a signature contains 501 elements of $\mathbb{G}$ and 2 elements of $\mathbb{Z}_p$. For a reference, we remark that the group signature of Groth [28] has a signature that consists of 52 elements of $\mathbb{G}$ and 1 elements of $\mathbb{Z}_p$. The message-dependent opening capability is achieved by roughly 10 times blowup

[3] A possible alternative choice here is Kiltz's tag-based encryption [35], which could reduce the size of NIZK proofs due to its public verifiability. One drawback of this scheme is that, to the best of the authors' knowledge, Kiltz's encryption does not allow encrypting multiple group elements with constant ciphertext overhead, while the Cramer-Shoup scheme (and its DLIN variant by Shacham) allow such a modification. See Sect. A.2 for details of this modification.

[4] These thirty-one elements come form one element for g^i, sixteen elements for the verification key vk_i, seven elements for $cert_i$, and seven elements for s.

of the signature size (The Groth scheme allows dynamic joining of members, whereas ours does not, though). From this evaluation, we see that our scheme is fairly practical, or at least implementable in a real system.

Acknowledgment. The authors would like to thank anonymous reviewers for their invaluable comments.

References

1. Abdalla, M., Warinschi, B.: On the Minimal Assumptions of Group Signature Schemes. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 1–13. Springer, Heidelberg (2004)
2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
3. Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005)
4. Abe, M., Haralambiev, K., Ohkubo, M.: Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133 (2010), http://eprint.iacr.org/
5. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385 (2005), http://eprint.iacr.org/
6. Ateniese, G., Camenisch, J.L., Joye, M., Tsudik, G.: A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
7. 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)
8. 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)
9. Bellare, M., Shoup, S.: Two-tier signatures from the Fiat–Shamir transform, with applications to strongly unforgeable and one-time signatures. IET Information Security 2(2), 47–63 (2008)
10. Bichsel, P., Camenisch, J., Neven, G., Smart, N.P., Warinschi, B.: Get Shorty via Group Signatures without Encryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 381–398. Springer, Heidelberg (2010)
11. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
12. Boneh, D., Papakonstantinou, P.A., Rackoff, C., Vahlis, Y., Waters, B.: On the impossibility of basing identity based encryption on trapdoor permutations. In: 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 283–292 (2008)

13. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: 11th ACM Conference on Computer and Communications Security, pp. 168–177. ACM, New York (2004)
14. Boyen, X., Waters, B.: Compact Group Signatures Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
15. Boyen, X., Waters, B.: Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
16. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. Cryptology ePrint Archive, Report 2008/375 (2008), http://eprint.iacr.org/
17. Camenisch, J., Chandran, N., Shoup, V.: A Public Key Encryption Scheme Secure against Key Dependent Chosen Plaintext and Adaptive Chosen Ciphertext Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)
18. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
19. Camenisch, J., Michels, M.: Separability and Efficiency for Generic Group Signature Schemes (Extended Abstract). In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 413–785. Springer, Heidelberg (1999)
20. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
21. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
22. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
23. Delerablée, C., Pointcheval, D.: Dynamic Fully Anonymous Short Group Signatures. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006)
24. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
25. Furukawa, J., Imai, H.: An Efficient Group Signature Scheme from Bilinear Maps. In: Boyd, C., Nieto, J.G. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 455–467. Springer, Heidelberg (2005)
26. Ghadafi, E., Smart, N.P., Warinschi, B.: Groth–Sahai Proofs Revisited. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 177–192. Springer, Heidelberg (2010)
27. Groth, J.: Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
28. Groth, J.: Fully Anonymous Group Signatures Without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
29. 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)

30. Heng, S.-H., Kurosawa, K.: k-Resilient Identity-Based Encryption in the Standard Model. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 67–80. Springer, Heidelberg (2004)
31. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable Signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004)
32. Kiayias, A., Yung, M.: Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076 (2004), http://eprint.iacr.org/
33. Kiayias, A., Yung, M.: Group Signatures with Efficient Concurrent Join. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 198–214. Springer, Heidelberg (2005)
34. Kilian, J., Petrank, E.: Identity Escrow. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 169–185. Springer, Heidelberg (1998)
35. Kiltz, E.: Chosen-Ciphertext Security from Tag-Based Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
36. MacKenzie, P.D., Reiter, M.K., Yang, K.: Alternatives to Non-malleability: Definitions, Constructions, and Applications (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 171–190. Springer, Heidelberg (2004)
37. Ohtake, G., Fujii, A., Hanaoka, G., Ogawa, K.: On the Theoretical Gap between Group Signatures with and without Unlinkability. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 149–166. Springer, Heidelberg (2009)
38. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
39. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science, pp. 543–553. IEEE Computer Society (1999)
40. Schwartz, E.J., Brumley, D., McCune, J.M.: A contractual anonymity system. In: NDSS 2010. The Internet Society (2010)
41. Shacham, H.: A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007), http://eprint.iacr.org/
42. Shoup, V., Gennaro, R.: Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)

A Building Blocks and Their Security Proofs

In the following, let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of a prime order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map.

A.1 Abe-Haralambiev-Ohkubo Signature

The Abe-Haralambiev-Ohkubo signature scheme is as follows [2,4]:

Key Generation. The verification key is $vk = (g'', h'', g', h', \{g_i, h_i\}_{i\in[l]}, a_0, \tilde{a}_0, b_0, \tilde{b}_0, a_1, \tilde{a}_1, b_1, \tilde{b}_1)$, where $g', h' \in \mathbb{G} \setminus \{1\}$, $g_i \leftarrow g'^{\gamma_i}$ and $h_i \leftarrow h'^{\delta_i}$ for random $\gamma_i, \delta_i \leftarrow \mathbb{Z}_p^*$ for $i \in [l]$, $g'' \leftarrow g'^{\gamma''}$, $h'' \leftarrow h'^{\delta''}$ for random $\gamma'', \delta'' \leftarrow \mathbb{Z}_p^*$, $(a_0, \tilde{a}_0, a_1, \tilde{a}_1) \leftarrow \mathsf{Extend}(g', g^\alpha)$ for a random $\alpha \leftarrow \mathbb{Z}_p^*$, and $(b_0, \tilde{b}_0, b_1, \tilde{b}_1) \leftarrow \mathsf{Extend}(h', g^\beta)$ for a random $\beta \leftarrow \mathbb{Z}_p^*$. The signing key is $sk = (\alpha, \beta, \gamma'', \delta'', \{\gamma_i, \delta_i\}_{i\in[l]})$.

Signing. For a message $(m_1, \ldots, m_l) \in \mathbb{G}^l$, choose randomly ζ, ρ, τ, φ, ω from $\mathbb{Z}_p$, compute $z = \tilde{g}^\zeta$, $r = \tilde{g}^{\alpha-\rho\tau-\gamma_z\zeta} \prod_{i=1}^l m_i^{-\gamma_i}$, $s = g'^\rho$, $t = \tilde{g}^\tau$, $u = \tilde{g}^{\beta-\varphi\omega-\delta_z\zeta} \prod_{i=1}^l m_i^{-\delta_i}$, $v = h_r^\varphi$, and $w = \tilde{g}^\omega$, and output (z, r, s, t, u, v, w) as a signature.

Verification. For a pair $(m, \sigma) = ((m_1, \ldots, m_l), (z, r, s, t, u, v, w))$ of a signature and a message, verify two equations $e(a_0, \tilde{a}_0)e(a_1, \tilde{a}_1) = e(g'', z)\, e(g', r)e(s, t) \prod_{i=1}^k e(g_i, m_i)$ and $e(b_0, \tilde{b}_0)e(b_1, \tilde{b}_1) = e(h'', z)e(h', u)e(v, w) \prod_{i=1}^l e(h_i, m_i)$. If both equations hold, output $\top$. Otherwise output $\bot$.

Here, $\mathsf{Extend}(g, h)$ is the algorithm that takes two group elements g and h, picks random $x \in \mathbb{G}$ and $r \in \mathbb{Z}_p$, and outputs $(\mathsf{Rand}(gx^r, h), \mathsf{Rand}(x, h^{-r}))$. Algorithm $\mathsf{Rand}(g, h)$, when $g \neq 1$ and $h \neq 1$, outputs $(g^s, h^{1/s})$ for random $s \in \mathbb{Z}_p^*$. When $g = 1$ or $h = 1$, it outputs $(1, 1)$ with probability $1/(2p-1)$, otherwise outputs one of $(1, x)$ or $(x, 1)$ with probability $1/2$ for random $x \in \mathbb{G} \setminus \{1\}$.

A.2 Shacham's Variant of Cramer-Shoup Encryption

Shacham [41] proposed a variant of the Cramer-Shoup encryption scheme [21,22] modified to be based on the decision linear assumption. The scheme below further modifies the Shacham's variants in two points: (1) Used as a key encapsulation mechanisms and (2) modified to encapsulate a long session key in a constant-size ciphertext. This modified Shacham's variant is as follows:

Key Generation. The public key is $pk = (u, v, h, X = u^x h^{x''}, \tilde{X} = v^{x'} h^{x''}, Y = u^y h^{y''}, \tilde{Y} = v^{y'} h^{y''}, \{Z_i = u^{z_i} h^{z'_i}, \tilde{Z}_i = v^{z'_i} h^{z''_i}\}_{i\in[l]})$, whose corresponding secret key is $dk = (x, x', x'', y, y', y'', \{z_i, z'_i, z''_i\}_{i\in[l]})$.

Encrypt. To encapsulate a session key with a tag t, choose random r and $\tilde{r}$ from $\mathbb{Z}_p$ and compute a ciphertext as $C_{\mathrm{PKE}} = (u^r, v^{\tilde{r}}, h^{r+\tilde{r}}, (XY^t)^r(\tilde{X}\tilde{Y}^t)^{\tilde{r}})$. The session key is $(Z_1^r \tilde{Z}_1^{\tilde{r}}, \ldots, Z_l^r \tilde{Z}_l^{\tilde{r}})$.

Decrypt. To decapsulate a ciphertext (c_1, c_2, c_3, c_4) with a tag t, verify whether $c_1^{x+ty} c_2^{x'+ty'} c_3^{x''+ty''} = c_4$ holds. If it does not hold, output $\bot$, and otherwise output $(c_1^{z_1} c_2^{z'_1} c_3^{z''_1}, \ldots, c_1^{z_l} c_2^{z'_l} c_3^{z''_l})$.

B Security Proofs for the Construction in Sect. 5

B.1 Proof of Theorem 3

Proof. Let $\mathcal{A}$ be an opener anonymity adversary against the proposed scheme. Let $\mathbf{OAnonym}_{\mathcal{A}}$ be the random variable that is 1 when $\mathcal{A}$ correctly guesses the

bit b in the opener anonymity game and is 0 when it does not. Let $\mathbf{OAnonym}'_{\mathcal{A}}$ be a similar random variable with one exception that the common reference string used in the scheme is generated with the zero-knowledge simulator $\mathcal{S}_1$. This change does not affect the probability that the adversary $\mathcal{A}$ wins the game, that is, $|\Pr[\mathbf{OAnonym}_{\mathcal{A}} = 1] - \Pr[\mathbf{OAnonym}'_{\mathcal{A}} = 1]|$ is negligible, due to the zero-knowledge property of the underlying non-interactive proof system. We then show that $|\Pr[\mathbf{OAnonym}'_{\mathcal{A}} = 1] - 1/2|$ is negligible, which concludes the proof.

We construct an adversary $\mathcal{B}$ which attacks the underlying (k-resilient) IBE scheme, and then we relate its success probability to that of $\mathcal{A}$ (in the experiment $\mathbf{OAnonym}'_{\mathcal{A}}$) to obtain the desired bound. The construction of $\mathcal{B}$ is as follows:

Setup. The adversary $\mathcal{B}$ is given as input the master public key par for the identity-based KEM. The adversary $\mathcal{B}$ then generates the rest of a group public key gpk as $(vk_{\mathrm{issue}}, sk_{\mathrm{issue}}) \leftarrow \mathsf{SigKg}(1^\lambda)$, $(pk, dk) \leftarrow \mathsf{TKg}(1^\lambda)$, $(\Sigma, \tau) \leftarrow \mathcal{S}_1(1^\lambda)$, generates user signing keys $(vk_i, sk_i) \leftarrow \mathsf{SigKg}(1^\lambda)$ and their certificates $cert_i \leftarrow \mathsf{Sign}_{sk_{\mathrm{issue}}}(\langle i, vk_i \rangle)$ for all $i \in [n]$. The adversary then sets gpk to $(vk_{\mathrm{issue}}, par, pk, \Sigma)$, sets gsk_i to $(i, vk_i, cert_i, sk_i)$, and run $\mathcal{A}$ with input $(gpk, dk, \{gsk_i\}_{i\in[n]})$.

Token Query (Phase I). When $\mathcal{A}$ makes a token query for a message M, $\mathcal{B}$ makes a key extraction query for M (as an identity) to obtain a decryption key dk_M, and responds $\mathcal{A}$ with dk_M.

Challenge. When $\mathcal{A}$ requests a challenge for (i_0, i_1, M^*), $\mathcal{B}$ proceeds as follows: $\mathcal{B}$ computes two signatures $s_0 \leftarrow \mathsf{Sign}_{vk_{i_0}}(M^*)$ and $s_1 \leftarrow \mathsf{Sign}_{vk_{i_1}}(M^*)$ of the group members i_0 and i_1, generates a one-time signature key pair $(vk^*_{\mathrm{OT}}, sk^*_{\mathrm{OT}})$, and requests a challenge for an identity M^*. Then $\mathcal{B}$ receives a challenge $(C^*_{\mathrm{IBE}}, K^*_{\mathrm{IBE}})$, computes a ciphertext $(C_{\mathrm{PKE}}, K_{\mathrm{PKE}}) \leftarrow \mathsf{TEnc}_{pk}(vk^*_{\mathrm{OT}})$, further computes $\chi^* \leftarrow \langle i_b, vk_{i_b}, cert_{i_b}, s_b \rangle \odot K_{\mathrm{PKE}} \odot K^*_{\mathrm{IBE}}$ for a random bit b, generates a simulated proof π^* with a token τ, and signs $\langle C^*_{\mathrm{PKE}}, C^*_{\mathrm{IBE}}, \chi^*, \pi^* \rangle$ with the one-time signing key sk^*_{OT} to obtain σ^*_{OT}. Finally $\mathcal{B}$ sends $(vk^*_{\mathrm{OT}}, C^*_{\mathrm{PKE}}, C^*_{\mathrm{IBE}}, \chi^*, \pi^*, \sigma^*_{\mathrm{OT}})$ to $\mathcal{A}$ as a challenge.

Token Query (Phase II). The adversary $\mathcal{A}$ continue to issue token queries, and they are answered by $\mathcal{B}$ as before.

Guess. At last $\mathcal{A}$ outputs a bit b', and $\mathcal{B}$ outputs 1 if and only if $b' = b$, otherwise outputs 0.

When the challenger gives the real session key (i.e. C^*_{IBE} is decrypted into K^*_{IBE}), $\mathcal{B}$ perfectly simulates the experiment $\mathbf{OAnonym}'_{\mathcal{A}}$, whereas when a random session key is given, the information on the bit b is perfectly hidden from $\mathcal{A}$. This fact justifies the equality below:

$$\Pr[\mathbf{OAnonym}'_{\mathcal{A}} = 1] - \frac{1}{2} = \Pr[\mathcal{B} \to 1 \mid K^*_{\mathrm{PKE}} \text{ is real}] - \Pr[\mathcal{B} \to 1 \mid K^*_{\mathrm{PKE}} \text{ is random}].$$

Due to the security of the k resilient identity-based KEM, the right hand side is negligible, and it completes the proof. □

B.2 Proof of Theorem 4

Proof. Let $\mathcal{A}$ be an admitter anonymity adversary against the proposed scheme. Let $\mathbf{AAnonym}_{\mathcal{A}}$ be a random variable that indicates $\mathcal{A}$ correctly guesses the bit in the admitter anonymity game. Let $\mathbf{AAnonym}'_{\mathcal{A}}$ be a similar random variable with the exception that the common reference string is replaced to that for simulation. Due to the zero-knowledge property of the proof system, $\mathcal{A}$'s success probability does not change non-negligibly, that is, $|\Pr[\mathbf{AAnonym}_{\mathcal{A}} = 1] - \Pr[\mathbf{AAnonym}'_{\mathcal{A}} = 1]|$ is negligible. We then show that $|\Pr[\mathbf{AAnonym}'_{\mathcal{A}} = 1] - 1/2|$ is negligible, which concludes the actual proof.

We construct an adversary $\mathcal{B}$ which attacks the underlying tag-based KEM. The construction of $\mathcal{B}$ is as follows:

Setup. The adversary $\mathcal{B}$ first runs $\mathsf{SigKg}^{\mathrm{OT}}(1^\lambda)$ to generate a verification/signing key pair $(vk^*_{\mathrm{OT}}, sk^*_{\mathrm{OT}})$, outputs vk^*_{OT} as a target tag, and then receives the public key pk of the tag-based KEM. The adversary $\mathcal{B}$ then generates the rest of a group public key as $(vk_{\mathrm{issue}}, sk_{\mathrm{issue}}) \leftarrow \mathsf{SigKg}(1^\lambda)$, $(par, mk) \leftarrow \mathsf{ISetup}(1^\lambda)$, $(\Sigma, \tau) \leftarrow \mathcal{S}_1(1^\lambda)$, user signing keys $(vk_i, sk_i) \leftarrow \mathsf{SigKg}(1^\lambda)$ for all $i \in [n]$, and their certificates $cert_i \leftarrow \mathsf{Sign}_{sk_{\mathrm{issue}}}(\langle i, vk_i\rangle)$ for all $i \in [n]$. The adversary $\mathcal{B}$ then sets $gpk \leftarrow (vk_{\mathrm{issue}}, pk, par, \Sigma)$ and $gsk_i \leftarrow (i, vk_i, cert_i, sk_i)$ and runs $\mathcal{A}$ with input $(gpk, mk, \{gsk_i\}_{i\in[n]})$.

Open Query (Phase I). When the adversary $\mathcal{A}$ submits an open query for a signature $(vk_{\mathrm{OT}}, C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi, \sigma_{\mathrm{OT}})$ and a message M, the adversary $\mathcal{B}$ responds as follows: (i) when $vk_{\mathrm{OT}} \neq vk^*_{\mathrm{OT}}$, $\mathcal{B}$ makes a decapsulation query for the ciphertext C_{PKE} with a tag vk_{OT} to obtain a session key K_{PKE} (note that this query is legitimate), and then extracts a user decryption key dk_M (of an identity-based KEM) from mk, decrypts C_{IBE} with dk_M to obtain a session key K_{IBE}, and verifies whether $\mathsf{Verify}^{\mathrm{OT}}_{vk_{\mathrm{OT}}}(\langle C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi\rangle, \sigma_{\mathrm{OT}}) = 1$ and $V_{\mathrm{NIZK}}(\ldots) = 1$ hold. If both of them hold, $\mathcal{B}$ further computes $\langle i, vk, cert, s\rangle \leftarrow \chi \odot K^{-1}_{\mathrm{IBE}} \odot K^{-1}_{\mathrm{PKE}}$ and responds with i. Otherwise $\mathcal{B}$ responds with $\perp$. (ii) When $vk_{\mathrm{OT}} = vk^*_{\mathrm{OT}}$, if $\mathsf{Verify}_{vk^*_{\mathrm{OT}}}(\langle C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi\rangle) = \perp$, $\mathcal{B}$ responds with $\perp$. Otherwise $\mathcal{B}$ aborts and outputs a random b'.

