Tsuyoshi Takagi
Tatsuaki Okamoto
Eiji Okamoto
Takeshi Okamoto (Eds.)

Pairing-Based Cryptography–Pairing 2007

LNCS 4575

First International Conference
Tokyo, Japan, July 2007
Proceedings

Springer

Lecture Notes in Computer Science 4575

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

Tsuyoshi Takagi Tatsuaki Okamoto
Eiji Okamoto Takeshi Okamoto (Eds.)

Pairing-Based Cryptography – Pairing 2007

First International Conference
Tokyo, Japan, July 2-4, 2007
Proceedings

Springer

Volume Editors

Tsuyoshi Takagi
Future University Hakodate, Japan
E-mail: takagi@fun.ac.jp

Tatsuaki Okamoto
NTT Laboratories, Nippon Telegraph and Telephone Corporation, Yokosuka, Japan
E-mail: okamoto.tatsuaki@lab.ntt.co.jp

Eiji Okamoto
University of Tsukuba, Japan
E-mail: okamoto@risk.tsukuba.ac.jp

Takeshi Okamoto
University of Tsukuba, Japan
E-mail: ken@risk.tsukuba.ac.jp

Library of Congress Control Number: 2007929851

CR Subject Classification (1998): E.3, D.4.6, F.2.1-2, G.2, K.6.5, C.2

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-73488-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-73488-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12087259 06/3180 5 4 3 2 1 0

Preface

The 1st International Conference on Pairing-Based Cryptography (Pairing 2007) was held in Tokyo, Japan, during July 2–4, 2007. Pairing-based cryptography has been causing a paradigm shift in cryptography–an ever increasing number of novel protocols using pairing have been appearing in the literature: identity-based encryption, short signature, and efficient broadcast encryption, to mention but a few. The aim of Pairing 2007 was to bring together leading researchers and practitioners from academia and industry, all concerned with problems related to pairing-based cryptography. This conference was sponsored by the University of Tsukuba, and it was offered in cooperation with Tokyo Section, The Institute of Electrical and Electronics Engineers (IEEE) and The Japan Society for Industrial and Applied Mathematics (JSIAM).

Pairing 2007 received 86 submissions from all over the world. Each paper was double-blindly evaluated by at least three reviewers regarding the technical relevance to pairing-based cryptography, and 18 papers were selected for presentation at Pairing 2007. We would like to thank all members of the Program Committee for their support and enormous investment of time throughout the whole delicate review process. We are also extremely grateful for all the help and support of a large number of external reviewers who reviewed submissions in their area of expertise.

The program also included five invited talks by Dan Boneh from Stanford University, USA; Steven Galbraith from Royal Holloway University of London, UK; Alfred Menezes from University of Waterloo, Canada; Takakazu Satoh from Tokyo Institute of Technology, Japan; and Michael Scott from Dublin City University, Ireland. At least one page of the extended abstract for each invited talk is included in the proceedings.

We would like to thank all the members of the Organizing Committee for their work in the publication and organization of Pairing 2007. The review server and its backup system unexpectedly went out of service, and the review process had to be restarted from scratch. We gratefully thank Atsuo Inomata and Akira Kanaoka for their enormous support in installing and operating the iChair system during the review process. Again, we gratefully appreciate the patient effort of all reviewers, especially in light of the server accident.

Finally, we would like to thank all the authors who submitted papers to Pairing 2007.

April 2007

Tsuyoshi Takagi
Tatsuaki Okamoto
Eiji Okamoto
Takeshi Okamoto

Pairing 2007

The First International Conference on Pairing-Based Cryptography

Tokyo, Japan, July 2–4, 2007

Sponsored by the *University of Tsukuba*, *IEEE Tokyo Section*, and *The Japan Society for Industrial and Applied Mathematics (JSIAM)*.

General Co-chairs

Eiji Okamoto, University of Tsukuba, Japan
Takeshi Okamoto, University of Tsukuba, Japan

Program Co-chairs

Tsuyoshi Takagi, Future University Hakodate, Japan
Tatsuaki Okamoto, NTT, Japan

Organizing Co-chairs

Hiroshi Doi, Institute of Information Security, Japan
Atsuo Inomata, Japan Science and Technology Agency, Japan
Akira Kanaoka, University of Tsukuba, Japan
Masahiro Mambo, University of Tsukuba, Japan

Program Committee

Paulo Barreto, University of Sao Paulo, Brazil
Johannes Buchmann, Technische Universität Darmstadt, Germany
Jan Camenisch, IBM Zurich Research Laboratory, Switzerland
Jinhui Chao, Chuo University, Japan
Jean-Sebastien Coron, University of Luxembourg, Luxembourg
Iwan Duursma, University of Illinois at Urbana-Champaign, USA
Andreas Enge, Ecole Polytechnique, France
Jun Furukawa, NEC, Japan
David Galindo, University of Malaga, Spain

Goichiro Hanaoka, AIST, Japan
Tetsuya Izu, Fujitsu, Japan
Michael Jacobson, University of Calgary, Canada
Antoine Joux, DGA and Universite de Versailles, France
Marc Joye, Thomson R&D, France
Kwangjo Kim, Information and Communications University, Korea
Tetsutaro Kobayashi, NTT, Japan
Soonhak Kwon, Sungkyunkwan University, Korea
Tanja Lange, Technische Universiteit Eindhoven, The Netherlands
Hyang-Sook Lee, Ewha Womans University, Korea
Atsuko Miyaji, JAIST, Japan
Dan Page, University of Bristol, UK
Jean-Jacques Quisquater, Universite catholique de Louvain, Belgium
Ryuichi Sakai, Osaka Electro-Communication University, Japan
Palash Sarkar, Indian Statistical Institute, India
Igor Shparlinski, Macquarie University, Australia
Nigel Smart, University of Bristol, UK
Willy Susilo, University of Wollongong, Australia
Routo Terada, University of Sao Paulo, Brazil
Shigenori Uchiyama, Tokyo Metropolitan University, Japan
Guilin Wang, Institute for Infocomm Research, Singapore
Victor Wei, Chinese University of Hong Kong, China
Moti Yung, RSA Labs and Columbia University, USA
Fangguo Zhang, Sun Yat-sen University, China

External Reviewers

Nuttapong Attrapadung
Kazumaro Aoki
Joonsang Baek
Daniel J. Bernstein
Jean-Luc Beuchat
Reinier Broker
Sanjit Chatterjee
Xiaofeng Chen
Jung Hee Cheon
Benoit Chevallier-Mames
Sherman S. M. Chow
Thomas Heide Clausen
Yang Cui
Yevgeniy Dodis
Hiroshi Doi
Dang Nguyen Duc
David Freeman
Toshinori Fukunaga
Steven Galbraith
DongGuk Han
Colm O hEigeartaigh
Laura Hitt
Fumitaka Hoshino
Xinyi Huang
Heng Swee Huay
Toshiyuki Issiki
Naoki Kanayama
Aggelos Kiayias
Yongdae Kim
Hirotsugu Kinoshita
Yuichi Komano
Noboru Kunihiro
Soonhak Kwon
Taekyoung Kwon
Dong Hoon Lee
Eunjeng Lee
Hyunrok Lee
Taekyoung Lee
Yoonjin Lee
Benoit Libert
Vo Duc Liem
Joseph K. Liu
Kazuto Matsuo
Maria Meyerovich
Michael Naehrig
Koh-ichi Nagao
Toshiya Nakajima
Gregory Neven
Keiichiro Nishimoto
Wakaha Ogata
Sunsuke Ohta
Takao Okubo
Akira Otsuka
Jehong Park

Young-Ho Park
Pieter Rozenhart
Eun-Kyung Ryu
Taiichi Saito
Abhi Shelat
Alan Silvester
Kyungah Shim
SeongHan Shin
Mike Scott
Martijn Stam
Koutarou Suzuki
Katsuyuki Takashima
Masahiko Takenaka
Satoru Tanaka
Adrian Tang
Mark Velichka
Jonathan Webster
Rui Zhang

Table of Contents

Algorithms

Protocol II

Bilinear Groups of Composite Order

Dan Boneh*

Computer Science Dept.
Stanford University
dabo@cs.stanford.edu

Abstract. Bilinear groups of composite order are groups with an efficient bilinear map where the group order is a product of two large primes. Such groups are constructed from pairing friendly curves over a finite field. Composite order bilinear groups were recently used in a number of interesting cryptographic constructions. This talk will survey three applications:

1. Private Information Retrieval,
2. Anonymous Identity Based Encryption, and
3. Non-Interactive Zero Knowledge.

* Supported by NSF and the Packard Foundation.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, p. 1, 2007.

Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System

Yong Ho Hwang[1] and Pil Joong Lee[2]

[1] Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA
yhhwang@cs.jhu.edu
[2] Department of Electronic and Electrical Engineering, POSTECH, Pohang, Korea
pjl@postech.ac.kr

Abstract. We study the problem of a public key encryption with conjunctive keyword search (PECK). The keyword searchable encryption enables a user to outsource his data to the storage of an untrusted server and to have the ability to selectively search his data without leaking information. The PECK scheme provides the document search containing each of several keywords over a public key setting. First, we construct an efficient PECK scheme whose security is proven over a decisional linear Diffie-Hellman assumption in the random oracle model. In comparison with previous schemes, our scheme has the shortest ciphertext size and private key size, and requires a comparable computation overhead. Second, we discuss problems related to the security proof of previous schemes and show they cannot guarantee complete security. Finally, we introduce a new concept called a multi-user PECK scheme, which can achieve an efficient computation and communication overhead and effectively manage the storage in a server for a number of users.

1 Introduction

A remote storage system allows a user to store his data in the storage of a (untrusted) remote server and access them over mobile or wireless networks. To preserve the confidentiality of the user, stored data in the storage of a remote server must be encrypted. However, it is hard to selectively retrieve the encrypted data, which a user is searching for, from the untrusted server without revealing any information about the data. A naive solution for this would be to have a user download all encrypted data from the server and then search for them whenever he wants to retrieve his data. In this situation, a user would indiscriminately receive and decrypt all encrypted data, regardless of what data he is looking for. This solution is not practical because too much bandwidth are required and user devices have limited resources. To cope with this problem various techniques for searching encrypted data have been proposed [30,21,31,19,11,4,26]. When storing the data in a server, the documents associate segments derived from keywords with encrypted data. Then, to search the encrypted data, a user generates a trapdoor for a certain keyword w and the server can search the encrypted documents

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 2–22, 2007.

containing w from it. However, the server learns nothing else about the documents. While many previous schemes provide a single keyword search, practical systems desire a multiple keyword search. Consider, for example, a secure audit log system [31]. The contents of an audit log should be protected from unauthorized parties since it may contain sensitive information. To assess past system activity, auditors will want to search audit logs. At the time of the inspection, an audit escrow agent gives the auditors the authority to search for specific data in the audit logs. While searchable encryption is a good solution for this example, a single keyword search cannot satisfy multiple search requirements. If the auditor wants to get the audit records for an "investment" by "Alice" in "Aug", the audit log system would perform the search procedure on a large amount of data repeatedly. The naive solutions to searching for multiple keywords are set intersection and meta-keywords [21,30]. However, these approaches have obvious flaws. Set intersection allows the server to learn information on the documents containing each individual keyword, and meta-keywords require an exponential storage and search time for the number of keywords. To overcome this shortcoming, the systems that provide *conjunctive* keyword search on encrypted data have been investigated [22,26,4,17]. In this paper, we study conjunctive keyword searchable encryption that minimizes communication and storage overhead for both the server and the client.

Related Work. Song, Wagner, and Perrig introduced the concept of searchable encryption and presented practical solutions [30]. Later, Goh defined the notion of security for searchable encryption and constructed an efficient scheme using Bloom filters [21]. Moreover, several works were introduced to improve the efficiency of the system and provide stronger security [18,31,19,1]. However, their works only support a single keyword search in symmetric key setting.

The concept of conjunctive keyword searchable encryption was first addressed by Golle, Staddon, and Waters [22]. They defined a security model for conjunctive keyword search over encrypted data and provided two secure constructions. In their schemes, an encrypted file is associated with searchable encryption for a number of keywords. Certain keywords are assigned to separate keyword fields and encrypted with the user's secret key. If a user queries a trapdoor with several keywords, then the server can search a file containing those keywords with a trapdoor and keyword encryption segments of searchable encryption just by one test operation. While their constructions are very efficient in comparison to single keyword searchable encryptions for searching multiple keywords, the first one requires the trapdoor size to be linear in the number of keywords and the second one needs a number of pairing computations to search for multiple keywords. Recently, Ballard, Kamara, and Monrose proposed two constructions that minimize the trapdoor size and the computation overhead [4]. One is based on the Shamir Secret Sharing [29] and the other is based on an asymmetric bilinear map [3]. While their first construction has a very efficient search algorithm, it requires a linear trapdoor size in the number of documents. The second one has a

constant trapdoor size and achieves efficient computation and storage overhead. However, their schemes can only search symmetrically encrypted data.

Most of the previous searchable encryptions were constructed in a symmetric key setting. In this setting, a user encrypts and stores his private data in the storage of remote server. A user can then retrieve his private data with a particular keyword from the remote storage. However, these systems cannot be used for practical applications such as in an email routing system. Suppose that a user *Alice* wants to download her encrypted emails from an email gateway into several devices such as laptop, pager, desktop, etc. She may want to receive only urgent or important mail to her mobile devices, but then later download all emails to desktop. To retrieve particular emails from the routing server without leaking any information about her emails, the email contents and keywords are encrypted under her public key since emails are collected from a number of senders. For this application, Boneh *et al.* first addressed public key encryptions with keyword search based on several assumptions [11]. In addition, Waters *et al.* showed that encrypted and searchable audit log system can be efficiently constructed from the public key searchable encryption [31]. However, these constructions cannot efficiently search emails when a user wants to retrieve emails with several keywords, such as an "urgent" email about "finance" from "Bob", since their constructs only support a single keyword search.

Park, Kim, and Lee introduced two efficient constructions that can search keywords conjunctively in a public key setting [26]. We call their schemes `PKL I` and `PKL II`. They have an efficient computation overhead and a constant trapdoor size. However, the `PKL I` scheme requires a number of bilinear pairing computations for encrypting keywords and the `PKL II` scheme has the private keys in proportion to the number of keyword fields. Recently, Boneh and Waters introduced public-key systems with conjunctive, subset, and range queries on encrypted data [16]. Their systems can also support searchable keyword-based message encryption. While their systems have interesting properties, the trapdoor size is linear with the number of conjunctively searching keywords and the ciphertext size is longer than that of other constructions because it uses bilinear groups of composite order [13]. For equality testing, the server in their systems should perform bilinear pairing operations in proportion to the number of searchable keywords. In addition, the security of their schemes is proven in a slightly weaker model in which the adversary commits to the search keywords at the beginning of the game. In this paper, we focus on public key encryption with conjunctive keyword search (PECK).

Our Contribution. Our aim is to construct an efficient and secure PECK scheme, which minimizes the communication and storage overhead for both the server and the user for various practical storage systems. Our results are summarized as follows.