Challenge. At some time $\mathcal{A}$ requests a challenge for (i_0, i_1, M^*), $\mathcal{B}$ computes a challenge as follows: $\mathcal{B}$ generates signatures $s_b \leftarrow \mathsf{Sign}_{sk_{i_b}}(M^*)$ for a random bit b, requests a challenge to obtain $(C^*_{\mathrm{PKE}}, K^*_{\mathrm{PKE}})$, generates a ciphertext and a session key as $(C^*_{\mathrm{IBE}}, K_{\mathrm{IBE}}) \leftarrow \mathsf{IEnc}_{par}(M^*)$, computes $\chi^* \leftarrow \langle i_b, vk_{i_b}, cert_{i_b}, s_{i_b}\rangle \odot K^*_{\mathrm{PKE}} \odot K_{\mathrm{IBE}}$, and generates a fake proof π^*. Finally $\mathcal{B}$ signs $\langle vk^*_{\mathrm{OT}}, C^*_{\mathrm{PKE}}, C^*_{\mathrm{IBE}}, \chi^*, \pi^*\rangle$ with the one-time signing key sk^*_{OT} to obtain σ^*_{OT} and sends $(vk^*_{\mathrm{OT}}, C^*_{\mathrm{PKE}}, C^*_{\mathrm{IBE}}, \chi^*, \pi^*, \sigma^*_{\mathrm{OT}})$ to $\mathcal{A}$.

Open Query (Phase II). Again $\mathcal{A}$ submits more open queries and $\mathcal{B}$ responds as before.

Guess. When $\mathcal{A}$ outputs a bit b, $\mathcal{B}$ outputs 1 if and only if $b' = b$, otherwise outputs 0.

Let F denote the event that the adversary $\mathcal{A}$ submits an open query $(vk_{\mathrm{OT}}, C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \pi, \sigma_{\mathrm{OT}})$ where $vk_{\mathrm{OT}} = vk^*_{\mathrm{OT}}$ and $\mathsf{Verify}_{vk_{\mathrm{OT}}}(\langle C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \pi\rangle) = \top$. Due to its perfect simulation of the experiment $\mathbf{AAnonym}'_{\mathcal{A}}(k, n)$ (with only

exception of aborting in the event F), when given the real session key $K^*_{\mathcal{PKE}}$, $\mathcal{B}$ outputs 1 whenever $\mathcal{A}$ successfully guesses a bit and the event F does not occur. In addition, when given a random session key, $\mathcal{B}$ gives no information on the bit b to $\mathcal{A}$. The inequality below can be obtained from these two facts:

$$\begin{aligned}
&\left|\Pr[\mathbf{AAnonym}'_{\mathcal{A}} = 1] - \frac{1}{2}\right| \\
&\quad = \left|\Pr[\mathbf{AAnonym}'_{\mathcal{A}} = 1 \wedge F] + \Pr[\mathbf{AAnonym}'_{\mathcal{A}} = 1 \wedge \neg F] - \frac{1}{2}\right| \\
&\quad \leq \left|\Pr[F] + \Pr[\mathbf{AAnonym}'_{\mathcal{A}} = 1 \wedge \neg F] - \frac{1}{2}\right| \\
&\quad \leq \Pr[F] + |\Pr[\mathcal{B} \to 1 \mid K^*_{\mathrm{PKE}} \text{ is real}] - \Pr[\mathcal{B} \to 1 \mid K^*_{\mathrm{PKE}} \text{ is random}]|
\end{aligned}$$

Finally we prove $\Pr[F]$ is negligible to complete the proof.

To prove $\Pr[F]$ is negligible, we will construct another adversary $\mathcal{F}$, which attacks strong unforgeability of the one-time signature scheme and relate its success probability with the probability of the event F. The construction of $\mathcal{F}$ is as follows:

Setup. The adversary $\mathcal{F}$ first receives a verification key vk^*_{OT} for the one-time signature scheme. The adversary then runs $\mathsf{GKg}(1^\lambda, 1^n, 1^k)$ to obtain a group public key $gpk = (vk_{\mathrm{issue}}, pk, par, \Sigma)$, the opening key ok, the message-specification key msk, and user signing keys $gsk_i = (i, vk_i, cert_i, sk_i)$ for all $i \in [n]$.

Open Query (Phase I). Queries are answered with the opening key ok and the message-specifying key msk. In addition, when $\mathcal{A}$ queries a group signature $(vk_{\mathrm{OT}}, C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi, \sigma_{\mathrm{OT}})$ in which $vk_{\mathrm{OT}} = vk^*_{\mathrm{OT}}$ and $\mathsf{Verify}_{vk_{\mathrm{OT}}}(\langle C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi\rangle, \sigma_{\mathrm{OT}}) = \top$, $\mathcal{F}$ stops the simulation and outputs this $(\langle C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi\rangle, \sigma_{\mathrm{OT}})$ as a forgery.

Challenge. To respond to the challenge request (i_0, i_1, M^*), $\mathcal{F}$ chooses a random bit b and generate a group signature $(vk^*_{\mathrm{OT}}, C^*_{\mathrm{PKE}}, C^*_{\mathrm{IBE}}, \chi^*, \pi^*, \sigma^*_{\mathrm{OT}})$ in the way exactly same to the construction with one exception that σ^*_{OT} is obtained by querying $\langle C^*_{\mathrm{PKE}}, C^*_{\mathrm{IBE}}, \chi^*, \pi^*\rangle$ to the signing oracle.

Open Query (Phase II). Further open queries are answered as in the phase I.

Guess. If $\mathcal{A}$ outputs a guess and halts, $\mathcal{F}$ halts without outputting a forgery.

Whenever the event F happens, this adversary $\mathcal{F}$ successfully outputs a forgery and wins the game (Because $(C_{\mathrm{PKE}}, C_{\mathrm{IBE}}, \chi, \pi, \sigma_{\mathrm{OT}})$ must be different from $(C^*_{\mathrm{PKE}}, C^*_{\mathrm{IBE}}, \chi^*, \pi^*, \sigma^*_{\mathrm{OT}})$, and it consists a legitimate strong forgery). Then we can conclude $\Pr[F]$ is negligible, because of the security of the underlying one-time signature scheme. □

B.3 Proof of Theorem 5

Proof. Let $\mathcal{A}$ be a traceability adversary against the proposed scheme. We first classify successful forgery that $\mathcal{A}$ may produce.

The forgery is opened to $\perp$: In this case, either C_{PKE} or C_{IBE} is invalid (decapsulated to $\perp$) or $\chi \odot K_{\mathrm{IBE}}^{-1} \odot K_{\mathrm{PKE}}^{-1}$ cannot be parsed. In all of these case, π is an invalid proof for a false statement.

The forgery is opened to $i \in \mathbb{N}$: In this case all C_{PKE}, C_{IBE}, and χ have been correctly decrypted, and when $\chi \odot K_{\mathrm{IBE}}^{-1} \odot K_{\mathrm{PKE}}^{-1}$ is parsed into $\langle i', vk', cert', s' \rangle$, either one of the following three cases will hold: (i) $\mathsf{Verify}_{vk_{\mathrm{issue}}}(\langle i', vk' \rangle, cert') = \perp$ or $\mathsf{Verify}_{vk'}(M, s) = \perp$, (ii) $cert'$ is a valid signature, but $\langle i', vk' \rangle$ was not singed at the setup phase, or (iii) $(\langle i, vk' \rangle, cert')$ is the same one generated at the setup phase. Note that in case (i) the proof π is a valid proof for the false statement, in case (ii) $cert'$ is a forgery for the verification key vk_{isuue}, and in case (iii) s' is a forgery for the user verification key vk_i.

To bound the probability that $\mathcal{A}$ outputs a forgery of case (iii), we construct a forger $\mathcal{B}$ against the underlying EUF-CMA signature scheme. The construction of $\mathcal{B}$ is as follows: The forger $\mathcal{B}$ receives a verification key vk, and $\mathcal{B}$ uses this verification key as a user verification key vk_{i^*}, where i^* is randomly chosen by $\mathcal{B}$. Other components of the public key and the secret keys for the group members and the authorities are honestly generated by $\mathcal{B}$. Then $\mathcal{B}$ runs $\mathcal{A}$ with input the group public key gpk, the opener key ok, and the admitter key msk. Signing queries for the user i^* can be simulated with the signing oracle of the underlying scheme, group signing key revealing query for the user i^* cannot be simulated, in which case $\mathcal{B}$ aborts. Other signing queries and key revealing queries can be answered by $\mathcal{B}$ itself. When $\mathcal{A}$ outputs a forgery (M, σ), $\mathcal{B}$ verifies the one-time signature and the non-interactive proof in σ, decrypts ciphertexts in σ to obtains the plaintext $\langle i', vk', cert', s' \rangle$, verifies $cert'$ by vk_{issue} and s' by vk', confirms that $(i', vk') = (i^*, vk)$, and finally outputs (M, s) as a forgery. If one of these procedures fails, $\mathcal{B}$ stops. Since $\mathcal{B}$ is a legitimate forger of the signature scheme and whenever $\mathcal{A}$ produces a forgery of case (iii) also $\mathcal{B}$ does a successful forgery, we obtains a bound for the case (iii).

To bound the probability that $\mathcal{A}$ outputs a forgery of case (ii), we construct another forger $\mathcal{B}'$. This time $\mathcal{B}$ receives a key vk and uses it as a certificate verification key vk_{issue}, and obtains certificates $cert_i$ for all group members $i \in [n]$ by querying the signing oracle of the underlying scheme. Signing queries and key revealing queries issued by $\mathcal{A}$ are correctly answered by $\mathcal{B}$ itself for all group members. When $\mathcal{A}$ outputs (M, σ), $\mathcal{B}$ verifies the validity of σ, confirms that σ contains a certificate forgery, and outputs this forged certificate, otherwise stops. Also in this case $\mathcal{B}$ is a legitimate forger against the underlying scheme, and thus the probability that $\mathcal{A}$ produces a forgery of the case (ii) is bounded.

Since the other cases of forgeries $\mathcal{A}$ may produce contains a valid proof of a false statement, the probability that $\mathcal{A}$ produces such a forgery is bounded due to the underlying non-interactive proof system. □

Short Pairing-Efficient Threshold-Attribute-Based Signature

Martin Gagné[1], Shivaramakrishnan Narayan[2], and Reihaneh Safavi-Naini[3,⋆]

[1] Saarland University, Germany
gagne@cs.uni-saarland.de
[2] Optimal Payments Plc, Canada
kris.narayan@optimalpayments.com
[3] University of Calgary, Canada
rei@ucalgary.ca

Abstract. In this paper, we construct a new threshold-attribute-based signature (t-ABS) scheme that is significantly more efficient than all previous t-ABS schemes. The verification algorithm requires the computation of only 3 pairing operations, independently of the attribute set of the signature, and the size of the signature is also independent of the number of attributes. The security of all our schemes is reduced to the computational Diffie-Hellman problem. We also show how to achieve shorter public parameters based on the intractability of computational Diffie-Hellman assumption in the random oracle model.

1 Introduction

Attribute-based cryptography has received much attention in recent years. It provides an elegant cryptographic solution for enforcing role-based and attribute-based access policies. Attribute-based encryption (ABE) schemes [8,2] were first proposed to control decryption of messages by attaching attributes to the ciphertext. In these schemes, access policy is associated with users' private key and the decryption of the ciphertext is conditional on the set of attributes attached to a ciphertext, to satisfy the users' private key policy. Such ABE is called *key-policy ABE*. Bethencourt et al [1] later proposed *ciphertext-policy ABE* where policies are attached to ciphertexts, and decryption is conditional on users' having private key for attributes satisfying the ciphertext-policy.

Threshold-attribute-based signatures (ABS), proposed in [9] and referred to as t-ABS, can be seen as a generalization of identity-based signatures, where private keys are associated with a set of n attributes satisfying a (t, n) threshold policy. A signature of the user can use a subset of size n', $t \leq n' \leq n$ of attributes. A signed message can be verified against a verification attribute set, and verification will succeed if t of the attributes in the verification attribute set match the attributes of the signature attribute set. Threshold ABS lets a user

⋆ Financial support for this research (all authors) was provided in part by Alberta Innovates - Technology Futures, in the Province of Alberta in Canada.

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 295–313, 2013.

sign a message once with a subset of his attributes without leaking any information about the other attributes he holds. He can also sign a message with a number n of attributes larger than the threshold, which allows verification for any $\binom{n}{t}$ possible verification sets. t-ABSs have been used [9] to construct attribute-based anonymous credential systems, in which a user's credential is a message that includes n attributes of the user, and is signed by a trusted authority using a t-ABS. Instead of directly presenting the signed credentials to a verifier, the system is equipped with protocols that allow credential holders to "prepare" (convert) and "show" their credentials such that verification only guarantees that the credential has t common attributes with the verification attribute set and nothing else.

Important efficiency measures of ABS schemes are the signature length (communication cost) and the computation required for signature and verification. Most ABS schemes are defined over groups which admit efficient bilinear pairing, where the pairing operation is used in signature verification. All known ABS schemes have signature sizes that are linear in the number of attributes used to generate the signature. Moreover, verification usually requires the computation of a number of pairing operation linear in the number of attributes used in verification. Pairings are costly operation when compared to exponentiation in the base group – when pairings are used on an elliptic curve defined over a field of q elements, the last operation of the pairing computation is an exponentiation in a field of q^k elements, where k is the embedding degree of the elliptic curve, which alone tends to be more expensive than an exponentiation on the elliptic curve. While recent results by Lauter et al. [3] have dramatically reduced the computational cost of pairings, their best techniques require the simultaneous computation of many pairings, something that may not be possible on memory-restricted devices.

In this paper, we focus on signature size and number of pairing operations required for signature verification as the main efficiency measures. These are the most important measures in applications where reducing the server computation and communication bandwidth are important. For example, consider a scenario where a server in a mobile service provider company needs to respond (verify) to high volumes of credential verification requests received from mobile devices. Here, small size of signatures save the communication cost and lower verification computation will reduce the computation load of the server. Our choice to restrict ourselves to threshold-attribute-based signature enables us to obtain an extremely efficient scheme, which we could not obtain for more general attribute policies, but still retain enough functionality for practical applications.

1.1 Our Contributions

We give constructions of t-ABS schemes with *constant number of pairing operations (3) for verification* and constant size signature. We give two variants of these schemes: the first for fixed verification attribute set, and the second for general verification attribute set and threshold verification. We discuss the

notion of privacy for t-ABS, and show that the constructions presented provide signer-attribute privacy.

◇ **t-ABS:** We present a (t,t)-ABS scheme in which the verification requires all the t attributes used for signing. The (t,t)-ABS scheme is first constructed for small attribute universe, then generalized for large universe (or for when the number of attributes is a priori unknown). The attribute universe in the small universe attribute construction consists of integers 1 to $\hat{n}$, while for large universe case, it consists of all binary strings of arbitrary size. Both schemes have constant size (2 group elements) signatures and require 3 pairing operations for signature verification. While both schemes still require a linear number of exponentiations for verification – and therefore this result does not present an *asymptotic* improvement over previous results – this still represents an important speedup over other schemes. They are also the first t-ABS schemes that have signature size independent from the size of the attribute set. Security of both schemes is proven against chosen message attack in standard model and relies on the difficulty of the computational Diffie-Hellman problem.

Next we show how to construct a (t,n)-ABS scheme from our (t,t)-ABS scheme such that the computation cost of verification remains 3 pairing operations (and a linear number of exponentiations). The signature length now grows linearly with the size of the attribute set used for signing. This however is inevitable as the signature needs to include all the attributes that are used for signing. We prove security of the (t,n)-ABS in the same framework (standard model and against chosen message attack) based on computational Diffie-Hellman assumption.

In Table 1 in Section 4, we compare the efficiency of our scheme with the known ABS, with special attention given to signature size and verification computation. We note that the computation cost of signature generation in all schemes are similar to previous schemes, and so have not been included. The size of public parameters in all schemes is linear in either the size of attribute universe (small attribute universe) or the maximum size of a user attribute set (large attribute universe).

◇ **Signer-Attribute Privacy:** Our signature scheme protects the privacy of the signer in that it is not possible from a signature to deduce anything about attributes of the signer other than the set of attributes that are attached to the signature, the other attributes of the signer remain hidden.

◇ **Extensions:** Finally, in Section 5, we discuss how the size of the system parameters can be reduced and the reduction tightened in the random oracle model. We also discuss different techniques that could be used to make the scheme more versatile by allowing for flexible thresholds, and how different pairing types could be used.

◇ **Relevance of the Result:** Digital signatures provide a powerful mechanism for authentication ensuring the origin of of signed documents. Threshold-ABS schemes allow authentication to be based on the users' attributes instead of their identities, and so protect identities of the users and also attributes that

are not used for signing. As shown in [9], they can be used as the main building block of an attributed based credential system also. The t-ABS signatures in this paper require 3 pairings for verification. The signatures are short (2 group elements for (t, t) signature, and $(n + 1)$ group elements for t, n signature), and their security rely on the hardness of the well established computational Diffie-Hellman problem. This, together with efficient verification and short signature size, makes our proposed schemes truly practical.

1.2 Related Work

There exist two variants for attribute-based signature: key-policy ABS, in which the signing key is associated with a policy and a message is signed with an attribute set satisfying the policy, and signature-policy ABS, in which the signing key is associated with an attribute set and a message is signed with a policy satisfied by the attribute set.

In [9], a formal definition and security model for threshold key-policy ABS were proposed and constructions for small attribute universe and large attribute universe were given. The proposed schemes provide existential unforgeability for selective message and selective attribute security model. These signatures require $2(n+1)$ group elements, and verification consists of t pairings and $3t$ exponentiations, where n and t denote the number of attributes used for signing, and the threshold, respectively. Authors show application of ABS to credential systems and construct protocols that allow the holder of a credential that is generated by a t-ABS, to prove the ownership of the credential in a privacy preserving way. Li et al., [5] presented an efficient t-ABS which compared to [9], improves verification cost of the signature by 60% and the signature size by 50%. The signature size of the scheme is $|B| + d - k + 2$ group elements, where B denotes the set of attributes of the claim predicate, k denotes the threshold of the policy and d is a parameter that allows threshold to be varied from 1 to $d - 1$. The verification requires $(|B| + d - k + 2)$ pairing operations. The scheme is existentially unforgeable in the selective attribute model against chosen message attack.

Maji et al., in [6] presented a general framework for constructing signature-policy ABS schemes which are fully secure in the adaptive attribute security model. In the fully secure model, the adversary does not commit to the attribute set of the forged message, and the attack is successful in existential sense (see Section 2.2). They showed how to construct signature-policy ABS schemes using credential bundles and non-interactive witness indistinguishable proofs. Two of their schemes are proven secure in the standard model using the strong Diffie-Hellman problem and symmetric external Diffie-Hellman problem, but are inefficient. Their third scheme is much more efficient, but is only proven secure in the generic group model. Recently, Okamoto and Takashima [7] presented an efficient signature-policy ABS scheme that can even accommodate non-monotone policies, and which can be proven secure in the standard model under the decision linear assumption. However, the length of their signatures and the number of pairing operations required for verification is still linear in the size of the access policy.

Attribute-based signatures in [6,5,9] also provide the additional property of attribute privacy. In Maji et al. [6], the signer privacy for signature policy ABS is defined as the property where the signature reveals nothing about the attributes used to produce the signature. Li et al. in [5] define signer attribute privacy for their key-policy ABS as, the signature reveals nothing about the identity or the attributes of the signer beyond what is explicitly revealed by the claim. In [9] the authors define two notions of attribute privacy for signature holder called weak signer-attribute privacy and full signer-attribute privacy. In the weak signer-attribute privacy t-ABS, the signature should not reveal any attributes of the signer other than the threshold t number of attributes in common with the verification attribute set. In full signer-attribute privacy, the signature verification proof reveals no information other than the validity of the prover's signature to the verifier.

2 Background

In this section, we recall the definition of threshold attribute-based signature, its security notions and the intractability assumptions used to construct our scheme. We assume there is a universal set of attributes $\mathcal{U}$ which is publicly available to all users. Each signer holds a set of attributes ω_u where $\omega_u \subset \mathcal{U}$. There exists a Trusted Authority (TA) who verifies attributes of users and issues private keys that match their attributes.

2.1 Threshold Attribute-Based Signature

A threshold attribute-based signature (t-ABS) scheme consists of four algorithms: **Setup**, **Extract**, **Sign** and **Verify**. For simplicity of exposition, we describe a scheme in which the threshold t in the key policy is always the same. It is straightforward to generalize these schemes to allow each key policy to have its own threshold.

Setup: This algorithm is run by the TA. Given a security parameter κ, the algorithm outputs the system public parameters $params$ and master secret key msk.

Extract: This algorithm is run by the TA. Given an attribute set $\omega_u \subset \mathcal{U}$ of size at least t, the algorithm outputs the private key sk_{ω_u}.

Sign: Given a message m, a secret key sk_{ω_u} and an attribute set $\omega_s \subseteq \omega_u$ of size t, this algorithm to produce a signature σ on m that attests that the user possesses the attributes in ω_s.

Verify: Given a message m, a signature σ and a verification attribute set ω_v of size t, this algorithm outputs 1 if σ is a valid signature on attributes ω_v, otherwise, it returns 0.

Note that the attribute set ω_u given as input to the **Extract** algorithm is really a shorthand for a t-out-of-ω_u policy, and that the key returned is a key for that policy.

2.2 Security Notions for Attribute-Based Signature

Existential Unforgeability. The standard notion of security for attribute-based signature is existential unforgeability under an adaptive chosen message attack (EUF-CMA). This notion is defined by the following game between an adversary $\mathcal{A}$ who tries to break the scheme, and a challenger $\mathcal{C}$ who provides the environment for the attack:

Setup: The challenger $\mathcal{C}$ chooses a security parameter 1^λ and runs **Setup**. $\mathcal{C}$ keeps the master secret and returns the *params* to the adversary $\mathcal{A}$.

Query Phase: This is the phase during which $\mathcal{A}$ can perform polynomially bounded number of **Extract** and **Sign** queries to $\mathcal{C}$ for any user attribute set ω_u and (m, ω_u, ω_s), respectively. The challenger responds with the the private key and the signature, respectively.

Forgery Phase: $\mathcal{A}$ outputs a signature pair (σ^*, m^*), and an attribute set ω_s^*. $\mathcal{A}$ wins if σ^* is a valid signature provided for all queried sets of attributes ω_u during the query phase, we have $|\omega_u \cap \omega_s^*| \leq t-1$, and for all queried pairs (m, ω_u, ω_s) during Query Phase, we have $m \neq m^*$ or $|\omega_u \cap \omega_s^*| \leq t-1$.

The adversary succeeds in the above attack game if it produces a valid forgery. A weaker notion of security for signature schemes is existential unforgeability under selective chosen message attack (EUF-sCMA), in which the adversary has commit to a chosen message before seeing the public parameters and the forgery must be produced on that message.

An even weaker notion of security is selective attribute unforgeability, which can be considered under adaptive chosen message attack (s-EUF-CMA), or selective chosen message attack (s-EUF-sCMA).
The attack game for s-EUF-CMA is as follows:

Commit: The adversary commits on a chosen attribute set ω_u^* for which it produces the forgery.

Setup: Same as the *Setup* in EUF-CMA game presented above.

Query Phase: $\mathcal{A}$ can perform polynomially bounded number of **Extract** and **Sign** queries to $\mathcal{C}$ for any user attribute set ω_u such that $|\omega_u \cap \omega_u^*| < t$ and for any (m, ω_u, ω_s), respectively. The challenger responds with the the private key and the signature, respectively.

Forgery Phase: $\mathcal{A}$ outputs a signature pair (σ^*, m^*), and the attribute set $\omega_s^* \subseteq \omega_u^*$. $\mathcal{A}$ wins if σ^* is a valid signature provided for all queried sets of attributes ω_u during the query phase, we have $|\omega_u \cap \omega_s^*| \leq t-1$, and for all queried pairs (m, ω_u, ω_s) during Query Phase, we have $m \neq m^*$ or $|\omega_u \cap \omega_s^*| \leq t-1$.

The adversary succeeds in the above attack game if it produces a valid forgery. The attack game for s-EUF-sCMA is similar, except that the attacker has to commit to both an attribute set and a message before receiving the public parameters.

The ABS constructions in [6], are proven secure against existential forgery under adaptive chosen message attack (EUF-CMA). The ABS scheme of [9] was

proven secure using the weaker notion of existential unforgeability under the selective attribute and selective chosen message attack in the standard model, whereas the ABS scheme of [5], is proven secure under the selective attribute, adaptive chosen message attack.

In this paper, we use the selective attribute existential unforgeability property under adaptive chosen message attack (s-EUF-CMA). Recently, Lewko et al. [4] presented the first adaptive attribute secure *ABE scheme* over composite groups. However the security of all t-ABS so far, has been proven in selective attribute model.

Definition 2.21. *A threshold attribute-based signature is existentially unforgeable against chosen message attack property in the selective attribute model, if no polynomially bounded adversary $\mathcal{A}$ can succeed in producing a valid forgery in the s-EUF-CMA attack game presented above. The adversary's advantage is:*

$$Adv_{\mathcal{A}}^{s\text{-}EUF\text{-}CMA} = Pr[Verify(m^*, \sigma^*, \omega_s^*) = 1].$$

Privacy. An attribute-based signature scheme is private if the distribution of the signature is independent of the key that was used to produce it. This has very different implications depending on whether one is considering an signature-policy ABS or a key-policy ABS. In the former case, privacy means that one cannot guess from a signature with policy ρ which attribute set, among all attribute sets satisfying the policy ρ, is held by the signer. In the latter case, privacy means that one cannot divine, from a signature on an attribute set ω, any information about the key-policy ρ held by the signer, other than the fact that is satisfies ω.

We specialize the definition of privacy to key-policy threshold-ABS as follows.

Definition 2.22. *A key-policy threshold ABS is private if for any message m, all $(params, msk)$ produced by the **Setup** algorithm, all attribute sets ω_1, ω_2, all signing keys $sk_{\omega_1} \xleftarrow{\$}$ **Extract**(msk, ω_1) and $sk_{\omega_2} \xleftarrow{\$}$ **Extract**(msk, ω_2), all attribute sets ω_s with $\omega_s \subset \omega_1 \cap \omega_2$, the distributions of **Sign**$(params, sk_{\omega_1}, \omega_s, m)$ and **Sign**$(params, sk_{\omega_2}, \omega_s, m)$ are equal.*

We note that Li et al. [5] present a key-policy threshold ABS scheme in which the attribute set ω_s of a signature can be *further* 'hidden' inside a threshold signature policy. The resulting scheme is therefore a hybrid of key-policy and signature-policy ABS, and offers the privacy guarantees of both variants. Since the main focus of this paper is efficiency, we leave it as an open problem to determine if the same can be done with our scheme.

2.3 Complexity Assumptions

Computational Diffie-Hellman Problem:
Given the values $(g, g^a, g^b) \in \mathbb{G}$ where $a, b \in Z_p^*$, and g is the generator of $\mathbb{G}$, the computational Diffie-Hellman problem is to compute the value g^{ab}.

We define the advantage of an adversary $\mathcal{B}$ in solving the computational Diffie-Hellman problem as,

$$Adv_{\mathcal{B}}^{CDH} = \Pr[\mathcal{B}(\mathbb{G}, p, g, g^a, g^b) = g^{ab}],$$

where the probability is over the choice of a, b and the coins of algorithm $\mathcal{B}$. We say that the computational Diffie-Hellman assumption holds if $Adv_{\mathcal{B}}^{CDH}$ is negligible in terms of the input for all probabilistic polynomial-time algorithms $\mathcal{B}$.

Collision Resistant Hash Function:
Let $\mathcal{F} = \{H : \{0,1\}^m \to \{0,1\}^l\}$ be a family of hash functions, where m and l are functions of the security parameter κ such that $m > l$. We define the advantage of an adversary $\mathcal{C}$ in breaking the collision resistance of $\mathcal{F}$ as

$$Adv_{\mathcal{C}}^{CR} = \Pr[(x, y) \leftarrow \mathcal{C}(H); H(x) = H(y)],$$

where the probability is over the random choice of H in $\mathcal{F}$ and the random coins of the algorithm $\mathcal{C}$. We say that the family of functions $\mathcal{F}$ is collision resistant if $Adv_{\mathcal{C}}^{CR}$ is negligible for all probabilistic polynomial-time algorithms $\mathcal{C}$.

2.4 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of prime order p. A bilinear map is a function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, satisfying the following properties.
- e is bilinear, i.e. for all $g \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$, $e(g^a, g^b) = e(g, g)^{ab}$.
- e is non-degenerate, i.e. given $g_1, g_2 \in \mathbb{G} \setminus \{1_{\mathbb{G}}\}$, $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$.
- e is efficiently computable.

3 Constructing Attribute-Based Signatures

In this section we provide the constructions for ABS and prove their security. We first describe two (t, t)-ABS schemes which allow a signer to generate a signature for a subset of t attributes and verification algorithm is on the same set. The first scheme is for a small attribute universe in which we assume that $|\mathcal{U}| = \hat{n}$. The second scheme we describe admits a large attribute universe, in which any elements of $\mathbb{Z}_p$ can be an attribute. Both schemes are proven secure against chosen message attack based on the intractability of the computational Diffie-Hellman assumption. In both constructions the signature size is constant (two group elements), and verification cost for both schemes is only three pairing operations. In Section 3.3, we extend the construction to a (t, n)-ABS. The main difference between the (t, t)-ABS scheme and (t, n)-ABS scheme is that in the former we can aggregate components of the signature (each corresponding to an attribute) since the verification is performed on the same t signing attributes, whereas in the (t, n)-ABS construction the signer can choose to reveal any number $n \geq t$ of his attributes, and the verification can be on any t-of-n attributes, so components of the signature cannot be aggregated.

3.1 (t,t)-ABS Scheme - Small Attribute Universe

In this construction we assume the universe $\mathcal{U}$ is of fixed size $\hat{n}$. This construction requires the universe to be relatively small, as the length of the public parameters grows linearly with the size of the universe.

We assume, without loss of generality, that attribute universe is $\{1, \ldots, \hat{n}\}$. In practice, one would simply need to establish a mapping from the real attribute set $\{attr_1, \ldots, attr_{\hat{n}}\}$ to $\{1, \ldots, \hat{n}\}$. We also denote the Lagrange coefficient $\Delta_{i,N}(x) = \prod_{j \neq i, j \in N} \frac{x-j}{i-j}$ for $i \in \mathbb{Z}_q^*$, and a set N of elements in $\mathbb{Z}_p$.

Setup: Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, and let g be a generator for $\mathbb{G}$. $H : \{0,1\}^* \to \{0,1\}^{n_m}$ is sampled from appropriate families of collision-resistant hash functions. Compute the following:

- Sample $a, b \in_R \mathbb{Z}_p^*$, and set $g_1 = g^a$, $g_2 = g^b$
- Sample $z_i \in_R \mathbb{Z}_p^*$ for $1 \leq i \leq \hat{n}$, and set $h_i = g^{z_i}$
- Sample $h_0, u_0, \ldots, u_{n_m} \in_R \mathbb{G} \setminus \{1_\mathbb{G}\}$

The public parameters are: $\{\mathbb{G}, \mathbb{G}_T, e, p, g, g_1, g_2, h_0, \ldots, h_{\hat{n}}, u_0, \ldots, u_{n_m}, H\}$, and the master secret is: $\{a, b, z_1, \ldots, z_{\hat{n}}\}$.

We define the function $W(x) = u_0 \prod_{i=1}^{n_m} u_i^{x[i]}$, where x is a binary string of length n_m and $x[i]$ represents the i-th bit of x.

Extract: Let ω_u be the attribute set of a user such that $\omega_u \subseteq \mathcal{U}$. The private key of ω_u is constructed as follows:

1. Let f_u be a random polynomial of degree $t-1$ such that $f_u(0) = a$.
2. For each $i \in \omega_u$, compute $D_i = \frac{g_2^{f_u(i)+z_i}}{h_0{}^{z_i}}$
3. The private key of a user is: $(\{D_i\}_{i \in \omega_u})$.

Sign: Given a signing key sk_{ω_u} and a signing attribute set $\omega_s \subset \omega_u$ with $|\omega_s| = t$, the signature of a message $m \in \{0,1\}^*$ is calculated as follows:

1. Sample $c \in_R \mathbb{Z}_p^*$.
2. Compute $\rho = H(m||\omega_s)$ and $\mathcal{S} = W(\rho)^c \cdot \prod_{i \in \omega_s} (D_i)^{\Delta_{i,\omega_s}(0)}$
3. The signature is: $(\mathcal{S}, g^c)$.

Verify: Given a message m and signature (S_1, S_2) and verification attribute set ω_v of size t, a verifier proceeds as follows:

1. Compute $\hat{\rho} = H(m||\omega_v)$ (Note, $\omega_s = \omega_v$).
2. Verify that the following equation holds: $\frac{e(g,S_1)e(h_0 \cdot {g_2}^{-1}, \prod_{i \in \omega_v} h_i^{\Delta_{i,\omega_v}(0)})}{e(W(\hat{\rho}),S_2)} = e(g_1, g_2)$. If so, output 1, else, output 0.

Since $e(g_1, g_2)$ can be pre-computed, it is easy to see that signature verification requires only three pairings. One can easily verify that, when the algorithms are executed correctly, the verification algorithm will always accept a signature produced by the signing algorithm.

Theorem 3.11. *Under the assumption that the hash function H possess the property stated above, any s-EUF-CMA adversary $\mathcal{A}$ against the scheme above*

can be used to construct algorithms $\mathcal{B}$ and $\mathcal{C}$ that have essentially the same running time as $\mathcal{A}$ and that solve the computational Diffie-Hellman problem and break the collision-resistance of H respectively, with

$$Adv_{\mathcal{A}}^{CMA} \leq 8Q(n_m + 1)Adv_{\mathcal{B}}^{CDH} + Adv_{\mathcal{C},H}^{CR},$$

where Q is the number of signature queries made by $\mathcal{A}$ and n_m is the output size of the hash function.

A sketch of the proof of this theorem is in Appendix A.

Theorem 3.12. *The scheme above is private*

Proof 3.11 *A signature on attribute set ω_s is always of the form: $(g_2^a \cdot W(\rho)^c \cdot \prod_{i\in\omega_s}\left(g_2^{z_i}h_0^{-z_i}\right)^{\Delta_{i,\omega_s}(0)}, g^c\Big)$ for random c and $\rho = H(m\|\omega_s)$. The distribution of the signature is therefore clearly the same regardless of the key that was used to produce it.*

3.2 (t, t)-ABS Large Universe Scheme

In our first construction, the size of the public parameters grows linearly with the number of attributes in the universe. In this section we present a scheme which uses all elements of $\mathbb{Z}_p$ as the attribute universe $\mathcal{U}$. The scheme requires a pre-defined number $\widetilde{n}$ which represents the maximum size of an attribute set a user can possess. This kind of 'large universe' construction is of particular interest when the whole attribute set is not known a priori, or when the total number of attribute is so large and the previous scheme becomes impractical.

In practice, one would use a collision resistant hash function to map the description of each attribute to an element in $\mathbb{Z}_p$.

Setup: Let $(\mathbb{G}, \mathbb{G}_T, e, p, g, g_1, g_2, h_0, u_0, \ldots, u_{n_m}, H)$ be defined as in Scheme 3.1. In addition, we sample $z'_i \in_R \mathbb{Z}_p^*$ for $1 \leq i \leq \widetilde{n} + 1$, and set $h_i = g^{z'_i}$

We also define the function $T(x) = g_1^{x^{\widetilde{n}}} \prod_{j=1}^{\widetilde{n}+1} h_j^{\Delta_{j,N}(x)}$, where $\Delta_{j,S}(x) = \prod_{k\neq j, k\in S} \frac{x-k}{j-k}$ is the Lagrange coefficient and N is the set $\{1, \ldots, \widetilde{n} + 1\}$.

The public parameters are $\{\mathbb{G}, \mathbb{G}_T, e, p, g, g_1, g_2, h_0, \ldots, h_{\widetilde{n}+1}, u_0, \ldots, u_{n_m}, H\}$ and the master secret is: $\{a, b, z'_1, \ldots, z'_{\widetilde{n}+1}\}$.

Extract: Let ω_u be the attribute set of a user such that $\omega_u \subseteq \mathcal{U}$. The private key of ω_u is constructed as follows:

1. Let f_u be a random polynomial of degree $t - 1$ such that $f_u(0) = a$.
2. For all $i \in \omega_u$, compute $D_i = \frac{g_2^{f_u(i)+z_i}}{h_0^{z_i}}$, with $z_i = ai^{\widetilde{n}} + \sum_{j=1}^{\widetilde{n}+1}(z'_i\Delta_{j,N}(i))$.
3. The private key of a user is $(\{D_i\}_{i\in\omega_u})$.

Sign: Follows the signature algorithm of Scheme 3.1.

Verify: Given a message m and signature (S_1, S_2) and verification attribute set ω_v, a verifier proceeds as follows:

1. Compute $\hat{\rho} = H(m||\omega_v)$ (Note, $\omega_s = \omega_v$).

2. Check if: $\frac{e(g,S_1)e(h_0 \cdot {g_2}^{-1}, \prod_{i \in \omega_v} T(i)^{\Delta_{i,\omega_v}(0)})}{e(S_2, W(\hat{\rho}))} = e(g_1, g_2)$. If true, output 1, else, output 0.

Theorem 3.21. *Under the same assumption about the distribution of the output of H as in Theorem 3.11, any s-EUF-CMA adversary $\mathcal{A}$ against the scheme above can be used to construct algorithms $\mathcal{B}$, and $\mathcal{C}$ that have essentially the same running time as $\mathcal{A}$ and that solve the computational Diffie-Hellman problem, and break the collision-resistance of H respectively, with*

$$Adv_{\mathcal{A}}^{CMA} \leq 8Q(n_m + 1)Adv_{\mathcal{B}}^{CDH} + Adv_{\mathcal{C},H}^{CR},$$

where Q is the number of signature queries made by $\mathcal{A}$ and n_m is the output size of the hash function.

The proof of this theorem is similar to the proof of the small universe scheme, we refer to the full version of this paper for the details.

Theorem 3.22. *The scheme above is private*

The proof is similar to the proof of Theorem 3.12

3.3 (t, n)-ABS Scheme: Threshold Verification

In a (t, n)-ABS [9], a signature is attached with n attributes of the signer, any t of which can be used for verification. This is a useful property that allows verifiers to select any subset of t attributes as the verification set and prove the sender's authenticity with respect to those attributes.