1. We first construct a simple and efficient PECK scheme with short ciphertext size and only one private key. We prove that our scheme is secure under the

decisional linear Diffie-Hellman (DLDH) assumption in the random oracle model. In a remote storage system, the server should store all encrypted data for a number of users. Hence, reducing the ciphertext size is a very important problem. Our scheme has the shortest ciphertext size in comparison to previous PECK schemes. The ciphertext size of our scheme is half that of the PKL II scheme which has a shorter ciphertext than that of the PKL I scheme. In addition, the PKL schemes have other limitations. In the PKL I scheme, a user performs bilinear pairing operations linear to the number of keyword fields to encrypt keywords. A number of the pairing computations impose a burden to a user with limited resources. While our scheme is based on the bilinear map, a bilinear pairing operation is not required for a user. In our scheme, a user stores only one private key in his secure device. However, in the PKL II scheme, a user should store private keys linear with the number of keyword fields. Therefore, our scheme minimizes the communication and storage overhead for both the server and users and has an efficient computation overhead.

2. We analyze the security of the PKL schemes. In [26], Park, Kim, and Lee modified the security model of Golle, Staddon, and Waters and showed their schemes are secure in a modified model. However, their security proofs do not satisfy their security model. Their security model restricts only trapdoor queries that can distinguish a challenge cipehrtext. However, all trapdoors including a word of the target documents cannot be asked in their security proofs, even if a trapdoor cannot distinguish the challenge ciphertext. Therefore, their schemes cannot guarantee complete security in practical applications. We discuss the problems of their security proofs and show that the PKL I scheme is broken by this limitation.
3. We introduce a new concept, called a multi-user PECK scheme, which provides an efficient remote storage system for a number of users. We define the security model for a multi-user PECK scheme and provide an efficient construction using our PECK scheme. Our multi-user system is also secure under the DLDH assumption in the random oracle model. We provide specific applications such as encrypted file sharing system and email routing system for a multi-user PECK scheme. Our multi-user PECK scheme optimizes the storage of the server for a number of users and achieves an efficient computation and communication overhead.

Organization of The Paper. The remainder of this paper is organized as follows. In Section 2, we define models for the PECK scheme and review the bilinear map and hardness assumptions for our constructions. In Section 3, we construct a simple and efficient PECK scheme and prove the security of our scheme in the random oracle model. In Section 4, we analyze the security of the previous schemes and compare our scheme with them. Then we introduce a multi-user PECK scheme and provide the specific applications in Section 5. Finally, we give concluding remarks in Section 6.

2 Preliminaries

Before presenting our results we briefly formalize models and definitions of security for the PECK scheme. In addition, we review the definition of bilinear groups and describe the decisional linear Diffi-Hellman (DLDH) assumption for our schemes.

2.1 Generic Model for PECK

We consider that a user's encrypted data is outsourced in the storage of the untrusted server, such as email gateway, secure audit logs, and remote database server. In a public key model for keyword search, the server stores encrypted data collected from third parties and the user enables the server to retrieve emails containing keywords, which the user wants to search, without leaking information. To support a conjunctive keyword search, we recall the assumptions used in the previous works [22,26]; (1) The same keyword never appears in two different keyword fields. (2) Every keyword field is defined for every document.

The assumptions can be easily satisfied in practical applications. One simple way to enforce the assumptions is to embed field names within keywords, for example, as "`Field Name`.w" where '.' denotes concatenation and w is an actual keyword. Consider the routing email system. Then we can define the field names as `To, From, Subject, Time` etc. In this case, the keyword is used as "`From`.Alice", and "`To`.Null" can be used for a field that does not have a valid keyword. Then, even if the same keywords are used for different keyword fields, a collision of keywords does not happen. To simplify the notations, we will use w instead of "`Field Name`.w." We construct the system under these assumptions.

The public key encryption with conjunctive keyword search consists of 4 polynomial time algorithms, (`KeyGen`, `PECK`, `Trapdoor`, `Test`) such that;

- `KeyGen`: takes as input a security parameter and returns `params` (system parameters) and the public/private key pair (pk, sk).
- `PECK`: is executed by the sender to encrypt a keyword set $W = \{w_1, \ldots, w_\ell\}$. It produces a searchable keyword encryption S of W with the public key pk.
- `Trapdoor`: takes as input the secret key sk and the keyword query $Q = \{I_1, \ldots, I_m, w_{I_1}, \ldots, w_{I_m}\}$ for $m \leq \ell$ where I_i is an index to denote a location of w_{I_i}, and returns a trapdoor T_Q for the conjunctive search of a given keyword query.
- `Test`: is executed by the server to search the documents with the keywords of a trapdoor T_Q. It takes as input the public key pk, the searchable keyword encryption S, and the trapdoor T_Q. Then output '1' if S includes Q and '0' otherwise.

Actually, in the PECK scheme, to send a message m with keyword set W, a ciphertext has a form of $\langle \texttt{Enc}(pk, m), \texttt{PECK}(pk, W) \rangle$ where $\texttt{Enc}(\cdot)$ is a secure public key encryption. We concentrate on searchable encryption part.

2.2 Adversarial Models for PECK

We consider a semantic security against chosen keyword attacks as mentioned in the previous works [22,4,26]. Golle, Staddon, and Waters first defined the security games of symmetric encryptions for conjunctive keyword search [22]. Then in [26] Park, Kim, and Lee modified the security games for public key setting. The security of the PECK scheme can be defined by two security games, *indistinguishability of ciphertext from ciphertext* (IND-CC-CKA) and *indistinguishability of ciphertext from random* (IND-CR-CKA) against chosen keyword attacks. We briefly define them. The IND-CC-CKA game is as follows

- **Setup:** The challenger takes a security parameter 1^k and runs the `Keygen` algorithm. The public key pk and the system parameters `params` are given to $\mathcal{A}$. The challenger $\mathcal{C}$ keeps the secret key sk to itself.
- **Phase 1:** $\mathcal{A}$ adaptively queries a number of keyword sets, $Q_1, \ldots, Q_d$, to trapdoor oracle as follows.
  - Trapdoor Queries $\langle Q_i \rangle$. The challenger runs $\texttt{Trapdoor}(sk, Q_i)$ and generates the trapdoor T_{Q_i}. And then responds to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ selects two target keyword sets W_0 and W_1, and sends them to the challenger $\mathcal{C}$. The challenger picks a random bit $\beta \in \{0, 1\}$. The only restriction is that W_0 and W_1 should not be distinguished from trapdoors issued in previous phase. And it sets $S_\beta = \text{PECK}(pk, W_\beta)$. Then send it to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ additionally queries keyword sets, $Q_{d+1}, \ldots, Q_\gamma$, to trapdoor oracle.
  - Trapdoor Queries $\langle Q_i \neq W_0 \text{ or } W_1 \rangle$. The challenger runs $\texttt{Trapdoor}(sk, Q_i)$ and generates the trapdoor T_{Q_i}. If T_{Q_i} cannot distinguish for W_0 and W_1, then it responds T_{Q_i} to $\mathcal{A}$.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$. It wins the game if $\beta = \beta'$.

We define the advantage of the adversary $\mathcal{A}$ against the PECK scheme as the function of the security parameter 1^k: $\text{Adv}^{\text{IND-CC-CKA}}_{\mathcal{PECK},\mathcal{A}}(1^k) = |\Pr[\beta = \beta'] - \frac{1}{2}|$.

We introduce another security game IND-CR-CKA which is a variant of the IND-CC-CKA game. This security game is the same as the IND-CC-CKA game except for the **Challenge** phase. While in the IND-CC-CKA game the adversary $\mathcal{A}$ selects two target keyword sets, W_0 and W_1, and gives them to the challenger $\mathcal{C}$, in the IND-CR-CKA game $\mathcal{A}$ selects a target keyword set W_0 and gives it to $\mathcal{C}$. Then $\mathcal{C}$ selects a random keyword set R and sets $W_1 = R$. The IND-CR-CKA game is as follows.