We show here how our (t, t)-ABS can be modified to allow for threshold verification. We only show the modification of our first scheme presented in Section 3.1. A similar approach can be used for the large universe construction. The resulting (t, n)-ABS scheme no longer has constant size signature, but still requires only three pairings for verification.

Setup: Same as in Scheme 3.1.
Extract: Same as in Scheme 3.1.
Sign: Given a signing key sk_{ω_u} and a signing attribute set $\omega_s \subset \omega_u$ with $|\omega_s| \geq t$, the signature of a message $m \in \{0,1\}^*$ is calculated as follows:
1. Sample $c \in_R \mathbb{Z}_p^*$.
2. Compute $\rho = H(m||\omega_s)$ and $\mathcal{S}_i = (D_i)\,(W(\rho))^c$ for each $i \in \omega_s$.
3. The signature is: $(\{\mathcal{S}_i\}_{i \in \omega_s}, g^c)$.

Verify: On receiving a message m, ω_s and signature $(\{S_{1_i}\}_{i \in \omega_s}, S_2)$, a verifier proceeds as follows:
1. Let $\Omega \subset \omega_s$ be a set of attributes of size at least t against which the verifier wishes to verify the signature
2. Compute $\hat{\rho} = H(m||\omega_s)$.
3. Verify the following: $\frac{e\left(g, \prod_{i \in \Omega} S_{1_i}^{\Delta_{i,S}(0)}\right) e\left(h_0 \cdot {g_2}^{-1}, \prod_{i \in \Omega} h_i^{\Delta_{i,S}(0)}\right)}{e(W(\hat{\rho}), S_2)} = e(g_1, g_2)$. If true, output 1, else, output 0.

Theorem 3.31. *Under the same assumption about the distribution of the output of H as in Theorem 3.11, any s-EUF-CMA adversary $\mathcal{A}$ against the (t, n)-ABS scheme can be used to construct algorithms $\mathcal{B}$, and $\mathcal{C}$ that have essentially the same running time as $\mathcal{A}$ and that solves the computational Diffie-Hellman problem and break the collision-resistance of H respectively, with*

$$Adv_{\mathcal{A}}^{CMA} \leq 8Q(n_m + 1)Adv_{\mathcal{B}}^{CDH} + Adv_{\mathcal{C},H}^{CR},$$

where Q is the number of signature queries made by $\mathcal{A}$ and n_m is the output size of the hash function.

The security of the scheme follows from the (t, t)-ABS scheme presented in Section 3.1.

Since this scheme follows a slightly different model from the schemes of the two previous sections, formally proving its signer-attribute privacy would require a new security definition since the signer can now display more attributes than is required by the threshold. However, like in the previous schemes, it should be obvious that all the attributes that the signer decides not to publish in the signature remain secret since the signature is independent from all the attribute which are not present in the signature.

4 Efficiency Analysis

In this section we analyze the efficiency of our signatures schemes using the public parameter size and signature size, both measured in number of group elements, and computations required for verification. In Table 1, n denotes the size of signing attribute set, $\hat{n}$ denotes the size of small attribute universe, $\widetilde{n}$ denotes the maximum size of an attribute set, n_m denotes the length of collision resistant hash function H output, k denotes the threshold of the policy in [5], and d denotes the size of default attribute set as defined in [5]. It can be seen that the all (t, n)-ABS schemes, both ours and the existing ones, have a signature size which is linear in the number of signing attributes. Our (t, t)-ABS schemes, however have signature size that is *constant*, while the (t, t)-schemes that can be derived from the (t, n)-ABS schemes in [9] and [5] have signature size $(2t+1)|\mathbb{G}|$ and $(t+d+2)|\mathbb{G}|$ respectively (note that if the individual signature components can be aggregated then these scheme would also have a constant signature size). The main advantage of all our (t, t)-ABS and (t, n)-ABS construction is that the number of pairing operations in verification and the signature size is independent of the size of attribute sets used in signature generation and verification, whereas all other schemes need a linear number or pairing operations. Further, the signature verification in all ABS except (t, t)-ABS of [5], requires linear number of exponentiations, thus we do not achieve constant pairing at the expense of exponentiation operations.

Table 1. Scheme Efficiency

Type	Schemes	*Params* Size	Signature Size	Pairings to Verify	Exp. to Sign	Exp. to Verify
(t,t)	Scheme 1	$(\hat{n}+n_m+4)$	2	3	$t+1$	t
	Scheme 2	$(\widetilde{n}+n_m+5)$	2	3	$t+1$	t
	[9]	$(\widetilde{n}+4)$	$(2t+1)$ [1]	t	$3t$	t
	[5]	$(\widetilde{n}+d+3)$	$(t+d-k+2)$	t	$(t+2)$	0
(t,n)	Scheme 3	$(\hat{n}+n_m+5)$	$(n+1)$	3	1	$2t$
	[9]	$(\widetilde{n}+4)$	$(2n+1)$ [2]	t	$3t$	t
	[5]	$(\widetilde{n}+d+3)$	$(n+d-k+2)$	t	$(t+2)$	t

5 Extensions

5.1 Reducing Public Parameters Size in Random Oracle Model

The public parameter size of our schemes can be reduced if we can model one of the hash functions as a random oracle. In this case, we replace the function $W : \{0,1\}^{n_m} \rightarrow \mathbb{G}$ used in the schemes presented in Section 3 by a random oracle $\mathcal{H} : \{0,1\}^* \times \mathbb{Z}_p^* \rightarrow \mathbb{G}$ which reduces the public parameter size by n_m group elements.

We show here how to modify our small universe construction, but a similar approach can be applied to both (t,t)-ABS large attribute universe scheme, and (t,n)-ABS. In addition to reducing the public parameter size, the use of the random oracle model allows us to obtain a tighter reduction.

Setup: Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map, and let g be a generator for $\mathbb{G}$. Compute

- Sample $a, b \in_R \mathbb{Z}_p^*$, and set $g_1 = g^a$, $g_2 = g^b$
- $h_0 \in_R \mathbb{G} \setminus \{1_{\mathbb{G}}\}$
- Sample $z_i \in_R \mathbb{Z}_p^*$ for $1 \leq i \leq \hat{n}$ and set $h_j = g^{z_i}$
- $\mathcal{H} : \{0,1\}^* \times \mathbb{Z}_p^* \rightarrow \mathbb{G}^*$ be a hash function. This function is modeled as a random oracle in the proof.

The public parameters are $\{\mathbb{G}, \mathbb{G}_T, e, p, g, g_1, g_2, h_0, \ldots, h_{\hat{n}}, \mathcal{H}\}$.
The master secret is: $\{a, b, z_1, \ldots, z_n\}$.

Extract: Let ω_u be the attribute set of a user such that $\omega_u \subseteq \mathcal{U}$. The private key of ω_u is constructed as follows:

1. Let f_u be a random polynomial of degree $t-1$ such that $f_u(0) = a$.
2. For each $i \in \omega_u$, compute $D_i = \frac{g_2^{f_u(i)+z_i}}{h_0^{z_i}}$

[1] Note that the signature length of the scheme presented in [9] is $3t$. The value $2t+1$ is the reduced signature size if the randomness used in sign is r for t attributes, instead of r_i for $i = 1$ to t.

[2] Note that the signature length of the scheme presented in [9] is $3n$. The value $2n+1$ is the reduced signature size if the randomness used in sign is r for n attributes, instead of r_i for $i = 1$ to n.

3. The private key is $(\{D_i\}_{i\in\omega_u})$.

Sign: Given a signing key sk_{ω_u} and a signing attribute set $\omega_s \subset \omega_u$ with $|\omega_s| = t$, the signature of a message $m \in \{0,1\}^*$ is performed as follows:

1. Sample $c, c_1 \in_R \mathbb{Z}_p^*$.
2. Compute $g_m = \mathcal{H}(c_1||m||\omega_s)$.
3. Compute $\mathcal{S} = {g_m}^c \cdot \prod_{i\in\omega_s} (D_i)^{\Delta_{i,\omega_s}(0)}$.
4. The signature is: $(\mathcal{S}, g^c, c_1)$.

Verify: Given a message m and signature (S_1, S_2, S_3) and verification attribute set ω_v, a verifier proceeds as follows:

1. Compute $g'_m = \mathcal{H}(S_3||m||\omega_v)$ (Note, $\omega_s = \omega_v$).
2. Verify that the following equation holds $\frac{e(g,S_1)e(h_0\cdot {g_2}^{-1}, \prod_{i\in\omega_v} h_i^{\Delta_{i,\omega_v}(0)})}{e(S_2,g'_m)} = e(g_1,g_2)$. If so, output 1, else, output 0.

Theorem 5.11. *Any s-EUF-CMA adversary $\mathcal{A}$ against the scheme above can be used to construct algorithm $\mathcal{B}$ that has essentially the same running time as $\mathcal{A}$ and that solves the computational Diffie-Hellman problem with*

$$Adv_{\mathcal{A}}^{CMA} \leq Adv_{\mathcal{B}}^{CDH} + \frac{Q_S(Q_{\mathcal{H}} + Q_S) + 1}{p},$$

where $Q_{\mathcal{H}}$ and Q_S are the number of random oracle and signing queries made by the adversary $\mathcal{A}$ and p is the size of the cyclic group.

The proof is essentially the same as the proof of Theorem 3.11, except that the function W is replaced by a random oracle. This allows us to 'program' the oracle differently when answering a signing query and a standard oracle query (which will almost never intersect because of the random value c_1 in the signature), thereby obtaining a tight reduction. The factor $(Q_s(Q_{\mathcal{H}} + Q_S) + 1)/p$ comes from the probability that the same value c_1 is used for two signing queries, and the probability that the adversary is able to forge a signature without querying the random oracle for his forgery. We refer to the complete version of this paper for the details.

5.2 Flexible Threshold

We mentioned that we chose to present our scheme with a fixed threshold for simplicity. The scheme can easily be modified to allow for a flexible threshold. This can be done using either or both of the following techniques.

First, the key generation algorithm could give a different threshold to each user by generating a polynomial of a different degree when answering each key generation query. One can find that this generalization does not affect the security proof.

Second, the key generation algorithm could include a number d of dummy attributes in the attribute set of the user. This would allow a user with threshold t to produce a signature with a range of $t - d$ to t attributes.

5.3 Type 2 and Type 3 Pairings

For simplicity of exposition, we used a *type 1* bilinear pairing, that is, a pairing of the form $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We note that it would be easy to modify our schemes so that a type 2 or type 3 pairings (of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) could be used. This would require slightly longer public parameters, as, in addition to the current values, they would have to contain a generator for $\mathbb{G}_2$, $\mathfrak{g}$, as well as values $\mathfrak{g}_2, \mathfrak{u}', \mathfrak{h}_0, \mathfrak{u}_0, \ldots, \mathfrak{u}_{n_m}$ with $\log_{\mathfrak{g}} \mathfrak{g}_2 = \log_g g_2$, $\log_{\mathfrak{g}} \mathfrak{h}_0 = \log_g h_0$, etc. These values would be used to compute the values $\mathfrak{g}$, $\mathfrak{g}_2$, $\mathfrak{h}_0\mathfrak{g}_2^{-1}$ and $W(\rho)$ used in the pairings in the verification algorithm. All other computations would remain in $\mathbb{G}_1$.

6 Conclusion

We presented (t, t)-ABS schemes with constant size signature and constant pairing computation for verification. We also extended these schemes to the general (t, n)-ABS case while maintaining the constant pairing property for verification. Security of all schemes are reduced in the standard model to the well established computational Diffie-Hellman problem. Security of the schemes presented are in s-EUF-CMA model. We also discussed privacy property of our scheme and in particular showed it satisfies a privacy definition adapted from signature-policy ABS.

Interesting questions left open by this paper would be to apply the techniques of Shahandashti and Safavi-Naini [9] to obtain an efficient attribute-based credential system determine if recent techniques for adaptively secure attribute-based encryption and signature ([4,6,7]) could be used to obtain an adaptively secure scheme, and see if it is possible to construct ABS schemes with constant pairings for verification or short signatures for more general policies.

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society, Washington, DC (2007)
2. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM (2006)
3. Lauter, K., Montgomery, P.L., Naehrig, M.: An Analysis of Affine Coordinates for Pairing Computation. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 1–20. Springer, Heidelberg (2010)
4. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)

5. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: ASIACCS 2010: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 60–69. ACM, New York (2010)
6. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-Based Signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (2011)
7. Okamoto, T., Takashima, K.: Efficient Attribute-Based Signatures for Non-monotone Predicates in the Standard Model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 35–52. Springer, Heidelberg (2011)
8. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
9. Shahandashti, S.F., Safavi-Naini, R.: Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 198–216. Springer, Heidelberg (2009)
10. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A Sketch of Proof of the (t, t)-ABS Small Universe Scheme

Let C be the event that $\mathcal{A}$ outputs a forgery $(m^*, (S_1^*, S_2^*))$ such that the hash $H(m^* \| \omega_u^*)$ is the same as one of the hashes produced by the challenger when answering one of the signing queries. We show that there exists an algorithm $\mathcal{C}$ can break the hash function with probability at least $\Pr[C \text{ occurs}]$. Then, given that C does not occurs, we show how an algorithm $\mathcal{B}$ can use a forgery produced by algorithm $\mathcal{A}$ to break the CDH problem. Throughout the reduction, we let Q be the number of signature queries made by adversary $\mathcal{A}$.

Algorithm $\mathcal{B}$
$\mathcal{B}$ is given the computational Diffie-Hellman problem instance $(\mathbb{G}, p, g, A = g^a, B = g^b)$, where $a, b \in_R \mathbb{Z}_p^*$, and $g \in \mathbb{G}$ is the generator. $\mathcal{B}$ runs algorithm $\mathcal{A}$ answering its queries in each phase of the security game as follows:
Commit: $\mathcal{A}$ sends the challenge attribute set (ω_u^*) to $\mathcal{B}$.

Setup: Algorithm $\mathcal{B}$ computes the system parameters as follows:

- $g_1 = A$, $g_2 = B$
- Sample $w_1, w_2, v \in_R \mathbb{Z}_p^*$, and set $h_0 = (g_2(g^{w_2})^{1/vw_1})$
- Sample $\gamma_i \in_R \mathbb{Z}_p^*$ for $i \in \omega_u^*$, and set $h_i = g^{\gamma_i}$
- Sample $\gamma_i, \nu_i \in_R \mathbb{Z}_p^*$ for $i \in \mathcal{U} \setminus \omega_u^*$, and set $h_i = g^{\gamma_i} g_1^{-\nu_i}$
- Sample $k \in_R \{0, \ldots, n_m\}$ $\zeta \in_R \{0, \ldots, 4Q-1\}$ and $\xi \in_R \mathbb{Z}_p^*$, and set $u_0 = g_2^{-4kQ+\zeta} g^{\xi}$
- Sample $\zeta_i \in_R \{0, \ldots, 4Q-1\}$ and $\xi_i \in_R \mathbb{Z}_p^*$ for $i = 1$ to n_m, and set $u_i = g_2^{\zeta_i} g^{\xi_i}$

By setting up this way we have, $W(x) = u_0 \prod_{i=1}^{n_m} u_i^{x[i]} = g_2^{F_u(x)} g^{J_u(x)}$, where

$F_u(x) = -4kQ + \zeta + \sum_{i=1}^{n_m} x[i]\zeta_i \bmod p$ and $J_u(x) = \xi + \sum_{i=1}^{n_m} x[i]\xi_i \bmod p$. To simplify the analysis of the algorithm, we also define the function $K_u(x)$ as

$$K_u(x) = \begin{cases} 0 \text{ if } F_u(x) = 0 \bmod 4Q \\ 1 \text{ otherwise} \end{cases}.$$

Clearly, if $K_u(x) = 1$, then $F_u(x) \neq 0 \bmod p$ because p is much larger than Q. Algorithm $\mathcal{B}$ then gives the public parameters $\{\mathbb{G}, \mathbb{G}_T, e, p, g, g_1, g_2, h_0, \ldots, h_{\hat{n}}, u_0, \ldots, u_{n_m}, H\}$ to $\mathcal{A}$. Notice that from $\mathcal{A}$'s point of view, all parameters have the same distribution as in the real construction.

Query Phase:

Extract: $\mathcal{A}$ queries $\mathcal{B}$ on an attribute set ω_u, such that $|\omega_u \cap \omega_u^*| < t$. The private key is issued as follows:

– Let z_i be the discrete logarithm base g of h_i. Observe that for $i \notin \omega_u^*$, $h_i = g^{\gamma_i - \nu_i a} = g^{z_i}$. Therefore, $z_i = \gamma_i - \nu_i a$. For $i \in \omega_u^*$, $h_i = g^{\gamma_i} = g^{z_i}$, $z_i = \gamma_i$.

– Let Γ, Γ', S be three sets such that, $\Gamma = \omega_u \cap \omega_u^*$ and Γ' be a set such that $\Gamma \subseteq \Gamma' \subseteq \omega_u$ and $|\Gamma'| = t - 1$ and $S = \Gamma' \cup \{0\}$. We will choose a random $t-1$ degree polynomial $f_u(x)$ by randomly choosing $t-1$ points $\lambda_i \in \mathbb{Z}_p$ and setting $f_u(i) = \lambda_i$ for every $i \in \Gamma'$, in addition to having $f_u(0) = a$. The secret key components corresponding to $\omega_u - \Gamma'$ will then be calculated consistently with our choice of $f_u(x)$.

1.1 For all $i \in \Gamma$, randomly sample $\lambda_i \in_R \mathbb{Z}_p^*$ and compute

$$D_i = \frac{g_2^{\lambda_i + \gamma_i}}{h_0^{\gamma_i}} = \frac{g_2^{\lambda_i + z_i}}{h_0^{z_i}}$$

1.2 For all $i \in \Gamma' - \Gamma$, randomly sample $\lambda_i \in_R \mathbb{Z}_p^*$, and compute

$$D_i' = g_2^{\lambda_i + \gamma_i}$$
$$D_i'' = g_1^{-w_2\nu_i/w_1 v}(g_2^{w_1 v} g^{w_2})^{\gamma_i/w_1 v}$$
$$D_i = \frac{D_i'}{D_i''} = \frac{g_2^{\lambda_i + z_i}}{h_0^{z_i}}$$

Due to our choice of λ_i, the polynomial $f_u(x)$ defined by $f_u(i) = \lambda_i$ for $i \in \Gamma'$ and $f_u(0) = a$ is a random polynomial of degree $t-1$ with $f_u(0) = a$.

1.3 For all $i \in \omega_u - \Gamma'$, set $\gamma_i' = \gamma_i - a\Delta_{i,S}(0)$ and compute

$$D_i' = g_2^{\left(\sum_{j \in \Gamma'} \Delta_{j,S}(i)\lambda_j\right) + \gamma_i}$$
$$D_i'' = g_1^{-w_2\nu_i + w_2\Delta_{i,S}(0)/(w_1 v)}(g_2^{w_1 v} g^{w_2})^{\gamma_i'/w_1 v}$$
$$D_i = \frac{D_i'}{D_i''} = \frac{g_2^{f_u(i) + z_i}}{h_0^{z_i}}$$

Sign: $\mathcal{A}$ queries $\mathcal{B}$ on a message m, and signature attribute set ω_s. First, if $|\omega_s \cap \omega_u^*| < t$, then $\mathcal{B}$ can simply use the procedure above to get the secret key corresponding to (ω_s) and sign the message using the **Sign** algorithm. For signature queries on attribute set $\omega_s \subset \omega_u^*, |\omega_s| = t$, $\mathcal{B}$ does the following:

1. Compute $\rho = H(m||\omega_s)$.
2. If $K_u(\rho) = 0$, algorithm $\mathcal{B}$ stops and fails.
3. Sample a $c' \in_R \mathbb{Z}_p^*$ and set $g^c = g^{c' - a/F_u(\rho)}$.

4. Compute: $\mathcal{S} = g_1^{\frac{-J_u(\rho)}{F_u(\rho)}} \cdot W(\rho)^{c'} \cdot \prod_{i \in \omega_s} \left(\frac{g_2^{\gamma_i}}{h_0^{\gamma_i}} \right)^{\Delta_{i,\omega_s}(0)}$
5. The signature is: $\left(\mathcal{S}, g^{c'} g_1^{-1/F_u(\rho)} \right)$.

Forgery: Eventually, $\mathcal{A}$ return a signature (S_1^*, S_2^*) on a message m^* for the challenge attribute set ω_u^*. If $H(m^*||\omega_u^*)$ is equal to the hash obtained while answering one of $\mathcal{A}$'s signing queries or if $F_u(H(m^*||\omega_u^*)) \neq 0$, then $\mathcal{B}$ aborts, else $\mathcal{B}$ computes and outputs: $\frac{S_1^* \prod_{i \in \omega_u^*} h_0^{\gamma_i \Delta_{i,\omega_u^*}(0)}}{(S_2^*)^{J_u(\hat{\rho})} \prod_{i \in \omega_u^*} B^{\gamma_i \Delta_{i,\omega_u^*}(0)}} = g^{ab}$.

Algorithm $\mathcal{C}$

$\mathcal{C}$ is given the description of a hash function H^* sampled at random from the family of collision-resistant hash function used in the scheme.

$\mathcal{C}$ initializes an empty list H_{list}^* and runs algorithm $\mathcal{A}$ answering its queries in each phase of the security game as follows:

Commit: $\mathcal{A}$ sends the challenge attribute set ω_u^* to $\mathcal{C}$

Setup: $\mathcal{C}$ runs the normal ***Setup*** algorithm from the scheme, except that instead of sampling the hash function H at random, it uses the function H^* it received as input.

Query Phase:

Extract: $\mathcal{C}$ answers those queries as in the security game.

Sign: $\mathcal{C}$ answers those queries as in the security game, and for each signing query (m, ω_s), $\mathcal{C}$ adds $(m||\omega_s, H^*(m||\omega_s))$ to H_{list}^*.

Forgery: when $\mathcal{A}$ outputs its attempted forgery (S_1^*, S_2^*) on a message m^*, $\mathcal{C}$ adds $(m^*||\omega_u^*, H^*(m||\omega_u^*))$ to H_{list}^* and then searches H_{list}^* for a collision in the hash function. If it finds one, it outputs it, otherwise, it halts and indicates failure.

Analysis:

Let C be the event that $\mathcal{A}$ outputs a forgery $(m^*, (S_1^*, S_2^*))$ such that the hash $H(m^* \| \omega_u^*)$ is the same as the one produced when answering one of the signing queries, and let $Forge$ be the event that algorithm $\mathcal{A}$ successfully outputs a valid forgery. It is clear that whenever C occurs, algorithm $\mathcal{C}$ successfully finds a collision in his given hash function H^*. Therefore,

$$\begin{aligned} Adv_{\mathcal{A}}^{CMA} &= \mathsf{Pr}[Forge] = \mathsf{Pr}[Forge \wedge C] + \mathsf{Pr}[Forge \wedge \overline{C}] \\ &= Adv_{\mathcal{C},H}^{CR} + \mathsf{Pr}[Forge|\overline{C}]\mathsf{Pr}[\overline{C}] \\ &\leq Adv_{\mathcal{C},H}^{CR} + \mathsf{Pr}[Forge|\overline{C}] \end{aligned}$$

Then, we note that, whenever $\mathcal{B}$ does not abort, its simulation of $\mathcal{A}$'s environment is perfect. Given that no hash collision occurs at the forgery step, $\mathcal{B}$ can only abort when answering a signing query (m, ω_s) if $K_u(H(m\|\omega_s)) = 0$, or at the forgery step if $F_u(H(m^*\|\omega_u^*)) \neq 0$. We denote by ρ_i the hash obtained when answering the i^{th} signing query, and by ρ^* the hash obtained at the forgery step. Using a technique identical to that in Waters' paper [10] (also detailed in the

full version of this paper), we can find that

$$\mathsf{Pr}\left[F_u(\rho^*) = 0 \wedge \left(\bigwedge_{i=1}^{Q} K_u(\rho_i) \neq 0\right)\right] \geq \frac{1}{8Q(n_m+1)}$$

Thus, we have

$$Adv_{\mathcal{B}}^{CDH} \geq \frac{1}{8Qn_m+1}\left(\mathsf{Pr}[Forge|\overline{C}]\right)$$

Hence

$$Adv_{\mathcal{A}}^{CMA} \leq 8Q(n_m+1)Adv_{\mathcal{B}}^{CDH} + Adv_{\mathcal{C},H}^{CR}\square$$

Divisible E-Cash in the Standard Model

Malika Izabachène[1] and Benoît Libert[2,⋆]

[1] LSV, CNRS & ENS Cachan & INRIA Saclay Île de France, France
[2] Université catholique de Louvain, Belgium

Abstract. Off-line e-cash systems are the digital analogue of regular cash. One of the main desirable properties is anonymity: spending a coin should not reveal the identity of the spender and, at the same time, users should not be able to double-spend coins without being detected. Compact e-cash systems make it possible to store a wallet of $O(2^L)$ coins using $O(L + \lambda)$ bits, where λ is the security parameter. They are called *divisible* whenever the user has the flexibility of spending an amount of 2^ℓ, for some $\ell \leq L$, more efficiently than by repeatedly spending individual coins. This paper presents the first construction of divisible e-cash in the standard model (*i.e.*, without the random oracle heuristic). The scheme allows a user to obtain a wallet of 2^L coins by running a withdrawal protocol with the bank. Our construction is built on the traditional binary tree approach, where the wallet is organized in such a way that the monetary value of a coin depends on how deep the coin is in the tree.

Keywords: E-Cash, provable security, anonymity, non-interactive proofs.

1 Introduction

Introduced by Chaum [22,23] and developed in [24,20,25,40,29], electronic cash is the digital equivalent of regular money. It allows a user to withdraw a wallet of electronic coins from a bank so that e-coins can be spent to merchants who can then deposit them back to the bank.

The withdrawal, spending and deposit protocols should be designed in such a way that it is infeasible to determine when a particular coin was spent: even if the bank colludes with the merchant, after the deposit protocol, it should be unable to link a received coin to a particular withdrawal protocol. At the same time, users should not be able to covertly double-spend coins: should a cheating user attempt to spend a given coin twice, his identity must be exposed and evidence of his misbehavior must be given. Ideally, dishonest users should be identified without the help of a trusted third party and, as in the off-line scenario [24], the bank should preferably not intervene in the spending protocol between the user and the merchant.

⋆ This author acknowledges the Belgian Fund for Scientific Research (F.R.S.-F.N.R.S) for his "Collaborateur scientifique" fellowship.

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 314–332, 2013.

Related Work. In 2005, Camenisch, Hohenberger and Lysyanskaya [10] described a *compact* e-cash system allowing a user to withdraw a wallet of 2^L coins with a computational cost of $O(L+\lambda)$, where λ is the security parameter, in the spending and withdrawal protocols. Using appropriate choices [27,28] of verifiable random functions [34], they also showed how to store a wallet using only $O(L+\lambda)$ bits and additionally described a coin tracing mechanism allowing to trace all the coins of a misbehaving user. The protocol of Camenisch *et al.* was subsequently extended into e-cash systems with coin endorsement [13] or transferability properties [15,16].

The aforementioned e-cash realizations all appeal to the random oracle model [6] – at least if the amount of interaction is to be minimized in the spending protocol – which is known not to accurately reflect real world situations (see [19] for instance). To fill this gap, Belenkiy, Chase, Kohlweiss and Lysyanskaya [5] described a compact e-cash system with non-interactive spending in the standard model. Their construction cleverly combines multi-block extensions of P-signatures [3] and simulatable verifiable random functions [21] with the Groth-Sahai non-interactive proof systems [31]. Independently, Fuchsbauer, Pointcheval and Vergnaud also used Groth-Sahai proofs [30] to build a transferable fair (*i.e.*, involving a semi-trusted third party) e-cash system in the standard model. More recently, Blazy *et al.* [7] gave a similar construction with stronger anonymity properties.

Divisible E-Cash. In the constructions of [10], users have to run the spending protocol N times if the amount to be paid is the equivalent of N coins. One possible improvement is to use wallets containing coins of distinct monetary values as in [17]. Unfortunately, this approach does not allow to split individual coins of large value. This problem is addressed by divisible e-cash systems where users can withdraw a coin of value 2^L that can be spent in several times by dividing the value of that coin. Divisible e-cash makes it possible for users to spend the equivalent of $N = 2^\ell$ (with $0 \leq \ell \leq L$) coins more efficiently than by iterating the spending protocol 2^ℓ times. Constructions of divisible e-cash were proposed in the 90's [36,38,26,37,20]. Okamoto provided a practical realization [37] that was subsequently improved in [20]. Unfortunately, these schemes are not fully unlinkable since several spends of a given divisible coin can be linked. To address this concern, Nakanishi and Sugiyama [35] described an unlinkable divisible e-cash system but their scheme requires a trusted third party to uncover the identity of double-spenders. In addition, by colluding with the bank, the merchant can obtain information on which part of the divisible coin the user is spending.

In 2007, Canard and Gouget [14] designed the first divisible e-cash system with full anonymity (and unlinkability) where misbehaving users can be identified without involving a trusted third party. Later on, Au *et al.* [1] came up with a more efficient implementation at the expense of substantially weaker security

guarantees. More recently, Canard and Gouget [18] showed how to improve upon the efficiency of their original proposal without sacrificing the original security.

Our Contribution. Prior implementations of truly anonymous divisible e-cash all require the random oracle idealization in their security analysis. In this paper, we describe the first anonymous divisible e-cash in the standard model. Like the scheme of Belenkiy *et al.* [5], our construction relies on the Groth-Sahai non-interactive proof systems [31].

Our scheme is less efficient than the fastest random-oracle-based scheme [18] in several metrics. While the spending phase has constant (*i.e.*, independent of the value 2^L of the wallet) communication complexity in [18], our spending protocol requires users to transmit $O(L)$ group elements to the merchant in the worst case. On the other hand, due to the use of bounded accumulators [2], the bank has to set up a public key of size $O(2^L)$ in [18][1] whereas we only need the bank to have a key of size $O(1)$.

Achieving divisibility without resorting to random oracles requires to solve several technical issues. The solutions of Canard and Gouget [14,18] associate each wallet with a binary tree – where nodes correspond to expandable amounts – combined with cyclic groups of *distinct* but related orders. Since these techniques do not appear compatible with the Groth-Sahai toolbox, we had to find other techniques to split the wallet across the nodes of a binary tree. In particular, we use a different method to authenticate the node corresponding to the spent divided coin in the tree.

As in the first truly anonymous construction of divisible e-cash [14], the communication complexity of our spending algorithm depends on how much the initial amount 2^L has to be divided: from an initial tree of value 2^L, when a coin of value 2^ℓ has to be spent, the communication cost of the spending phase is $O(L - \ell)$. Hence, the more we want to divide the wallet into small coins, the more expensive the spending phase is.

The downside of our e-cash construction is the complexity of the deposit phase, where the computational workload of the bank depends on the number of previously received coins when it comes to check that the received coin does not constitute a double-spending. Even though the bank can be expected to have significant computational resources (and although the double-spending checks can be performed in parallel by clerks testing a subset of previously spent coins each), this would be a real bottleneck in practice. For this reason, our system is not meant to be a practical solution and should only be seen as a feasibility result. We leave it as an open problem to build such a system with a more efficient deposit procedure from the bank's standpoint: as in previous constructions of compact e-cash (e.g. [10,5]), the bank should only have to look up the coin's serial number in a table of previously spent coins. It would also be interesting to reduce the communication complexity of the spending phase so as to only transmit a constant number of group elements.

[1] The reason is that, in all known bounded accumulators, the public key has linear size in the maximal number of accumulated values.

2 Background and Definitions

2.1 Definitions for Divisible E-Cash

An e-cash scheme involves a bank $\mathcal{B}$, many users $\mathcal{U}$ and merchants $\mathcal{M}$ (who can be viewed as special users). All these parties interact together with respect to the following protocols:

CashSetup(λ)**:** takes as input a security parameter λ and outputs the public parameters params.

BankKG(params, L)**:** generates bank's public and secret parameters $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$ that allow $\mathcal{B}$ to issue wallets of value 2^L (we assume that L is part of $pk_{\mathcal{B}}$). It also defines an empty database $\mathrm{DB}_{\mathcal{B}}$ for later use.

UserKG(params)**:** generates a user key pair $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$. We denote as $\mathcal{H}_{\mathcal{U}}$ the set of honestly generated public keys.

Withdraw$\big(\mathcal{U}(\mathsf{params}, pk_{\mathcal{B}}, sk_{\mathcal{U}}), \mathcal{B}(\mathsf{params}, pk_{\mathcal{U}}, sk_{\mathcal{B}})\big)$**:** is an interactive protocol between a user $\mathcal{U}$ and the bank $\mathcal{B}$ that allows an honest user to obtain a coin of value 2^L. The wallet $\mathcal{W}$ comprises the coins, the user's secret key, a signature from the bank on it and some state information state. The bank debits $\mathcal{U}$'s account and stores a piece of tracing information $\mathsf{T}_{\mathcal{W}}$ in a database T that allows uncovering the identity of double-spenders.

Spend$\big(\mathsf{params}, pk_{\mathcal{B}}, \mathcal{W}, 2^{\ell}, pk_{\mathcal{M}}, \mathtt{info}\big)$**:** is invoked by $\mathcal{U}$ to spend a coin of value 2^{ℓ} from his wallet and generates a proof Π that the coin is valid. The output is the coin that includes the proof Π and some fresh public information `info` specifying the transaction.

VerifyCoin$\big(\mathsf{params}, pk_{\mathcal{M}}, pk_{\mathcal{B}}, coin, v = 2^{\ell}\big)$**:** allows $\mathcal{M}$ to check the validity of a given coin and outputs a bit depending on whether the test is successful.

Deposit$\big(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, 2^{\ell}, \mathrm{DB}_{\mathcal{B}}\big)$**:** $\mathcal{B}$ updates the database $\mathrm{DB}_{\mathcal{B}}$ with $\{(coin, \mathsf{flag}, 2^{\ell})\}$ where flag indicates whether *coin* is a valid coin of value 2^{ℓ} and whether a cheating attempt is detected.

- If *coin* does not verify, $\mathcal{B}$ rejects it and sets $\mathsf{flag} =$ "$\mathcal{M}$" to indicate a cheating merchant.
- If *coin* verifies, $\mathcal{B}$ runs a double spending detection algorithm, using the database $\mathrm{DB}_{\mathcal{B}}$ containing already received coins. If an overspent is detected, the bank sets $\mathsf{flag} =$ "$\mathcal{U}$", outputs the two coins and reports the double-spending.
- If the coin passes all the tests, the bank accepts the coin, sets $\mathsf{flag} =$ "accept" and credits the merchant's account.

Identify$\big(\mathsf{params}, pk_{\mathcal{B}}, coin_a, coin_b\big)$**:** the bank retrieves the double-spender's public key $pk_{\mathcal{U}}$ using its database $\mathrm{DB}_{\mathcal{B}}$ and the two different coins.

The security model builds on the model of non-interactive compact e-cash from [5]. An e-cash scheme is secure if it provides *Correctness, Anonymity, Balance, Identification* and *Exculpability* simultaneously.

Anonymity. Unlike the model of [14], ours adopts a simulation-based formulation of anonymity (note that simulation-based definitions are often stronger than

indistinguishability-based ones). No coalition of banks and merchants should distinguish a real execution of the Spend protocol from a simulated one. In the security experiment, the adversary is allowed to obtain users' public keys, withdraw and spend coins using the oracles $\mathcal{Q}_{\mathsf{GetKey}}, \mathcal{Q}_{\mathsf{withdraw}}, \mathcal{Q}_{\mathsf{Spend}}$, respectively, which are defined below. Formally, an e-cash system is *anonymous* if there exists a simulator $(\mathsf{SimCashSetup}, \mathsf{SimSpend})$ such that, for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there is a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ such that:

$$
\begin{aligned}
&\big| \Pr\big[\mathsf{params} \leftarrow \mathsf{CashSetup}(\lambda); (pk_{\mathcal{B}}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}) : \\
&\qquad \mathcal{A}_2^{\mathcal{Q}_{\mathsf{Spend}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot), \mathcal{Q}_{\mathsf{GetKey}}(\mathsf{params}, \cdot), \mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot)}(\mathsf{state}) = 1\big] \\
&\quad - \Pr[(\mathsf{params}, \mathsf{Sim}) \leftarrow \mathsf{SimCashSetup}(\lambda); (pk_{\mathcal{B}}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}) : \\
&\qquad \mathcal{A}_2^{\mathcal{Q}_{\mathsf{SimSpend}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot), \mathcal{Q}_{\mathsf{GetKey}}(\mathsf{params}, \cdot), \mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot)}(\mathsf{state}) = 1] \big| < \mathsf{negl}(\lambda)
\end{aligned}
$$

To formalize security against coalition of users, bank and merchants, the anonymity game allows the adversary to generate the bank's public key. It is granted dynamic access to the list of oracles hereafter and has to decide whether it is playing the real game, where the spending oracle is an actual oracle, or the simulation, where the spending oracle is a simulator.

- $\mathcal{Q}_{\mathsf{GetKey}}(\mathsf{params}, j)$: outputs $pk_{\mathcal{U}_j}$. If $\mathcal{U}_j$ does not exist, the oracle generates $(sk_{\mathcal{U}_j}, pk_{\mathcal{U}_j}) \leftarrow \mathsf{UserKG}(\mathsf{params})$ and outputs $pk_{\mathcal{U}_j}$.
- $\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{B}}, j, f)$: given a wallet identifier f, this oracle plays the role of user j – and creates the key pair $(sk_{\mathcal{U}_j}, pk_{\mathcal{U}_j})$ if it does not exist yet – in an execution of $\mathsf{Withdraw}\big(\mathcal{U}(\mathsf{params}, pk_{\mathcal{B}}, sk_{\mathcal{U}_j}), \mathcal{A}(\mathtt{states})\big)$, while the adversary $\mathcal{A}$ plays the role of the bank. The oracle then creates a wallet $\mathcal{W}_f$ of value 2^i.
- $\mathcal{Q}_{\mathsf{Spend}}\big(\mathsf{params}, pk_{\mathcal{B}}, f, v = 2^\ell, pk_{\mathcal{M}}, \mathtt{info}\big)$: the oracle firstly checks if wallet $\mathcal{W}_f$ has been created via an invocation of $\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{B}}, j, f)$. If not, the oracle outputs $\perp$. Otherwise, if $\mathcal{W}_f$ contains a sufficient amount, $\mathcal{Q}_{\mathsf{Spend}}$ runs $\mathsf{Spend}\big(\mathsf{params}, pk_{\mathcal{B}}, \mathcal{W}_f, i, v = 2^\ell, pk_{\mathcal{M}}, \mathtt{info}\big)$ and outputs a coin of value v from the wallet $\mathcal{W}_f$. In any other case (e.g. if the expandable amount of $\mathcal{W}_f$ is less than 2^ℓ), it outputs $\perp$.
- $\mathcal{Q}_{\mathsf{SimSpend}}\big(\mathsf{params}, pk_{\mathcal{B}}, f, v = 2^\ell, pk_{\mathcal{M}}, \mathtt{info}\big)$: if f is not the index of a valid withdrawn wallet obtained from $\mathcal{Q}_{\mathsf{withdraw}}$, $\mathcal{Q}_{\mathsf{SimSpend}}$ outputs $\perp$. Otherwise, it runs a simulator $\mathsf{SimSpend}$ on input of $\big(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, v = 2^\ell, \mathtt{info}\big)$. Note that $\mathsf{SimSpend}$ does not use the user's wallet or his public key.