- **Setup:** The challenger takes a security parameter 1^k and runs the `Keygen` algorithm. The public key pk and the system parameters `params` are given to $\mathcal{A}$. The challenger $\mathcal{C}$ keeps the secret key sk to itself.
- **Phase 1:** $\mathcal{A}$ adaptively queries a number of keyword sets, $Q_1, \ldots, Q_d$, to trapdoor oracle as follows.
  - Trapdoor Queries $\langle Q_i \rangle$. The challenger runs $\texttt{Trapdoor}(sk, Q_i)$ and generates the trapdoor T_{Q_i}. And then responds to $\mathcal{A}$.

- **Challenge:** $\mathcal{A}$ selects a target keyword set W^* and sends it to the challenger $\mathcal{C}$. The challenger selects a random keyword set R and picks a random bit $\beta \in \{0,1\}$. The only restriction is that W^* should not be distinguished for R from trapdoors issued in previous phase. And it sets $S_\beta = \text{PECK}(pk, W_\beta)$ where $W_0 = W^*$ and $W_1 = R$. Then send $\langle S_\beta, W_0, W_1 \rangle$ to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ additionally queries keyword sets, $Q_{d+1}, \ldots, Q_\gamma$, to trapdoor oracle.
  - Trapdoor Queries $\langle Q_i \neq W_0 \text{ or } W_1 \rangle$. The challenger runs $\mathsf{Trapdoor}(sk, Q_i)$ and generates the trapdoor T_{Q_i}. If T_{Q_i} cannot distinguish for W_0 and W_1, then it responds T_{Q_i} to $\mathcal{A}$.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. It wins the game if $\beta = \beta'$.

We define the advantage of the adversary $\mathcal{A}$ against the PECK scheme as the function of the security parameter 1^k: $\text{Adv}^{\text{IND-CR-CKA}}_{\mathcal{PECK},\mathcal{A}}(1^k) = |\Pr[\beta = \beta'] - \frac{1}{2}|$. The two security games, IND-CC-CKA and IND-CR-CKA, are all asymptotically equivalent [22,4].

Definition 1. *We say that a PECK scheme $\mathcal{PECK}$ is (t, q_t, ϵ)-secure if for any t-time IND-CC-CKA (respectively IND-CR-CKA) adversary $\mathcal{A}$ who makes at most q_t trapdoor queries, we have that* $\text{Adv}^{\text{atk}}_{\mathcal{PECK},\mathcal{A}}(1^k) < \epsilon$ *where* atk *is IND-CC-CKA (resp. IND-CR-CKA).*

2.3 Bilinear Maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be the two multiplicative cyclic groups of order p for some large prime p. Our schemes make use of the *bilinear* map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ between these two groups. The *bilinear* map should be satisfied the following properties:

1. **Bilinear:** We say that a map $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is bilinear if $\hat{e}(g^a, h^b) = \hat{e}(g,h)^{ab}$ for all $g, h \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p^*$.
2. **Non-degenerate:** The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Observe that since $\mathbb{G}_1$, $\mathbb{G}_2$ are groups of prime order this implies that if g is a generator of $\mathbb{G}_1$ then $\hat{e}(g, g)$ is a generator of $\mathbb{G}_2$.
3. **Computable:** There is an efficient algorithm to compute $\hat{e}(g, h)$ for any $g, h \in \mathbb{G}_1$.

A bilinear map satisfying the three properties above is said to be an *admissible* bilinear map. We can make this map using the Weil pairing or the Tate pairing [5,6,12,27].

2.4 Complexity Assumption

The security of our schemes is based on a complexity assumption called Decision Linear Diffie-Hellman (DLDH) assumption. This assumption was introduced by Boneh *et al.*[10] and was recently used to construct a short group signature [10] and an efficient tag-based encryption scheme [23]. Let g_1, g_2, g_3 be random elements in $\mathbb{G}_1$ and a, b, c elements in $\mathbb{Z}_p^*$. We define the DLDH problem in the $\mathbb{G}_1$

as; Given 6 tuples $\langle g_1, g_2, g_3, g_1^a, g_2^b, g_3^c \rangle \in \mathbb{G}_1$ as input, output 1 if $c = a+b$ and 0 otherwise. One can easily show that an algorithm for solving the DLDH problem in $\mathbb{G}_1$ gives an algorithm for solving DDH in $\mathbb{G}_1$. The converse is believed to be false. That is, it is believed that the DLDH problem is a hard problem even in the bilinear groups where DDH is easy [10,23]. [1] We define the advantage of an algorithm $\mathcal{A}$ to deciding the DLDH problem in $\mathbb{G}_1$ as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{DLDH}} = \left| \begin{array}{l} \Pr[g_1, g_2, g_3 \leftarrow_R \mathbb{G}_1; a, b \leftarrow_R \mathbb{Z}_p^* : \mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, g_3^{a+b}) = 1] \\ \quad -\Pr[g_1, g_2, g_3, z \leftarrow_R \mathbb{G}_1; a, b \leftarrow_R \mathbb{Z}_p^*; z = g_3^c : \\ \qquad\qquad \mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, z) = 1] \end{array} \right|.$$

Definition 2. *We say that the (t, ϵ)-DLDH assumption holds in $\mathbb{G}_1$ if no t-time adversary has an advantage at least ϵ in solving the DLDH problem in $\mathbb{G}_1$.*

3 Proposed PECK Scheme

We introduce a new and simple public key encryption with conjunctive keyword search. However, our scheme has efficient communication and storage overhead in both the server and users and reasonable computation overhead. In addition, we provide a concrete security proof of our scheme, while other previous works have had a weakness in their security proofs. Our scheme is as follows.

- `KeyGen`(1^k): Given the a security parameter 1^k, it returns `params` $= (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, H_1(\cdot), H_2(\cdot), g)$ where H_1:$\{0,1\}^{\log w} \rightarrow \mathbb{G}_1$ and H_2:$\{0,1\}^{\log w} \rightarrow \mathbb{G}_1$ are two different collision-resistance hash functions and g is a generator of $\mathbb{G}_1$. And it picks a random value x in $\mathbb{Z}_p^*$ and computes $y = g^x$. The public/private key pair $(pk,\ sk)$ is given by

$$(pk, sk) = (y, x)$$

- `PECK`$(pk,\ W)$: The sender selects a keyword set $W = \{w_1, \ldots, w_\ell\}$ and two random values s, r in $\mathbb{Z}_p^*$, and computes $A = g^r, B = y^s, C_i = h_i^r f_i^s$ for $1 \le i \le \ell$ where $h_i = H_1(w_i)$ and $f_i = H_2(w_i)$. Then outputs $S = \langle A, B, C_1, \ldots, C_\ell \rangle$.
- `Trapdoor`$(sk,\ Q)$: It selects a random value $t \in \mathbb{Z}_p^*$ and computes $T_{Q,1} = g^t$, $T_{Q,2} = (h_{I_1} \cdots h_{I_m})^t$, and $T_{Q,3} = (f_{I_1} \cdots f_{I_m})^{t/x}$ where $Q = \{I_1, \ldots, I_m, w_{I_1}, \ldots, w_{I_m}\}$. Then output $T_Q = (T_{Q,1},\ T_{Q,2}, T_{Q,3}, I_1, \ldots, I_m)$.
- `Test`(pk, S, T_Q): Check $\hat{e}(T_{Q,1}, \prod_{i=1}^m C_{I_i}) = \hat{e}(A, T_{Q,2}) \cdot \hat{e}(B, T_{Q,3})$.