Balance. No coalition of users can spend more coins than they withdrew. The adversary is a user and can withdraw or spend coins via oracles defined below. An e-cash system provides the *Balance* property if, for any adversary, every value $L \in \mathsf{poly}(\lambda)$, we have:

$$
\begin{aligned}
&\Pr\big[\mathsf{params} \leftarrow \mathsf{CashSetup}(\lambda); (pk_{\mathcal{B}}, sk_{\mathcal{B}}) \leftarrow \mathsf{BankKG}(\mathsf{params}, L); \\
&\quad (q_w, n_d) \leftarrow \mathcal{A}^{\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, \cdot, pk_{\mathcal{B}}, \cdot)(), \mathcal{Q}_{\mathsf{deposit}}(\mathsf{params}, pk_{\mathcal{B}}, \mathrm{DB}_{\mathcal{B}})} : q_w \cdot 2^L < n_d\big] < \mathsf{negl}(\lambda),
\end{aligned}
$$

where n_d is the total amount of deposited money after q_d successful calls to oracle $\mathcal{Q}_{\mathsf{deposit}}$ (by successful, we mean that the oracle sets $\mathsf{flag} = \text{“}\mathsf{accept}\text{”}$), q_w is the number of successful calls to Q_{Withdraw}.

- $\mathcal{Q}_{\mathsf{withdraw}}\big(\mathsf{params}, pk_{\mathcal{U}}, sk_{\mathcal{B}}\big)$: the oracle plays the role of the bank in an execution of the $\mathsf{Withdraw}$ protocol, on input $\big(\mathcal{A}(\mathtt{states}), \mathcal{B}(\mathsf{params}, pk_{\mathcal{U}}, sk_{\mathcal{B}})\big)$, in interaction with the adversary acting as a cheating user. At the end of the protocol, $\mathcal{Q}_{\mathsf{withdraw}}$ stores a piece of tracing information T_w in a database T.
- $\mathcal{Q}_{\mathsf{deposit}}\big(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, v, \mathrm{DB}_{\mathcal{B}}\big)$: this oracle plays the role of the bank while the adversary plays the role of the merchant in the protocol. The oracle initializes the bank database $\mathrm{DB}_{\mathcal{B}}$ at $\emptyset$ and returns the same response as $\mathsf{Deposit}\big(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, v, \mathrm{DB}_{\mathcal{B}}\big)$.

Identification. Given two fraudulent but well-formed coins, the bank should identify the double-spender. This property is defined via an experiment where the adversary $\mathcal{A}$ is the double-spender and has access to a $\mathcal{Q}_{\mathsf{withdraw}}$ oracle defined hereafter. Its goal is to deposit a coin twice without being identified by the bank. We denote by $coin_a$ and $coin_b$ the two coins produced by $\mathcal{A}$. Their corresponding entries in $\mathrm{DB}_{\mathcal{B}}$ are of the form $(coin_a, \mathsf{flag}_a, v_a, pk_{\mathcal{M}_a})$ and $(coin_b, \mathsf{flag}_b, v_b, pk_{\mathcal{M}_b})$ with $coin_a = (\mathtt{info}_a; *)$ and $coin_b = (\mathtt{info}_b; *)$, respectively. We also define a predicate $\mathsf{SameCoin}$ that given two coins $coin_a$ and $coin_b$ and their respective values v_a and v_b, outputs 1 if the bank detects a double-spending during the deposit of $coin_a$ and $coin_b$: in the context of divisible e-cash, it means that either $coin_a$ and $coin_b$ are the same coin or that one of them, say $coin_a$, is the result of dividing the other one (and thus v_a divides v_b). The adversary is successful if its coins satisfy $\mathsf{SameCoin}(coin_a, coin_b, v_a, v_b) = 1$ but the bank fails to identify the user using the database T of tracing pieces of information that were collected during executions of $\mathcal{Q}_{\mathsf{withdraw}}$. An e-cash scheme provides double-spenders identification if for any adversary $\mathcal{A}$ and any $L \in \mathsf{poly}(\lambda)$,

$$
\begin{aligned}
\Pr\big[&\mathsf{params} \leftarrow \mathsf{CashSetup}(\lambda); (pk_{\mathcal{B}}, sk_{\mathcal{B}}) \leftarrow \mathsf{BankKG}(\mathsf{params}, L);\\
&\big((coin_a, v_a), (coin_b, v_b)\big) \leftarrow \mathcal{A}^{\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, \cdot, sk_{\mathcal{B}}, \cdot)}(\mathsf{params}, pk_{\mathcal{B}}) :\\
&(\mathtt{info}_a; pk_{\mathcal{M}_a}) \neq (\mathtt{info}_b; pk_{\mathcal{M}_b}) \wedge\ \mathsf{SameCoin}(coin_a, coin_b, v_a, v_b) = 1\\
&\wedge\ \mathsf{VerifyCoin}(\mathsf{params}, pk_{\mathcal{M}_t}, pk_{\mathcal{B}}, coin_t, v_t) = 1 \text{ for } t \in \{a, b\}\\
&\wedge\ \mathsf{Identify}(\mathsf{params}, pk_{\mathcal{B}}, coin_a, coin_b) \notin \mathsf{T}\big] < \mathsf{negl}(\lambda)
\end{aligned}
$$

The oracle $\mathcal{Q}_{\mathsf{withdraw}}$ has the same specification as in the Balance property.

Exculpability. No coalition of bank and merchants interacting with an honest user $\mathcal{U}$ should be able to produce two coins $(coin_a, coin_b)$ such that $\mathsf{Identify}(\mathsf{params}, pk_{\mathcal{B}}, coin_a, coin_b) = pk_{\mathcal{U}}$ while user $\mathcal{U}$ never double-spent. More formally, we define a game with the challenger playing the role of an honest user and the adversary playing the role of the bank and merchants. The adversary $\mathcal{A}$ is given access to oracles $\mathcal{Q}_{\mathsf{GetKey}}, \mathcal{Q}_{\mathsf{withdraw}}, \mathcal{Q}_{\mathsf{Spend}}$ that allow it to obtain users' keys, create wallets and spend coins. The exculpability property holds if, for any

PPT adversary $\mathcal{A}$, we have

$$\Pr\Big[\mathsf{params} \leftarrow \mathsf{CashSetup}(\lambda);\ (pk_{\mathcal{B}}, st) \leftarrow \mathcal{A}(\mathsf{params});$$
$$(pk_{\mathcal{B}}, coin_a, coin_b, v_a, v_b)$$
$$\leftarrow \mathcal{A}^{\mathcal{Q}_{\mathsf{Spend}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot), \mathcal{Q}_{\mathsf{GetKey}}(\mathsf{params}, \cdot), \mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, \cdot, \cdot, \cdot)}(\mathsf{params}, st);$$
$$\mathsf{SameCoin}(coin_a, coin_b, v_a, v_b) = 1;$$
$$pk_{\mathcal{U}} \leftarrow \mathsf{Identify}(\mathsf{params}, coin_a, coin_b) : pk_{\mathcal{U}} \in \mathcal{H}_{\mathcal{U}}\Big] < \mathsf{negl}(\lambda),$$

where $\mathcal{H}_{\mathcal{U}}$ denotes the set of honest users. Oracles $\mathcal{Q}_{\mathsf{GetKey}}$, $\mathcal{Q}_{\mathsf{withdraw}}$ and $\mathcal{Q}_{\mathsf{Spend}}$ are defined exactly as in the notion of anonymity.

2.2 F-Unforgeable Signatures

Since the e-cash construction described in the paper relies on a common reference string, the following algorithms all take as input a set of common public parameters $\mathsf{params}_{\mathsf{GS}}$. To lighten notations, we omit to explicitly write them in the syntax hereafter.

Definition 1. *A multi-block signature scheme consists of efficient algorithms* $\Sigma = (\mathsf{SigSetup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *with the following specification.*

SigSetup(λ)**:** *takes as input a security parameter* $\lambda \in \mathbb{N}$ *and outputs* params *that gives the length* $n \in \mathsf{poly}(\lambda)$ *of message vectors to be signed.*

Keygen(params)**:** *takes as input the public parameters and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$.

Sign$(\mathsf{sk}, \vec{m})$**:** *is a (possibly randomized) algorithm that takes as input a private key* sk *and a vector* $\vec{m} = (m_1, \ldots, m_n)$ *of messages. It outputs a signature* σ.

Verify$(\mathsf{pk}, \vec{m}, \sigma)$**:** *is a deterministic algorithm that takes as input a public key* pk*, a signature* σ *and a message vector* $\vec{m} = (m_1, \ldots, m_n)$*. It outputs* 1 *if* σ *is deemed valid for* $\vec{m}$ *and* 0 *otherwise.*

Definition 2 ([5]). *A multi-block signature scheme* Σ *is F-unforgeable, for some injective function* $F(.)$*, if no probabilistic polynomial time (PPT) adversary has non-negligible advantage in the following game:*

1. *The challenger runs* Setup *and* Keygen *to obtain a pair* $(\mathsf{pk}, \mathsf{sk})$*, it then sends* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *adaptively queries a signing oracle. At each query* i*,* $\mathcal{A}$ *chooses a vector* $\vec{m} = (m_1, \ldots, m_n)$ *and obtains* $\sigma_i = \mathsf{Sign}(\mathsf{sk}, \vec{m})$.
3. *The adversary* $\mathcal{A}$ *outputs a pair* $\big((F(m_1^\star), \ldots, F(m_n^\star)), \sigma^\star\big)$ *and wins if: (a)* $\mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1$*; (b)* $\mathcal{A}$ *did not obtain any signature on the vector* $(m_1^\star, \ldots, m_n^\star)$.

Definition 3 ([5]). *A multi-block P-signature combines an F-unforgeable multi-block signature scheme* Σ *with a commitment scheme* $(\mathsf{Com}, \mathsf{Open})$ *and three protocols:*

1. *An algorithm* $\mathsf{SigProve}(\mathsf{params}, \mathsf{pk}, \sigma, \vec{m} = (m_1, \ldots, m_n))$ *that generates a series of* n *commitments* C_σ, $C_{m_1}, \ldots, C_{m_n}$ *and a NIZK proof*

$$\pi \leftarrow \mathsf{NIZPK}\big(m_1 \text{ in } C_{m_1}, \ldots, m_n \text{ in } C_{m_n},\ \sigma \text{ in } C_\sigma \mid \{(F(m_1), \ldots, F(m_n), \sigma) : \mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) = 1\}\big)$$

and the corresponding $\mathsf{VerifyProof}(\mathsf{params}, \mathsf{pk}, C_{m_1}, \ldots, C_{m_n}, C_\sigma)$ *algorithm.*
2. *A NIZK proof that two commitments open to the same value, i.e., a proof for the relation*

$$R = \{((x, y), (open_x, open_y)) \mid C = \mathsf{Com}(x, open_x) \ \wedge\ D = \mathsf{Com}(y, open_y) \ \wedge\ x = y\}.$$

3. *A protocol* $\mathsf{SigIssue} \rightleftarrows \mathsf{SigObtain}$ *allowing a user to obtain a signature on a committed vector* $\vec{m} = (m_1, \ldots, m_n)$ *without letting the signer learn any information on* $\vec{m}$.

2.3 Bilinear Maps and Complexity Assumptions

We consider a configuration of *asymmetric* bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p with a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that: (1) $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G}_1 \times \mathbb{G}_2$ and $a, b \in \mathbb{Z}$; (2) $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g \neq 1_{\mathbb{G}_1}$ and $h \neq 1_{\mathbb{G}_2}$. Since we rely on the hardness of DDH in $\mathbb{G}_1$ and $\mathbb{G}_2$, we additionally require that there is no efficiently computable isomorphism between $\mathbb{G}_2$ and $\mathbb{G}_1$.

Definition 4. *The* q**-Hidden Strong Diffie-Hellman problem** *(q-HSDH) in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is, given* $(g, u, h, \Omega = h^\omega) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$ *and tuples* $(g^{1/(\omega+c_i)}, g^{c_i}, h^{c_i}, u^{c_i})$ *with* $c_1, \ldots, c_q \xleftarrow{R} \mathbb{Z}_p^*$*, finding* $(g^{1/(\omega+c)}, h^c, u^c)$ *such that* $c \neq c_i$ *for* $i = 1, \ldots, q$.

Definition 5. *The* q**-Decision Diffie-Hellman Inversion problem** *(q-DDHI) in* $(\mathbb{G}_1, \mathbb{G}_2)$ *consists in, given* $(g, g^{(\alpha)}, \ldots, g^{(\alpha^q)}) \in \mathbb{G}_1^{q+1}$ *and* $\eta \in \mathbb{G}_1$*, deciding if* $\eta = g^{1/\alpha}$ *or* $\eta \in_R \mathbb{G}_1$.

Definition 6 ([3]). *The* **Triple Diffie-Hellman problem** *(TDH) in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is, given a tuple* $(g, g^a, g^b, h, h^a) \in \mathbb{G}_1^3 \times \mathbb{G}_2^2$*, and pairs* $(c_i, g^{1/a+c_i})_{i=1,\ldots,q}$ *where* $a, b, c_1, \ldots, c_q \xleftarrow{R} \mathbb{Z}_p^*$*, to find a triple* $(g^{\mu b}, h^{\mu a}, g^{\mu ab})$ *such that* $\mu \neq 0$.

Definition 7. *The* **Decision 3-party Diffie-Hellman problem** *(D3DH) in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is, given elements* $(g, g^a, g^b, g^c, h, h^a, h^b, h^c, \Gamma) \in \mathbb{G}_1^4 \times \mathbb{G}_2^4 \times \mathbb{G}_1$*, where* $a, b, c \xleftarrow{R} \mathbb{Z}_p$*, to decide if* $\Gamma = g^{abc}$ *or* $\Gamma \in_R \mathbb{G}_1$.

2.4 Building Blocks

Non-interactive Witness Indistinguishable Proofs. Our construction uses Groth-Sahai proofs for pairing product equations (PPE) of the form:

$$\prod_{j=1}^{n} e(\mathcal{A}_j, \mathcal{Y}_j) \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{B}_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{Y}_j)^{\gamma_{i,j}} = t_T,$$

where $\mathcal{X}_i, \mathcal{Y}_j$ are variables in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $\mathcal{A}_j \in \mathbb{G}_1, \mathcal{B}_i \in \mathbb{G}_2$ and $t_T \in \mathbb{G}_T$ are constants for $i \in [1, m]$ and $j \in [1, n]$.

A proof system is a tuple of four algorithms $(\mathsf{Setup}_{\mathsf{GS}}, \mathsf{Prove}_{\mathsf{GS}}, \mathsf{VerifyProof}_{\mathsf{GS}})$: $\mathsf{Setup}_{\mathsf{GS}}$ outputs a common reference string (CRS) crs, $\mathsf{Prove}_{\mathsf{GS}}$ first generates commitments of variables and constructs proofs that these variables satisfy the statement, and $\mathsf{VerifyProof}_{\mathsf{GS}}$ verifies the proof. GS proofs are witness-indistinguishable and some of these can be made zero-knowledge as shown later. The proofs satisfy correctness, soundness and witness-indistinguishability. *Correctness* requires that a verifier always accepts honestly generated proofs for true statements. *Soundness* guarantees that cheating provers can only prove true statements. *Witness-indistinguishability* requires that an efficient simulator $\mathsf{GSSimSetup}$ should be able to produce a common reference string (CRS) crs' that is computationally indistinguishable from a normal crs. When commitments are computed using crs', they are perfectly hiding and the corresponding non-interactive proofs are witness indistinguishable: *i.e.*, they leak no information on the underlying witnesses. *Zero-knowledge* additionally requires the existence of an algorithm $\mathsf{GSSimProve}$ that, given a simulated CRS crs' and some trapdoor information τ, generates a simulated proof of the statement without using the witnesses and in such a way that the proof is indistinguishable from a real proof.

As a building block, we will use a NIZK proof of equality of committed group elements as defined in [3,5].

If $C_x = \mathsf{GSCom}(x, open_x)$ and $C_y = \mathsf{GSCom}(y, open_y)$ are Groth-Sahai commitments to the group element $x = y \in \mathbb{G}_1$, the NIZK proof can be a proof that committed variables $(x, y, \theta) \in \mathbb{G}_1^2 \times \mathbb{Z}_p$ satisfy the equations $e(x/y, h^\theta) = 1$ and $e(g, h^\theta)e(1/g, h) = 1_{\mathbb{G}_T}$. Using the trapdoor of the CRS, we can trapdoor open to 1 a commitment to 0 and generate fake proofs for the latter relation. Setting $\theta = 0$ and $\theta = 1$, respectively, allows to construct a valid (simulated) witness for each of the two equations. Under the SXDH assumption, commitments cost 2 elements in the group. Thus, if the commitment to y is in $\mathbb{G}_1^2$, the proof above costs 8 elements $\mathbb{G}_1$ and 6 in $\mathbb{G}_2$, 6 multi-exponentiations and 26 pairings (to verify). This includes commitments to $y \in \mathbb{Z}_p$ and h^θ.

Multi Block P-signatures. In [5], a multi-block P-signature was proved F-secure under the HSDH and the TDH assumptions. Let $(p, \mathbb{G}_1, \mathbb{G}_2, G_T, e, g, h)$ be parameters for a bilinear map, the public parameters are then defined as $(p, \mathbb{G}_1, \mathbb{G}_2, G_T, e, g, h, \mathsf{params}_{\mathsf{GS}}, e(g, h))$, where g and h are random elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The public key and the private key are defined to be $\mathsf{pk} = (u, U = g^\beta, \tilde{U} = h^\beta, \{V_i = g^{a_i}, \tilde{V}_i = h^{a_i}\}_{i=1}^n)$ and $\mathsf{sk} = (\beta, \vec{a} = (a_1, \ldots, a_n))$, where $u \xleftarrow{R} \mathbb{G}_1$ and for random scalars $\beta, a_1, \ldots, a_n$. To sign a vector of message $\vec{m} = (m_1, \ldots, m_n)$, the signer chooses $r \xleftarrow{R} \mathbb{Z}_p$ such that $r \neq -(\beta + \sum_{i=1}^n a_i m_i)$ and computes $\sigma = (g^{1/\beta + r + \sum_{i=1}^n a_i m_i}, h^r, u^r)$. Verification of a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ on some block $\vec{m}$ is done by checking whether

$$e\big(\sigma_1, \tilde{U} \cdot \sigma_2 \cdot \prod_{i=1}^{n} \tilde{V}_i^{m_i}\big) = e(g, h) \qquad \text{and} \qquad e(u, \sigma_2) = e(\sigma_3, h).$$

As shown in [5], the above scheme can be augmented with the following P-signature protocols.

SigProve$(\mathsf{params}, \mathsf{pk}, \sigma, \vec{m})$**:** parse the signature σ as $(\sigma_1, \sigma_2, \sigma_3)$ and the vector $\vec{m}$ as $(m_1, \ldots, m_n)$. To commit to an exponent $m_i \in \mathbb{Z}_p$, compute Groth-Sahai commitments of h^{m_i} and u^{m_i} as

$$\begin{aligned}(C_{i,1}, C_{i,2}, C_{i,3}) &= \mathsf{Com}\big(m_i, (open_{m_i,1}, open_{m_i,2}, open_{m_i,3})\big)\\ &= \big(\mathsf{GSCom}(h^{m_i}, open_{m_i,1}),\\ &\qquad \mathsf{GSCom}(u^{m_i}, open_{m_i,2}), \mathsf{GSCom}(\tilde{V}_i^{m_i}, open_{m_i,3})\big).\end{aligned}$$

Generate an auxiliary variable $\theta = 1 \in \mathbb{Z}_p$ with its own commitment $C_\theta = \mathsf{GSCom}(\theta, open_\theta)$. Then, generate commitments $\{C_{\sigma_\tau}\}_{\tau=1}^3$ to $\{\sigma_\tau\}_{\tau=1}^3$ and give a NIZK proof that

$$\begin{aligned}&e(g^\theta, h) = e\big(\sigma_1, \tilde{U} \cdot \sigma_2 \cdot \prod_{i=1}^{n} \tilde{V}_i^{m_i}\big),\\ &e(u, \sigma_2) = e(\sigma_3, h),\\ &\qquad \theta = 1\\ &e(g, \tilde{V}_i^{m_i}) = e(V_i, h^{m_i}), \qquad e(u, h^{m_i}) = e(u^{m_i}, h) \qquad \text{for } i \in \{1, \ldots, n\}\end{aligned}$$

We denote the complete proof by

$$\pi^{sig} = \big(\{C_{\sigma_\tau}\}_{\tau=1}^3, \pi_1^{sig}, \pi_2^{sig}, \pi_\theta^{sig}, \{\pi_{m_i,1}^{sig}, \pi_{m_i,2}^{sig}\}_{i=1}^n\big).$$

Note that π_1^{sig} is a proof for a quadratic equation and requires 4 elements of $\mathbb{G}_1$ and 4 elements of $\mathbb{G}_2$. Other equations are linear: each of π_2^{sig} and $\{\pi_{m_i,2}^{sig}\}_{i=1}^n$ demands 2 elements of $\mathbb{G}_1$ and 2 elements of $\mathbb{G}_2$ whereas proofs $\{\pi_{m_i,1}^{sig}\}_{i=1}^n$ only takes two elements of $\mathbb{G}_1$ each since all variables are in $\mathbb{G}_2$. The NIZK property stems from the fact that, on a simulated CRS, a commitment to 0 can be trapdoor opened to 1. For this reason, except for the equation $\theta = 1$ (for which one can simply equivocate the commitment), all other proofs can be simulated using the witnesses $1_{\mathbb{G}_1}$, $1_{\mathbb{G}_2}$ and $\theta = 0$.

VerifyProof$(\mathsf{params}, \mathsf{pk}, \pi^{sig}, (C_1, \ldots, C_n))$**:** works in the obvious way and returns 1 if and only if the proof π^{sig} generated by SigProve is convincing.

EqComProve$(\mathsf{params}, \mathsf{pk}, x, y)$**:** the protocol for proving that two commitments open to the same value employ the usual technique already used in [3,5] and is reviewed section 2.4.

SigIssue$(\mathsf{sk}, (C_1, \ldots, C_n)) \leftrightarrows$ **SigObtain**$(\mathsf{params}, \mathsf{pk}, \vec{m}, \{(C_i, open_i)\}_{i=1}^n)$**:** is a secure two-party protocol between the issuer and the receiver where the latter obtains a signature on a committed vector of messages. As suggested in [5], this can be done using the 2-party protocol of [32] for computing a circuit on committed inputs. Another option would be to use the two-party computation protocol from [4] that relies on homomorphic encryption.

Theorem 1 ([5]). *If the HSDH and the TDH assumptions hold in* $(\mathbb{G}_1, \mathbb{G}_2)$*, the scheme is F-unforgeable w.r.t. the injective function* $F(m) = (h^m, u^m)$.

3 Construction of Divisible E-cash

Known approaches for divisible e-cash [37,35,15,18] make use of a binary tree with $L+1$ levels (for a monetary value of 2^L) where each node corresponds to an amount of money which is exactly one of half the amount of its father. Double-spenders are detected by making sure that each user cannot spend a coin corresponding to a node and one of its descendants or one of its ancestors.

In these tree-based constructions, one difficulty is for the user to efficiently prove that the path connecting the spent node to the root is well-formed. In [14,18], this problem is solved using groups of distinct order: [14] uses a sequence of $L+1$ groups $\mathbb{G}_1, \ldots, \mathbb{G}_{L+1}$ of prime order p_ν where $\mathbb{G}_\nu$ is a subgroup of $\mathbb{Z}^\star_{p_{\nu-1}}$ for $\nu = 1, \ldots, L+1$. The solution of [18] uses $L+2$ bounded accumulators (one for each level of the tree and one for the whole tree) so as to only use two distinct group orders. The use of groups of distinct order (and double discrete logarithms in [14]) is hardly compatible with Groth-Sahai proofs and, in our system we need to find a different technique to prevent users from spending coins associated with a node and one of its ancestors in the same tree.

3.1 General Description of the Scheme

Our construction uses the tree-based approach. Each wallet $\mathcal{W}$ consists of a divisible coin of value 2^L, for some $L \in \mathbb{N}$, and the complexity of the spending phase depends on the depth of the node in the tree $\mathcal{W}$: the deeper the node is, the more expensive the spending phase will be. When an honest user $\mathcal{U}$ with key pairs $(pk_\mathcal{U}, sk_\mathcal{U})$ interacts with the bank $\mathcal{B}$, he obtains a wallet $\mathcal{W} = (s, t, sk_\mathcal{U}, \sigma, \mathsf{state})$ consisting of the bank's signature σ on the vector $(s, t, sk_\mathcal{U})$ where s, t are seeds for the Dodis-Yampolskiy PRF [28]. In our notation, state is a variable indicating the availability of coins.

To spend a coin of value $v = 2^\ell$ (with $\ell \leq L$) in the tree, the user $\mathcal{U}$ determines the next node corresponding to an unspent coin at height ℓ: the root of the tree is used if the user wants to spend his entire wallet at once whereas the leaves correspond to the smallest expandable amounts. Each node will be assigned a unique label consisting of an integer in the interval $[1, 2^{L+1} - 1]$. A simple assignment is obtained by labeling the root as $x_0 = 1$ and the rightmost leaf as

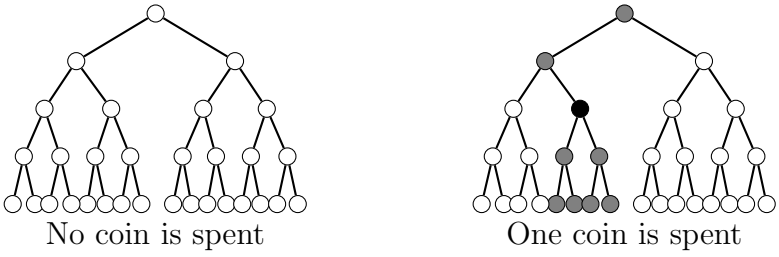


Fig. 1. Binary tree for spending one coin in a wallet of 2^4 coins

$2^{L+1}-1$, all other nodes being considered, in order of appearance, from the left to the right and starting from the root.

In order to construct a valid coin, the user has to choose a previously unspent node of label x_{coin} at the appropriate level and do the following: (1) Prove his knowledge of a valid signature on committed messages $(s, t, sk_{\mathcal{U}})$ and his knowledge of $sk_{\mathcal{U}}$. (2) Commit to the PRF seeds via commitments to the group elements $(S, T) = (h^s, h^t)$. (3) Commit to the path that connects x_{coin} to the root and prove that commitments pertain to a valid path. (4) Evaluate a coin serial number $Y_{L-\ell} = g^{1/(s+x_{coin})}$ where the input is the label of the node to be spent. (5) Generate NIZK proofs that everything was done consistently. (6) Add some material that makes it impossible to subsequently spend an ancestor or a descendant of x_{coin} without being detected.

At step (1), we use the multi-block P-signature scheme to sign the block $(s, t, sk_{\mathcal{U}})$. Using the proof produced by SigProve in the P-signature, we can efficiently prove knowledge of a signature on committed inputs in NIZK.

The trickiest problem to solve is actually (6). If $\{x_0, \ldots, x_{L-\ell}\}$ denotes the path from the root x_0 to $x_{L-\ell} = x_{coin}$, for each $j \in \{0, \ldots, L-\ell\}$, we include in the coin a pair $(T_{j,1}, T_{j,2})$ where $T_{j,1} = h^{\delta_{j,1}}$, for some random $\delta_{j,1} \xleftarrow{R} \mathbb{Z}_p$, and $T_{j,2} = e(Y_j, T_{j,1})$, where $Y_j = g^{1/(s+x_j)}$ is the value of the PRF for the label x_j. In addition, $\mathcal{U}$ must add a NIZK proof that the pair $(T_{j,1}, T_{j,2})$ was correctly calculated. By doing so, at the expense of n_s pairing evaluations at each deposit (where n_s denotes the number of previously spent coins), the bank will be able to detect whether a spent node is in the path connecting a previously spent node to the root. At the same time, if $\mathcal{U}$ does not overspend at any time, the coins he spends remain computationally unlinkable.

By itself, the pair $(T_{j,1}, T_{j,2})$ only renders cheating attempts evident. In order to expose the public key $pk_{\mathcal{U}}$ of double spenders, $\mathcal{U}$ is required to add a pair $(T_{j,3}, T_{j,4}) = (h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(Y_j, T_{j,3}))$ at each node of the path: by doing so, $pk_{\mathcal{U}}$ is exposed if $\mathcal{U}$ subsequently spends a node above x_{coin} in the path. However, we have to consider a second kind of double-spending, where the two coins involve the same tree node $x_{coin} = x_{L-\ell}$. To deal with this case, we require $\mathcal{U}$ to additionally use the seed t of his wallet and the merchant's data R and compute another security tag $Z_{L-\ell} = g^{sk_{\mathcal{U}}} g^{R/(t+x_{L-\ell})}$. The latter will be used to identify cheating users in the same way as in [10].

Finally, in order to obtain the exculpability property (and prevent a dishonest bank from wrongly accusing the user of double-spending coins), we need to add yet another pair of the form $(h^{\delta_{j,3}}, e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(Y_j, h^{\delta_3}))$, where $g_0 \in \mathbb{G}_1$ is part of the CRS, in such a way that framing the user requires to compute $e(g_0, h^{sk_{\mathcal{U}}})$ and solve a (computational) Bilinear Diffie-Hellman instance.

In order to solve problem (5), we need to generate non-interactive proofs for a number of pairing-product equations. Since the notion of anonymity requires to build a simulator that emulates the prover without knowing any witness, it means that we need NIZK proofs for pairing product equations on multiple occasions. Fortunately, the specific equations fall into the category of equations for which NIZK proofs are possible at the cost of introducing extra variables.

For this reason, we will have to introduce auxiliary variables for each pairing product equations.

Example. Suppose that, in his wallet $L = 4$, $\mathcal{U}$ uses the seed s to spend the amount of $v = 2^2$. The left part of Figure 1 represents the state of the wallet when no coin has been spent. In the rightmost tree, the black node indicates the target node x_{coin} of value v and greyed nodes are those that cannot be spent any longer once the black node was spent.

3.2 Construction

We now describe our divisible e-cash system where the withdrawal protocol allows users to obtain a wallet of a divisible coin of value 2^L.

CashSetup(λ)**:** chooses bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of order $p > 2^\lambda$ and generators $g, g_0 \xleftarrow{R} \mathbb{G}_1$, $h \xleftarrow{R} \mathbb{G}_2$. It also generates a Groth-Sahai common reference string $\mathsf{params}_{GS} = \{g, h, \vec{u}_1, \vec{u}_2, \vec{v}_1, \vec{v}_2\}$ for the perfectly soundness setting. The algorithm also selects a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$. The output is $\mathsf{params} := \{(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T),\ g_0,\ \mathsf{params}_{GS},\ H\}$.

BankKG(params, L)**:** runs $\mathsf{SigSetup}(\lambda, n)$ with $n = 3$ to obtain a key pair $(\mathsf{sk}, \mathsf{pk})$ for the P-signature of section 2.4. The bank's key pair is defined to be $(sk_{\mathcal{B}}, pk_{\mathcal{B}}) = (\mathsf{sk}, \mathsf{pk})$ and $pk_{\mathcal{B}}$ consists of

$$pk_{\mathcal{B}} = \big(u, U = g^\beta, \tilde{U} = h^\beta, \{V_i = g^{a_i}, \tilde{V}_i = h^{a_i}\}_{i=1}^3,\ L\big).$$

UserKG(params)**:** the user $\mathcal{U}$ defines his key pair as $(sk_{\mathcal{U}}, pk_{\mathcal{U}} = e(g,h)^{sk_{\mathcal{U}}})$ for a random $sk_{\mathcal{U}} \xleftarrow{R} \mathbb{Z}_p$.

Withdraw$\big(\mathcal{U}(\mathsf{params}, pk_{\mathcal{B}}, sk_{\mathcal{U}}), \mathcal{B}(\mathsf{params}, pk_{\mathcal{U}}, sk_{\mathcal{B}})\big)$**:** $\mathcal{U}$ and $\mathcal{B}$ run the following interactive protocol:

1. The user $\mathcal{U}$ first picks $s', t' \xleftarrow{R} \mathbb{Z}_p$ at random and computes perfectly hiding commitments $C_{s'} = \mathsf{Com}(s', open_{s'})$, $C_{t'} = \mathsf{Com}(t', open_{t'})$ and $C_{sk_{\mathcal{U}}} = \mathsf{Com}(sk_{\mathcal{U}}; open_{sk_{\mathcal{U}}})$. The user sends $(C_{s'}, C_{t'}, C_{sk_{\mathcal{U}}})$ to $\mathcal{B}$ and provides interactive witness indistinguishable proofs that he knows how to open $(C_{s'}, C_{t'})$. In addition, he provides an interactive zero-knowledge[2] proof that $C_{sk_{\mathcal{U}}}$ is a commitment to the private key $sk_{\mathcal{U}}$ that was used to generate $pk_{\mathcal{U}}$.
2. If the proofs verifies, $\mathcal{B}$ picks $(s'', t'') \leftarrow \mathbb{Z}_p^2$ which are sent to $\mathcal{U}$.
3. The user $\mathcal{U}$ sets $s = s' + s''$ and $t = t' + t''$, updates commitments $C_{s'}$ and $C_{t'}$ into commitments $C_s = \mathsf{Com}(s, open_s)$ and $C_t = \mathsf{Com}(t, open_t)$. The user sends (C_s, C_t) to the bank with a proof that these commitments were properly calculated.
4. $\mathcal{U}$ and $\mathcal{B}$ jointly run the protocol

$$\mathsf{SigIssue}(\mathsf{params}, \mathsf{sk}, (C_s, C_t, C_{sk_{\mathcal{U}}})) \leftrightarrows \mathsf{SigObtain}(\mathsf{params}, \mathsf{pk}, (s, t, sk_{\mathcal{U}}), (open_s, open_t, open_{sk_{\mathcal{U}}}))$$

[2] The zero-knowledge property will be needed in the proof of weak exculpability.

in such a way that $\mathcal{U}$ eventually obtains $\mathcal{B}$'s signature σ on $(s, t, sk_{\mathcal{U}})$. The user $\mathcal{U}$ stores the wallet $\mathcal{W} = (s, t, sk_{\mathcal{U}}, \sigma, \mathsf{state})$, where $\mathsf{state} = \emptyset$.

5. $\mathcal{B}$ records a debit of value $v = 2^L$ on $\mathcal{U}$'s account. It stores the transcript of the protocol and the tracing information $pk_{\mathcal{U}}$ in its database T.

Spend$\big(\mathsf{params}, pk_{\mathcal{B}}, \mathcal{W} = (s, t, sk_{\mathcal{U}}, \sigma, \mathsf{state}), 2^{\ell}, pk_{\mathcal{M}}, \mathtt{info}\big)$**:** Let us assume that $\mathcal{U}$ wants to spend a coin of value 2^{ℓ} for the wallet $\mathcal{W}$ of initial value 2^L. Using state, $\mathcal{U}$ determines the label $x_{coin} \in [1, 2^{L+1} - 1]$ of the first node corresponding to an unspent coin at height ℓ in the tree associated with the wallet. Let $\{x_0, x_1, \ldots, x_{L-\ell}\}$ denote the path connecting node $x_{coin} = x_{L-\ell}$ to the root $x_0 = 1$ of the tree. The user $\mathcal{U}$ computes $S = h^s$ and $T = h^t$ and conducts the following steps.

1. $\mathcal{U}$ has to prove that he knows a signature σ on the committed vector $(s, t, sk_{\mathcal{U}}) \in \mathbb{Z}_p^3$. To this end, he first generates commitments and proofs $\big(\{C_{S,i}\}_{i=1}^3, \{C_{T,i}\}_{i=1}^3, \{C_{\mathcal{U},i}\}_{i=1}^3, \pi^{sig}\big) \leftarrow \mathsf{SigProve}(\mathsf{params}, \mathsf{pk}, \sigma, (s, t, sk_{\mathcal{U}}))$. The output of $\mathsf{SigProve}$ includes $\{C_{\mathcal{U},i}\}_{i=1}^3$, which are commitments to $(L_{\mathcal{U},1}, L_{\mathcal{U},2}, L_{\mathcal{U},3}) = (h^{sk_{\mathcal{U}}}, u^{sk_{\mathcal{U}}}, \tilde{V}_3^{sk_{\mathcal{U}}})$, and $\{C_{S,i}, C_{T,i}\}_{i=1}^3$, that contain $(L_{S,1}, L_{S,2}, L_{S,3}) = (h^s, u^s, \tilde{V}_1^s)$ and $(L_{T,1}, L_{T,2}, L_{T,3}) = (h^t, u^t, \tilde{V}_2^t)$, respectively. In addition, $\mathcal{U}$ computes $C_{K_{\mathcal{U}}} = \mathsf{GSCom}(h^{sk_{\mathcal{U}}}, open'_{\mathcal{U}})$ as a commitment to $K_{\mathcal{U}} = h^{sk_{\mathcal{U}}}$ and generates a NIZK proof $\pi_{K_{\mathcal{U}}} \leftarrow \mathsf{EqComProve}(L_{\mathcal{U},1}, K_{\mathcal{U}})$ that $C_{K_{\mathcal{U}}}$ and $C_{\mathcal{U},1}$ are commitment to the same value. This amounts to prove that

$$e(L_{\mathcal{U},1}/K_{\mathcal{U}}, h^{\theta}) = 1_{\mathbb{G}_T} \quad \text{and} \quad \theta = 1, \tag{1}$$

for some variable $\theta \in \mathbb{Z}_p$ contained in $C_{\theta} = \mathsf{GSCom}(\theta, open_{\theta})$ and that will be re-used in subsequent steps of the spending protocol.