Our scheme does not require a pairing operation for a searchable encryption and a trapdoor. To test the equality of a keyword set, 3 pairing operations are needed. In addition, a ciphertext consists of $(\ell+2)$ group elements in $\mathbb{G}_1$ and the number of the private keys stored in each user's secure device is only one. These complexities are an interesting improvement to the other existing systems. We will discuss this in Section 4.

[1] In [10], it was shown that the DLDH problem is hard in generic bilinear groups.

3.1 Security

We now prove that our scheme is IND-CR-CKA secure over the DLDH assumption in the random oracle model.

Theorem 3. *Let our scheme be $\mathcal{PECK}$. Assume that (t, ϵ)-DLDH assumption holds in $\mathbb{G}_1$. Then $\mathcal{PECK}$ is $(t', q_t, q_h, e\ell q_t\epsilon)$-secure against IND-CR-CKA in the random oracle model for arbitrary ℓ, q_t, q_h, and $t' < t - \Theta(\tau \ell q_t q_h)$ where τ is the maximum time for an exponentiation in $\mathbb{G}_1$*

Proof. Assume that $\mathcal{A}$ is an adversary with advantage ϵ' in breaking $\mathcal{PECK}$ against IND-CR-CKA and $H_1(\cdot)$, $H_2(\cdot)$ are modelled as random oracles. Then we can construct an adversary $\mathcal{B}$ who attacks the DLDH problem using $\mathcal{A}$ as described below.

- **Setup:** The DLDH challenger gives the DLDH parameters $\langle g_1, g_2, g_3, v_1, v_2, v_3\rangle$ to the adversary $\mathcal{B}$ where $v_1 = g_1^a$, $v_2 = g_2^b$, and $v_3 = g_3^{a+b}$ or z. $\mathcal{B}$ sets $g = g_1$ and $y = g_2$. At that time, the private key is regarded as α where $g_2 = g_1^\alpha$. In addition, it selects a random value η in $\mathbb{Z}_p^*$ and keeps it securely. This value is used in random oracles and **Challenge** phase. It gives $\mathcal{A}$ the system parameters $\texttt{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, H_1(\cdot), H_2(\cdot), g)$ and the public key $pk = y$.
  - H_1, H_2-queries: $\mathcal{A}$ issues at most q_h keyword queries to the random oracles. It simultaneously responds these queries. $\mathcal{B}$ maintains lists of tuples $\langle w_i, c_i, h_i, d_i\rangle$ called the H_1^{list} and $\langle w_i, c_i, f_i, e_i\rangle$ called the H_2^{list}. If the keyword w_i is already queried, $\mathcal{B}$ returns $h_i = H_1(w_i)$ in H_1^{list} and $f_i = H_2(w_i)$ in H_2^{list}. Otherwise, it generates a random coin $c_i \in \{0, 1\}$ so that $\Pr[c_i = 1] = 1/(\ell q_t)$. If $c_i = 0$, then it selects two random values d_i and e_i in $\mathbb{Z}_p^*$, and sets $g_1^{d_i}$ for h_i and $g_2^{e_i}$ for f_i. Otherwise, it selects a random value d_i in $\mathbb{Z}_p^*$ and computes $e_i = d_i/\eta$. And then it sets $g_3^{d_i}$ for h_i and $g_3^{e_i}$ for f_i. It adds $\langle w_i, c_i, h_i, d_i\rangle$ and $\langle w_i, c_i, f_i, e_i\rangle$ to H_1^{list} and H_2^{list} respectively, and returns h_i, f_i to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ queries a number of keyword sets to trapdoor oracle.
  - Trapdoor Queries: $\mathcal{A}$ queries a keyword set $Q_i = \{I_{i,1}, \ldots, I_{i,m}, w_{i,1}, \ldots, w_{i,m}\}$ to obtain a trapdoor T_{Q_i}. Let $\mathcal{I}_i$ be $\{I_{i,1}, \ldots, I_{i,m}\}$. $\mathcal{B}$ obtains two lists such that $\langle w_{i,j}, c_{i,j}, h_{i,j}, d_{i,j}\rangle$ and $\langle w_{i,j}, c_{i,j}, f_{i,j}, e_{i,j}\rangle$ for $1 \le j \le m$ by running the above algorithm for responding to H_1, H_2-queries. If there is any $c_{i,j} = 1$ for $1 \le j \le \ell$, then $\mathcal{B}$ aborts. Otherwise, it selects a random value t_i in $\mathbb{Z}_p^*$ and outputs $T_{Q_i} = (T_{Q_i,1}, T_{Q_i,2}, T_{Q_i,3})$ where $T_{Q_i,1} = g_1^{t_i}$, $T_{Q_i,2} = g_1^{t_i(\sum_{j=1}^m d_{i,j})}$, and $T_{Q_i,3} = g_1^{t_i(\sum_{j=1}^m e_{i,j})}$.
- **Challenge:** $\mathcal{A}$ selects a target document W^* and sends it to $\mathcal{B}$. It selects a random document R and sets $W_0 = W^*$ and $W_1 = R$ where $W_0 = \{w_{0,1}, \ldots, w_{0,\ell}\}$ and $W_1 = \{w_{1,1}, \ldots, w_{1,\ell}\}$. The only restriction is that W_0 and W_1 should not be distinguished from trapdoors issued in previous

phase.[2] And then it selects a random bit $\beta \in \{0,1\}$. $\mathcal{B}$ queries all keywords of W_β to H_1, H_2-oracles and obtains lists $\langle w_{\beta,i}, c_{\beta,i}, h_{\beta,i}, d_{\beta,i}\rangle$ and $\langle w_{\beta,i}, c_{\beta,i}, f_{\beta,i}, e_{\beta,i}\rangle$ for all i from oracles. If there is no $c_{\beta,i} = 1$ for all i, it aborts. Otherwise, it computes a challenge ciphertext $S_\beta = \langle A, B, C_{\beta,1}, \ldots, C_{\beta,\ell}\rangle$ where $A = v_1$, $B = v_2^\eta$, and $C_{\beta,i} = v_1^{d_{\beta,i}} v_2^{e_{\beta,i}\eta}$ (in case that $c_{\beta,i} = 0$) or $v_3^{d_{\beta,i}}$ (in case that $c_{\beta,i} = 1$). $\mathcal{A}$ is given $\langle S_\beta, W_0, W_1\rangle$.

- **Phase 2:** $\mathcal{A}$ continues to issue trapdoor queries which are not equal to W_0 and W_1. The only restriction is that it cannot issue trapdoor query distinguishing W_0 for W_1. $\mathcal{B}$ responds as in **Phase 1**.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. If $\beta = \beta'$, then $\mathcal{B}$ outputs 1 meaning $v_3 = g_3^{a+b}$. Otherwise, it outputs 0 meaning $v_3 = z$.