2. For $j = 0$ to $L - \ell$ do the following.

   a. If $j > 0$, generate a commitment $C_{X_j} = \mathsf{GSCom}(h^{x_j}, open_{x_j})$ to $X_j = h^{x_j}$ and a proof that $x_j = 2x_{j-1} + b_j$, for some bit $b_j \in \{0, 1\}$. To this end, generate the commitments $C_{b_j} = \mathsf{GSCom}(g^{b_j}, open_{b_j})$ and $C'_{b_j} = \mathsf{GSCom}(h^{b_j}, open'_{b_j})$ as well as a NIZK proof $\pi_{x_j} \leftarrow \mathsf{EqComProve}(C_{X_j}, C'_{X_j})$ that C_{X_j} and $C'_{X_j} = C_{X_{j-1}}^2 \cdot C'_{b_j}$ open to the same value. To prove that $b_j \in \{0, 1\}$, $\mathcal{U}$ generates a NIZK proof $(\pi_{b_j,1}, \pi_{b_j,2})$ for the pairing-product equations $e(g^{b_j}, h) = e(g, h^{b_j})$ and $e(g^{b_j}, h^{b_j}) = e(g^{b_j}, h)$, which guarantee that $b_j^2 = b_j$, so that $b_j \in \{0, 1\}$.

   b. If $j < L - \ell$, generate a commitment $C_{Y_j} = \mathsf{GSCom}(Y_j, open_{Y_j})$ to the PRF value $Y_j = g^{1/(s+x_j)}$ as well as a NIZK proof π_{Y_j} that it satisfies $e(Y_j, L_{S,1} \cdot X_j) = e(g, h)$, where $X_j = h^{x_j}$. This consists of a commitment $C_{\Phi_{Y_j}}$ to a variable $\Phi_{Y_j} \in \mathbb{G}_1$ and a proof that $e(\Phi_{Y_j}, L_{S,1} \cdot X_j) = e(g, h)$ and $e(Y_j/\Phi_{Y_j}, h^{\theta}) = 1_{\mathbb{G}_T}$. Then, pick

$\delta_{j,1}, \delta_{j,2}, \delta_{j,3} \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\begin{aligned} T_{j,1} &= h^{\delta_{j,1}}, & T_{j,2} &= e(Y_j, h)^{\delta_{j,1}} \\ T_{j,3} &= h^{\delta_{j,2}}, & T_{j,4} &= pk_{\mathcal{U}} \cdot e(Y_j, h)^{\delta_{j,2}}, \\ T_{j,5} &= h^{\delta_{j,3}}, & T_{j,6} &= e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(Y_j, h)^{\delta_{j,3}}. \end{aligned}$$

Generate NIZK proofs $(\pi_{j,T_1}, \pi_{j,T_3}, \pi_{j,T_5})$ that $(Y_j, K_{\mathcal{U}})$ satisfy

$$\begin{aligned} T_{j,2} &= e(Y_j, T_{j,1}) \\ T_{j,4} &= e(g, K_{\mathcal{U}}) \cdot e(Y_j, T_{j,3}) \\ T_{j,6} &= e(g_0, K_{\mathcal{U}}) \cdot e(Y_j, T_{j,5}). \end{aligned} \tag{2}$$

These proofs require new commitments $\{C_{\Phi_{j,k}}\}_{k=1,3,5}$, $C_{\Phi'_{Y_j}}$ and $C_{\Phi''_{Y_j}}$ to auxiliary variables $\{\Phi_{j,k}\}_{k=1,3,5}$, $\Phi'_{Y_j}, \Phi''_{Y_j} \in \mathbb{G}_1$ respectively and proofs for relations

$$\begin{aligned} &T_{j,2} = e(Y_j, \Phi_{j,1}), \\ &T_{j,4} = e(g, K_{\mathcal{U}}) \cdot e(\Phi_{Y'_j}, \Phi_{j,3}) \\ &T_{j,6} = e(g_0, K_{\mathcal{U}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5}), \end{aligned} \qquad \begin{aligned} &e(Y_j/\Phi_{Y'_j}, h^{\theta}) = 1_{\mathbb{G}_T}, \\ &e(Y_j/\Phi_{Y''_j}, h^{\theta}) = 1_{\mathbb{G}_T}, \\ &\{e(g^{\theta}, T_{j,k}/\Phi_{j,k}) = 1_{\mathbb{G}_T}\}_{k\in\{1,3,5\}}. \end{aligned}$$

c. If $j = L - \ell$, compute the serial number $Y_{L-\ell} = g^{1/(s+x_{L-\ell})}$ and generate a NIZK proof $\pi_{Y_{L-\ell}}$ that $e(Y_{L-\ell}, L_{S,1} \cdot X_{L-\ell}) = e(g, h)$. This proof consists of a commitment $C_{\Phi_{Y_{L-\ell}}}$ to $\Phi_{Y_{L-\ell}} \in \mathbb{G}_1$ and proofs for equations

$$e(\Phi_{Y_{L-\ell}}, L_{S,1} \cdot X_{L-\ell}) = e(g, h), \qquad e(Y_{L-\ell}/\Phi_{Y_{L-\ell}}, h^{\theta}) = 1_{\mathbb{G}_T}.$$

Compute $Z_{L-\ell} = g^{sk_{\mathcal{U}}} \cdot g^{R/(t+x_{L-\ell})}$, where $R = H(\mathtt{info}, pk_{\mathcal{M}}) \in \mathbb{Z}_p$, and a NIZK proof $\pi_{Z_{L-\ell}}$ that $Z_{L-\ell}$ is well-formed. This requires new Groth-Sahai commitments $C_{W_{L-\ell}}, C_{\Phi_{W_{L-\ell}}}$ to auxiliary variables $W_{L-\ell} = g^{1/(t+x_{L-\ell})}$, $\Phi_{W_{L-\ell}} = g^{1/(t+x_{L-\ell})}$ and a proof that:

$$\begin{aligned} e(g, K_{\mathcal{U}}) \cdot e(W_{L-\ell}, h^R) &= e(Z_{L-\ell}, h), \\ e(W_{L-\ell}, L_{T,1} \cdot X_{L-\ell}) &= e(g, h) \\ e(W_{L-\ell}/\Phi_{W_{L-\ell}}, h^{\theta}) &= 1_{\mathbb{G}_T}. \end{aligned}$$

Finally, update state into $\mathsf{state}' = \mathsf{state} \cup \{(x_{coin})\}$ and output the coin

$$\begin{aligned} coin = \Big(& \{C_{S,i}\}_{i=1}^{3}, \{C_{T,i}\}_{i=1}^{3}, \{C_{\mathcal{U},i}\}_{i=1}^{3}, \ C_{K_{\mathcal{U}}}, \ \pi_{K_{\mathcal{U}}}, \ \pi^{sig}, \\ & \{C_{X_j}, \ C_{b_j}, \ C'_{b_j}, \ \pi_{x_j}, \ \pi_{b_j,1} \ \pi_{b_j,2}\}_{j=1}^{L-\ell}, \\ & \{(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6}), \ C_{Y_j}, \ C_{\Phi_{Y_j}}, \ C'_{\Phi_{Y_j}}, \ C''_{\Phi_{Y_j}}, \\ & \{C_{\Phi_{j,k}}\}_{k\in\{1,3,5\}}, \ \pi_{Y_j}, \ \pi_{j,T_1}, \ \pi_{j,T_3}, \ \pi_{j,T_5}\}_{j=0}^{L-\ell-1}, \\ & Y_{L-\ell}, \ Z_{L-\ell}, \ C_{\Phi_{Y_{L-\ell}}}, C_{W_{L-\ell}}, \ C_{\Phi_{W_{L-\ell}}}, \ \pi_{Y_{L-\ell}}, \ \pi_{Z_{L-\ell}}, \ \mathtt{info} \Big) \end{aligned}$$

VerifyCoin$(\mathsf{params}, pk_{\mathcal{M}}, pk_{\mathcal{B}}, v = 2^{\ell}, coin)$**:** parse *coin* as above. Return 1 iff all proofs verify.

Deposit$(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, 2^{\ell}, \mathrm{DB}_{\mathcal{B}})$**:** parse *coin* as above and perform the same checks as VerifyCoin. Then, define $\mathrm{DB}'_{\mathcal{B}} = \mathrm{DB}_{\mathcal{B}} \cup \{(coin, \mathsf{flag}, 2^{\ell}, pk_{\mathcal{M}})\}$ where the value of flag depends on whether *coin* is a valid coin of value 2^{ℓ} and whether a cheating attempt is detected.

- If *coin* does not properly verify, $\mathcal{B}$ sets $\mathsf{flag} =$ "$\mathcal{M}$" to indicate a cheating merchant.
- If *coin* properly verifies, the bank $\mathcal{B}$ runs the following test. For each entry $(coin_s, \mathsf{flag}_s, 2^{\ell_s}, pk_{\mathcal{M}_s}) \in \mathrm{DB}_{\mathcal{B}}$, where $s = 1$ to $|\mathrm{DB}_{\mathcal{B}}|$, $\mathcal{B}$ parses $coin_s$ as above. If $(\mathtt{info}_s, pk_{\mathcal{M}_s}) = (\mathtt{info}, pk_{\mathcal{M}})$, $\mathcal{B}$ sets $\mathsf{flag} =$ "$\mathcal{M}$". Otherwise, from $coin_s$, it extracts the path $\{(T_{s,j,1}, T_{s,j,2}, T_{s,j,3}, T_{s,j,4})\}_{j=0}^{L-\ell_s-1}$, the serial number $Y_{L-\ell_s} \in \mathbb{G}_1$ and the tag $Z_{L-\ell_s} \in \mathbb{G}_1$. It also parses *coin* to extract the path $\{(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4})\}_{j=0}^{L-\ell-1}$, the serial number $Y_{L-\ell} \in \mathbb{G}_1$ and the tag $Z_{L-\ell}$. If $\ell < \ell_s$ and $T_{L-\ell_s,2} = e(Y_{L-\ell_s}, T_{L-\ell_s,1})$, $\mathcal{B}$ sets $\mathsf{flag} =$ "$\mathcal{U}$", outputs $coin_s$ and *coin* and reports a double-spending. Likewise, if $\ell > \ell_s$ and $T_{s,L-\ell,2} = e(Y_{L-\ell}, T_{s,L-\ell,1})$, $\mathcal{B}$ also sets $\mathsf{flag} =$ "$\mathcal{U}$" and outputs $coin_s$ and *coin*. Finally, if $\ell = \ell_s$, $\mathcal{B}$ sets $\mathsf{flag} =$ "$\mathcal{U}$" if and only if $Y_{L-\ell} = Y_{L-\ell_s}$.
- If *coin* verifies and no double-spending is detected, $\mathcal{B}$ sets $\mathsf{flag} =$ "accept" and credits the account of $pk_{\mathcal{M}}$ by the amount of 2^{ℓ}.

After the above tests, the updated database $\mathrm{DB}'_{\mathcal{B}}$ supersedes $\mathrm{DB}_{\mathcal{B}}$.

Identify$(\mathsf{params}, pk_{\mathcal{B}}, coin_a, coin_b)$**:** on input of fraudulent coins $coin_a$ and $coin_b$, the bank $\mathcal{B}$ can identify the double-spender as follows.

1. Extract $\mathtt{info}_a$, $\{(T_{j,1}^{(a)}, T_{j,2}^{(a)}, T_{j,3}^{(a)}, T_{j,4}^{(a)})\}_{j=0}^{L-\ell_a-1}$, $(Y_{L-\ell_a}^{(a)}, Z_{L-\ell_a}^{(a)}) \in \mathbb{G}_1^2$ from $coin_a$. Also, parse $coin_b$ to retrieve $\mathtt{info}_b$, $\{(T_{j,1}^{(b)}, T_{j,2}^{(b)}, T_{j,3}^{(b)}, T_{j,4}^{(b)})\}_{j=0}^{L-\ell_b-1}$ and $(Y_{L-\ell_b}^{(b)}, Z_{L-\ell_b}^{(b)}) \in \mathbb{G}_1^2$.
2. If $\ell_b > \ell_a$, recover $pk_{\mathcal{U}}$ as $pk_{\mathcal{U}} = T_{L-\ell_b,4}^{(a)} / e(Y_{L-\ell_b}^{(b)}, T_{L-\ell_b,3}^{(a)})$. If $\ell_b < \ell_a$, $pk_{\mathcal{U}}$ can be obtained as $pk_{\mathcal{U}} = T_{L-\ell_a,4}^{(b)} / e(Y_{L-\ell_a}^{(a)}, T_{L-\ell_a,3}^{(b)})$. In the case $\ell_a = \ell_b$, we must have $Y_{\ell_a} = Y_{\ell_b}$. Then, $\mathcal{B}$ computes $R_a = H(\mathtt{info}_a, pk_{\mathcal{M}_a})$, $R_b = H(\mathtt{info}_b, pk_{\mathcal{M}_b})$ and then $\kappa = (Z_{L-\ell_a}^{(a)} / Z_{L-\ell_b}^{(b)})^{1/(R_a - R_b)}$, which allows recovering $g^{sk_{\mathcal{U}}} = Z_{L-\ell_a}^{(a)} / \kappa^{R_a}$ and thus $pk_{\mathcal{U}} = e(g^{sk_{\mathcal{U}}}, h)$.

The security of the scheme relies on the collision-resistance of H and the intractability assumptions recalled in Section 2.3. More precisely, we state the following theorem for which a proof is given in the full version of the paper.

Theorem 2. *Assuming that H is a collision-resistant hash function and that the SXDH, D3DH, TDH, D3DH, q_w-HSDH and the 2^{L+2}-DDHI assumptions where q_w denotes the number of $\mathcal{Q}_{\mathsf{withdraw}}$ queries all hold in $(\mathbb{G}_1, \mathbb{G}_2)$, our e-cash scheme provides anonymity, balance, identification and weak-exculpability.*

The most difficult part of the security proof is the proof of anonymity. More precisely, when it comes to build a simulator, we need to simulate NIZK proofs

for pairing product equations of the form (2), which is non-trivial. Indeed, as noted in [31], this is only known to be possible when the target element of the equation (which lives in $\mathbb{G}_T$) can be written as a pairing of known elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. The problem is that, in equations like (2), some pairing values have to be gradually replaced by uniformly random values of $\mathbb{G}_T$. To deal with this problem, we appeal to the D3DH assumption in a similar way to [33]. Namely, the D3DH input element Γ, which is either g^{abc} or a random element of $\mathbb{G}_1$, is available as a "pre-image" of the target pairing value and makes it possible to simulate proofs for pairing product equations at the expense of introducing auxiliary variables.

References

1. Au, M.H., Susilo, W., Mu, Y.: Practical Anonymous Divisible E-Cash from Bounded Accumulators. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 287–301. Springer, Heidelberg (2008)
2. Au, M.H., Wu, Q., Susilo, W., Mu, Y.: Compact E-Cash from Bounded Accumulator. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 178–195. Springer, Heidelberg (2006)
3. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and Noninteractive Anonymous Credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
4. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable Proofs and Delegatable Anonymous Credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
5. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Compact E-Cash and Simulatable VRFs Revisited. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 114–131. Springer, Heidelberg (2009)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73 (1993)
7. Blazy, O., Canard, S., Fuchsbauer, G., Gouget, A., Sibert, H., Traoré, J.: Achieving Optimal Anonymity in Transferable E-Cash with a Judge. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 206–223. Springer, Heidelberg (2011)
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. Boyen, X., Waters, B.: Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
10. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-Cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
11. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Balancing Accountability and Privacy Using E-Cash (Extended Abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 141–155. Springer, Heidelberg (2006)
12. Camenisch, J., Kohlweiss, M., Soriente, C.: An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)

13. Camenisch, J., Lysyanskaya, A., Meyerovich, M.: Endorsed E-Cash. In: IEEE Security & Privacy 2007, pp. 101–115 (2007)
14. Canard, S., Gouget, A.: Divisible E-Cash Systems Can Be Truly Anonymous. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 482–497. Springer, Heidelberg (2007)
15. Canard, S., Gouget, A., Traoré, J.: Improvement of Efficiency in (Unconditional) Anonymous Transferable E-Cash. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 202–214. Springer, Heidelberg (2008)
16. Canard, S., Gouget, A.: Anonymity in Transferable E-cash. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 207–223. Springer, Heidelberg (2008)
17. Canard, S., Gouget, A., Hufschmitt, E.: A Handy Multi-coupon System. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 66–81. Springer, Heidelberg (2006)
18. Canard, S., Gouget, A.: Multiple Denominations in E-cash with Compact Transaction Data. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 82–97. Springer, Heidelberg (2010)
19. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. In: STOC 1998, pp. 209–218. ACM Press (1998)
20. Chan, A., Frankel, Y., Tsiounis, Y.: Easy Come - Easy Go Divisible Cash. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 561–575. Springer, Heidelberg (1998)
21. Chase, M., Lysyanskaya, A.: Simulatable VRFs with Applications to Multi-theorem NIZK. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 303–322. Springer, Heidelberg (2007)
22. Chaum, D.: Blind Signatures for Untraceable Payments. In: Crypto 1982, pp. 199–203 (1982)
23. Chaum, D.: Blind Signature Systems. In: Crypto 1983, p. 153 (1983)
24. Chaum, D., Fiat, A., Naor, M.: Untraceable Electronic Cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg (1990)
25. Chaum, D., Pedersen, T.P.: Transferred Cash Grows in Size. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 390–407. Springer, Heidelberg (1993)
26. D'Amiano, S., Di Crescenzo, G.: Methodology for Digital Money Based on General Cryptographic Tools. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 156–170. Springer, Heidelberg (1995)
27. Dodis, Y.: Efficient Construction of (Distributed) Verifiable Random Functions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 1–17. Springer, Heidelberg (2002)
28. Dodis, Y., Yampolskiy, A.: A Verifiable Random Function with Short Proofs and Keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
29. Franklin, M.K., Yung, M.: Secure and Efficient Off-Line Digital Money. In: Lingas, A., Carlsson, S., Karlsson, R. (eds.) ICALP 1993. LNCS, vol. 700, pp. 265–276. Springer, Heidelberg (1993)
30. Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Transferable Constant-Size Fair E-Cash. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 226–247. Springer, Heidelberg (2009)
31. 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)

32. Jarecki, S., Shmatikov, V.: Efficient Two-Party Secure Computation on Committed Inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
33. Libert, B., Vergnaud, D.: Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 498–517. Springer, Heidelberg (2009)
34. Micali, S., Rabin, M.-O., Vadhan, S.: Verifiable Random Functions. In: FOCS 1999, pp. 120–130 (1999)
35. Nakanishi, T., Sugiyama, Y.: Unlinkable Divisible Electronic Cash. In: Okamoto, E., Pieprzyk, J.P., Seberry, J. (eds.) ISW 2000. LNCS, vol. 1975, pp. 121–134. Springer, Heidelberg (2000)
36. Okamoto, T., Ohta, K.: Universal Electronic Cash. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 324–337. Springer, Heidelberg (1992)
37. Okamoto, T.: An Efficient Divisible Electronic Cash Scheme. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 438–451. Springer, Heidelberg (1995)
38. Pailles, J.-C.: New Protocols for Electronic Money. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 263–274. Springer, Heidelberg (1993)
39. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
40. Tsiounis, Y.: Efficient Electronic Cash: New Notions and Techniques. PhD Thesis, Northeaster University, Boston (1997)

Author Index