The adversary $\mathcal{B}$ should not abort in Trapdoor Queries and **Challenge** phase for the success. The probability that it does not abort in Trapdoor Queries is $(1 - 1/\ell q_t)^{\ell q_t}$ for q_t queries. And the probability that it does not abort in **Challenge** phase is $(1 - (1 - \frac{1}{\ell q_t})^\ell)$. Therefore, the probability that it did not abort during the simulation is greater than $1/(4 \cdot \ell q_t)$ because $(1 - 1/\ell q_t)^{\ell q_t} \geq 1/4$ for any $\ell q_t \geq 2$ and $(1 - (1 - \frac{1}{\ell q_t})^\ell) \geq (1 - (1 - \frac{1}{\ell q_t})) = \frac{1}{\ell q_t}$. If $v_3 = g_3^{a+b}$, then the challenge ciphertext is a valid encryption of the keyword set W_β.

$$
\begin{aligned}
A &= v_1 = g_1^a = g^a, \\
B &= v_2^\eta = g_2^{b\eta} = y^{b\eta} \\
\text{(If } c_{\beta,i} = 0\text{,) } C_{\beta,i} &= v_1^{d_{\beta,i}} v_2^{e_{\beta,i}\eta} = g_1^{d_{\beta,i}a} g_2^{e_{\beta,i}b\eta} = h_{\beta,i}^a f_{\beta,i}^{b\eta} \\
\text{(Otherwise,) } C_{\beta,i} &= v_3^{d_{\beta,i}} = g_3^{(a+b)d_{\beta,i}} = (g_3^{d_{\beta,i}})^a (g_3^{d_{\beta,i}/\eta})^{b\eta} \\
&= (h_{\beta,i})^a (g_3^{e_{\beta,i}})^{b\eta} = h_{\beta,i}^a f_{\beta,i}^{b\eta}
\end{aligned}
$$

In this case, $\mathcal{A}$'s view is identical to its view in a real attack game and it must satisfy $|\Pr[\beta = \beta'] - 1/2| \geq \epsilon'$. If $v_3 = z$ and $\mathcal{B}$ does not abort, then $\Pr[\beta = \beta'] = 1/2$. Therefore, we have that

$$
\begin{aligned}
|\Pr[\mathcal{B}(g_1, g_2, g_3, g_1^a, g_2^b, g_3^{a+b}) = 0] - \Pr[\mathcal{B}(g_1, g_2, g_3, g_1^a, g_2^b, z) = 0]| \\
\geq \tfrac{1}{4\ell q_t} |(1/2 \pm \epsilon') - 1/2| = \tfrac{\epsilon'}{4\ell q_t}
\end{aligned}
$$

We complete the proof of the theorem. □

4 Discussions

In this section, we discuss the security problems of the previous systems and compare our system to them. We will analyze and compare the PKL schemes because they have the most efficient system complexity to support conjunctive keyword search in a public key setting.

[2] In the simulation, the only restriction is that the target keyword sets W_0 and W_1 should not be distinguished from trapdoor queries. Namely, the adversary cannot issue a trapdoor query for a keyword set $Q_i \subset W_0$ or W_1. However, the adversary should be allowed to query a keyword set Q_i where $Q_i \cap W_{0(\text{or}1)} \neq \emptyset$ and $Q_i \not\subseteq W_{0(\text{or}1)}$, while in the security proof of the PKL II scheme all trapdoor queries including a word in W_0 and W_1 are restricted. Our simulation allows that with some probability.

4.1 Flaws in Security Proofs of Previous Schemes

The PKL schemes proved their security under the indistinguishability of limited ciphertext from random (ILCR) game. The scenario of their security proof is not correct, although the security game ILCR appears to be reasonable. In the ILCR game, the adversary $\mathcal{A}$ selects the target keyword w^* and its position κ, and then returns (w^*, κ) to the challenger $\mathcal{C}$. And then, $\mathcal{A}$ is given the challenge ciphertext S_β with $W_0 = \{w_{0,1}, \ldots, w_{0,\ell}\}$ and $W_1 = \{w_{1,1}, \ldots, w_{1,\ell}\}$ where $w^* = w_{0,\kappa}$. After receiving the target ciphertext, $\mathcal{A}$ issues the trapdoor queries. Here, the only restriction is that it cannot ask any trapdoor query that can distinguish W_0 for W_1. For this, the proof of [26] blocks all trapdoor queries including w^*. Although a trapdoor query for a document $W_i = \{w_{i,1}, \ldots, w_{i,\ell}\}$ with $w_{i,\kappa} = w^*$ cannot distinguish W_0 for W_1 if $W_i \nsubseteq W_0$, this query cannot be issued. It is that it violates a rule of the security game. The challenger has to respond to a legitimate trapdoor query with some probability. However, the challenger aborts whenever it receives such a trapdoor query in the security proof of the PKL schemes.

We show that the PKL I scheme is broken in a real attack environment because of this flaw. After the adversary asks trapdoor queries two times, it can check if a document has the keyword he selects. Consider the system with two keyword fields. The adversary first obtains two trapdoors by issuing queries for $\{w_1, w_2\}$ and $\{w_2\}$. Then it can search the documents with $\{w_1\}$ by use of two trapdoors. Assume that the adversary selects $\{w_1, w_3\}$ as a target document and is given $\langle S_\beta, W_0 = \{w_1, w_3\}, W_1 = \{w_4, w_5\}\rangle$ as a target ciphertext in a security game. The adversary can distinguish S_β for W_0 or W_1. The adversary's behavior is appropriate because the trapdoors for $\{w_1, w_2\}$ and $\{w_2\}$ cannot distinguish between W_0 and W_1. We provide a detailed description for a real attack in Appendix A.

The security proof of the PKL II scheme has more restrictions. After receiving the challenge ciphertext $\langle S_\beta, W_0, W_1\rangle$, the adversary $\mathcal{A}$ cannot issue any trapdoor query including any keyword of W_0. It cannot query even the trapdoor for a document including $w_{0,i}$ for any i as well as a target keyword w^*. In an example with three keyword fields, when W_0 is $\{w_{0,1}, w_{0,2}, w_{0,3}\}$ and $w^* = w_{0,2}$, the adversary is not able to ask the trapdoor for a keyword set like $\{w_{0,1}, w'_2, w'_3\}$, though $w'_2 \neq w_{0,2}$, $w'_3 \neq w_{0,3}$, and the trapdoor for this keyword set cannot distinguish target documents. This proof is not appropriate. Hence, the security of the PKL II scheme is difficult to be guaranteed in a practical environment since it does not have a reasonable security proof.[3]

4.2 Efficiency

In this section, we compare the efficiency of our scheme with previous works. As shown in the previous section, our scheme has a very short ciphertext size

[3] That the security proof is not complete does not mean that the PKL II scheme is broken. However, there might be a security hole from this flaw as shown in the PKL I scheme.

Table 1. Comparison of the PECK schemes

	PKL I [26]	PKL II [26]	Our system
Ciphertext size	$2L_1 + \ell \cdot L_2$	$(2\ell + 3)L_1$	$(\ell + 2)L_1$
Private key size	$2\|\mathbb{Z}_p^*\|$	$(\ell + 2)\|\mathbb{Z}_p^*\|$	$\|\mathbb{Z}_p^*\|$
Trapdoor size	$L_1 + \|\mathbb{Z}_p^*\|$	$2L_1 + \|\mathbb{Z}_p^*\|$	$3L_1$
Encryption	$(\ell + 2) \cdot \mathtt{e} + \ell \cdot (\mathtt{p} + \mathtt{H})$	$(3\ell + 2) \cdot \mathtt{e}$	$(2\ell + 2) \cdot \mathtt{e} + 2\ell \cdot \mathtt{H}$
Trapdoor	$\mathtt{e} + \ell \cdot \mathtt{H}$	2e	$3\mathtt{e} + 2\ell \cdot \mathtt{H}$
Test	$\mathtt{e} + \mathtt{p}$	$\mathtt{e} + 2\mathtt{p}$	3p
Security	insecure	incomplete	complete

L_1 (or L_2): a bit length of an element of $\mathbb{G}_1$ (or $\mathbb{G}_2$).
p: a pairing operation.
e: an exponentiation in $\mathbb{G}_1$ or $\mathbb{G}_2$.
H: an admissible encoding hash function.

and one private key. In addition, it has very efficient overhead for the storage of both the server and users. In particular, the ciphertext size of our scheme has an interesting result since the server should store a lot of ciphertexts for a number of users. We can use the fact that $\mathbb{G}_1$ is of size approximately 2^{170}, elements in $\mathbb{G}_1$ are 171-bit strings, and the discrete log in $\mathbb{G}_1$ is as hard as the discrete log in $\mathbb{Z}_q$ where q is 1020 bits [10,15,28]. At that time, $\mathbb{G}_2$ is a subgroup of the multiplicative group of the finite field $F_{p^\eta}^*$ where p is the size of a group $\mathbb{G}_1$ and η is the *embedding degree* of map, and the elliptic curve with $\eta = 6$ is generally used for the balance of two maps $\mathbb{G}_1$ and $\mathbb{G}_2$ [10,15]. In this case, the ciphertext size of the PKL I scheme is about six times as large as that of our scheme. In addition, the PKL I scheme requires a pairing computation linear to the number of keyword fields in the encryption phase. To reduce the ciphertext size and the computation overhead, the PKL II scheme uses different private keys and random values for every keyword fields. Hence, the PKL II scheme requires $(\ell+2)$ private keys for each user and $(2\ell + 3)$ group elements of $\mathbb{G}_1$ for a ciphertext. In comparison, our scheme uses only one private key and the ciphertext size is half that of the PKL II scheme. Table 1 shows the system complexities of the PKL scheme and our scheme. Here, we omit miscellaneously small computation operations such as a multiplication in $\mathbb{G}_1$ or $\mathbb{G}_2$, and an addition in $\mathbb{Z}_p^*$. If a is a string, then $|a|$ denotes its length, while if A is a finite set then $|A|$ denotes its size.

The PKL I scheme has an efficient test algorithm and the PKL II scheme has an efficient computation overhead with respect to users, while they consume a large amount of storage in the server. However, the most significant problem about the PKL schemes is they cannot guarantee the concrete security in real applications as shown in Section 4.1. Our scheme provides a complete security proof as well as a short ciphertext size and a private key. In addition, our computation overhead is comparable to the PKL schemes.

5 Multi-user PECK System

In this section, we extend our PECK scheme to a multi-user PECK scheme. In a PECK model, a user outsources his data to the storage of a server which stores data for a number of users. Suppose user Alice wants to send an encrypted email to a number of recipients in the example of an email routing system. To route the email to devices of recipients in the email gateway, Alice would encrypt the same document and the same keywords with the public keys of the recipients repeatedly. A multi-user PECK scheme can eliminate repeated operations and reduce the communication overhead for a number of users. In the public key model and the identity-based model, the systems for a multi-receiver have been investigated to reduce the communication and computation cost [7,8,9,10]. However, there is no PECK scheme for the multi-user even though the multi-user PECK scheme is more important with respect to storage and communication overhead since all of the ciphertexts must be stored in the storage of the server. In this paper, we propose the first multi-user PECK scheme.

We first define a model for the multi-user public key encryption with conjunctive keyword search (mPECK) and a security game for it. The mPECK is modelled by combining the multi-receiver public key encryption and the PECK scheme. Let n be the number of users. The mPECK consists of 4 polynomial time algorithms, (`KeyGen`, `mPECK`, `Trapdoor`, `Test`) such that;

- `KeyGen`: takes as input a security parameter and returns `params` (system parameters) and the public/private key pairs $(pk_1, sk_1), \ldots, (pk_n, sk_n)$.
- `mPECK`: is executed by the sender to encrypt an keyword set $W = \{w_1, \ldots, w_\ell\}$. It produces a searchable keyword encryption S of W with the public keys $(pk_1, \ldots, pk_n)$.
- `Trapdoor`: takes as input the secret key sk_j and the keyword query $Q = \{I_1, \ldots, I_m, w_{I_1}, \ldots, w_{I_m}\}$ for $m \leq \ell$ where I_i is an index to denote a location of w_{I_1}, and returns a trapdoor $T_{j,Q}$ for the conjunctive search of a given keyword query.
- `Test`: is executed by the server to search the documents with the keywords of a trapdoor $T_{j,Q}$. It takes as input the public key pk_j, the searchable keyword encryption S, and the trapdoor $T_{j,Q}$. It outputs '1' if S includes Q and '0' otherwise.

Adversarial Model. We define the security game for mPECK system. We consider a semantic security against chosen keyword attacks. In our security game, the adversary is given the public keys for n users in the **Setup** phase, the keyword set is encrypted with n public keys. In addition, the adversary issues the trapdoor query with a user index. The security of the mPECK scheme can be also defined by indistinguishability of ciphertext from random against chosen keyword attacks (IND-mCR-CKA) as follows.

– **Setup:** The challenger takes a security parameter k and runs the `Keygen` algorithm. The public keys $(pk_1, \ldots, pk_n)$ and the system parameters `params` are given to $\mathcal{A}$. The challenger $\mathcal{C}$ keeps the secret keys $(sk_1, \ldots, sk_n)$ to itself.

– **Phase 1:** $\mathcal{A}$ adaptively queries a number of keyword sets, $Q_1, \ldots, Q_d$, to trapdoor oracle as follows where q_i is $\langle j, Q_i \rangle$.
  - Trapdoor Queries $\langle j, Q_i \rangle$. The challenger runs $\texttt{Trapdoor}(sk_j, Q_i)$ and generates the trapdoor T_{j,Q_i}. And then responds to $\mathcal{A}$.
– **Challenge:** $\mathcal{A}$ selects a target keyword set W^* and sends it to the challenger $\mathcal{C}$. The challenger selects a random keyword set R and picks a random bit $\beta \in \{0,1\}$. The only restriction is that W^* should not be distinguished for R from trapdoors issued in previous phase. And it sets $S_\beta = \text{PECK}(pk_1, \ldots, pk_n, W_\beta)$ where $W_0 = W^*$ and $W_1 = R$. Then send $\langle S_\beta, W_0, W_1 \rangle$ to $\mathcal{A}$.
– **Phase 2:** $\mathcal{A}$ additionally queries keyword sets, $Q_{d+1}, \ldots, Q_\gamma$, to trapdoor oracle.
  - Trapdoor Queries $\langle j, Q_i \neq W_0 \text{ or } W_1 \rangle$. The challenger runs $\texttt{Trapdoor}(sk_j, Q_i)$ and generates the trapdoor T_{j,Q_i}. If T_{j,Q_i} cannot distinguish for W_0 and W_1, then it responds T_{j,Q_i} to $\mathcal{A}$.
– **Guess:** Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. He wins the game if $\beta = \beta'$.

We define the advantage of the adversary $\mathcal{A}$ against the mPECK scheme as the function of the security parameter 1^k: $\text{Adv}^{\text{IND-mCR-CKA}}_{\mathcal{MPECK},\mathcal{A}}(1^k) = \mid \Pr[\beta = \beta'] - \frac{1}{2} \mid$.

Definition 4. *We say that a mPECK scheme $\mathcal{MPECK}$ is (t, q_t, ϵ)-secure if for any t-time IND-mCR-CKA adversary $\mathcal{A}$ who makes at most q_t trapdoor queries, we have that* $\text{Adv}^{\text{IND-mCR-CKA}}_{\mathcal{MPECK},\mathcal{A}}(1^k) < \epsilon$.

5.1 Construction

We construct a first mPECK scheme. Our scheme uses a technique called *randomness re-use* [9,24] to improve the computation and communication overhead. Our scheme is as follows.

- $\texttt{KeyGen}(1^k)$: Given the a security parameter 1^k, it returns $\texttt{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, H_1(\cdot), H_2(\cdot), g)$ where H_1:$\{0,1\}^{\log w} \rightarrow \mathbb{G}_1$ and H_2:$\{0,1\}^{\log w} \rightarrow \mathbb{G}_1$ are two different collision-resistance hash functions and g is a generator of $\mathbb{G}_1$. It chooses n random values $x_1, \ldots, x_n$ in $\mathbb{Z}_p^*$ and computes $y_i = g^{x_i}$. The public/private key pairs $(pk_1, sk_1), \ldots, (pk_n, sk_n)$ are given by

$$(pk_i, sk_i) = (y_i, x_i).$$

- $\texttt{mPECK}(pk_1, \ldots, pk_n, W)$: The sender selects a keyword set $W = \{w_1, \ldots, w_\ell\}$ and two random values s, r in $\mathbb{Z}_p^*$, and computes $A = g^r, B_j = y_j^s$ for $1 \leq j \leq n$, $C_i = h_i^r f_i^s$ for $1 \leq i \leq \ell$ where $h_i = H_1(w_i)$ and $f_i = H_2(w_i)$. Then outputs $S = \langle A, B_1, \ldots, B_n, C_1, \ldots, C_\ell \rangle$.
- $\texttt{Trapdoor}(sk_j, Q)$: It selects a random value $t \in \mathbb{Z}_p^*$ and computes $T_{j,Q1} = g^t$, $T_{j,Q2} = (h_{I_1} \cdots h_{I_m})^t$, and $T_{j,Q3} = (f_{I_1} \cdots f_{I_m})^{t/x_j}$ where $Q = \{I_1, \ldots, I_m, w_{I_1}, \ldots, w_{I_m}\}$. Then output $T_{j,Q} = (T_{j,Q1}, T_{j,Q2}, T_{j,Q3}, I_1, \ldots, I_m)$.
- $\texttt{Test}(pk_j, S, T_Q)$: Check $\hat{e}(T_{j,Q1}, \prod_{i=1}^m C_{I_i}) = \hat{e}(A, T_{j,Q2}) \cdot \hat{e}(B_j, T_{j,Q3})$.

In this scheme, a mechanism generating a trapdoor for searching keywords and `Test` algorithm are the same as those of our PECK scheme with respect to a recipient and the server. To send an encrypted message with conjunctive keyword search to n users, the sender has only to add B_i from the recipient's public keys. Generally, an encrypted data with the conjunctive keyword search is stored by a form of $\langle \texttt{Enc}(pk_i, m), \texttt{PECK}(pk_i, W)\rangle$ in the storage of the server where `Enc` is a secure public encryption scheme. Therefore, when a user wants to send the same message to n users by a general PECK scheme, he should generate n (public key) encryptions as $\langle \texttt{Enc}(pk_i, m), \texttt{PECK}(pk_i, W)\rangle$ for $1 \le i \le n$ and the server should separately store ciphertexts for each user. However, in case of our mPECK scheme, the sender can encrypt a message for n users by additionally computing B_j parts of encryption and a multi-user public key encryption can be used for encryption of an *actual* message m. In this case, the server can store the ciphertext as $\langle \texttt{mEnc}(pk_1, \ldots, pk_n, m), \texttt{mPECK}(pk_1, \ldots, pk_n, W)\rangle$ where `mEnc` is a secure multi-receiver public key encryption scheme. If we apply our mPECK scheme to multi-receiver encryption scheme of ElGamal [20] type, we can construct an efficient multi-user system as

- $\texttt{mEnc}(y_1, \ldots, y_n,\ m)$+$\texttt{mPECK}(y_1, \ldots, y_n,\ W)$: Let $h : \mathbb{G}_2 \to \mathcal{M}$ be one-way hash function. The sender selects two random values s, r in $\mathbb{Z}_p^*$, and computes $E = h(\hat{e}(g,g)^{rs}) \oplus m$, $A = g^r, B_j = y_j^s$ for $1 \le j \le n$, $C_i = h_i^r f_i^s$ for $1 \le i \le \ell$ where $h_i = H_1(w_i)$ and $f_i = H_2(w_i)$. Then outputs (E, S) where $S = \langle A, B_1, \ldots, B_n, C_1, \ldots, C_\ell\rangle$.
- $\texttt{mDec}(x_j,\ E, B_j)$: It computes $X_j = h(\hat{e}(A, B_j)^{1/x_j})$ and outputs $E \oplus X_j$. If a user is a legitimate receiver, he outputs m since $X_j = h(\hat{e}(A, B_j)^{1/x_j}) = h(\hat{e}(g^r, y_j^s)^{1/x_j}) = h(\hat{e}(g, y_j^{1/x_j})^{rs}) = h(\hat{e}(g,g)^{rs})$.

In this example, the server searches the encryption with the keywords, that a user u_j wants to search, by trapdoor of u_j and then returns (A, B_j, E) to u_j. The user u_j can decrypt this ciphertext with his private key. The server can store $|\texttt{C}_n| + (n + \ell + 1) \cdot L_1$ data instead of $n(|\texttt{C}_n| + (\ell + 2) \cdot L_1)$ for n users where $\texttt{C}_n = \texttt{mEnc}(pk_1, \ldots, pk_n, m)$. This is very efficient complexity with respect to the storage of the server and the communication between a user and the server.

5.2 Security

We now prove that our scheme is IND-mCR-CKA secure over the DLDH assumption in the random oracle model.

Theorem 5. *Let our scheme be $\mathcal{MPECK}$. Assume that (t, ϵ)-DLDH assumption holds in $\mathbb{G}_1$. Then $\mathcal{PECK}$ is $(t', q_t, q_h, e\ell q_t \epsilon)$-secure against IND-mCR-CKA in the random oracle model for arbitrary ℓ, q_t, q_h, and $t' < t - \Theta(\tau \ell q_t q_h)$ where τ is the maximum time for an exponentiation in $\mathbb{G}_1$*

Proof. Assume that $\mathcal{A}$ is an adversary with advantage ϵ' in breaking $\mathcal{MPECK}$ against IND-mCR-CKA and $H_1(\cdot)$, $H_2(\cdot)$ are modelled as random oracles.

Then we can construct an adversary $\mathcal{B}$ who attacks the DLDH problem using $\mathcal{A}$ as described below.

- **Setup:** The DLDH challenger gives the DLDH parameters $\langle g_1, g_2, g_3, v_1, v_2, v_3\rangle$ to the adversary $\mathcal{B}$ where $v_1 = g_1^a$, $v_2 = g_2^b$, and $v_3 = g_3^{a+b}$ or z. The adversary $\mathcal{B}$ randomly selects n values $\pi_1, \ldots, \pi_n$ in $\mathbb{Z}_p^*$, and sets $g = g_1$ and $y_1 = g_2^{\pi_1}, \ldots, y_n = g_2^{\pi_n}$. At that time, the private key for a user u_i is regarded as $\alpha\pi_i$ where $g_2 = g_1^\alpha$. In addition, it selects a random value η in $\mathbb{Z}_p^*$ and keeps it securely. This value is used in random oracles and **Challenge** phase. It gives $\mathcal{A}$ the system parameters $\texttt{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, H_1(\cdot), H_2(\cdot), g)$ and the public key $pk_1 = y_1, \ldots, pk_n = y_n$.
  - H_1, H_2-queries: These oracles are the same as those of Theorem 1.
- **Phase 1:** $\mathcal{A}$ queries a number of keyword sets to trapdoor oracle.
  - Trapdoor Queries $\mathcal{A}$ queries a user's index j and a keyword set $Q_i = \{I_{i,1}, \ldots, I_{i,m},\ w_{i,1}, \ldots, w_{i,m}\}$ to obtain a trapdoor T_{j,Q_i}. We denote $\{I_{i,1}, \ldots, I_{i,m}\}$ by $\mathcal{I}_i$. $\mathcal{B}$ obtains two lists such that $\langle w_{i,j}, c_{i,j}, h_{i,j}, d_{i,j}\rangle$ and $\langle w_{i,j}, c_{i,j}, f_{i,j}, e_{i,j}\rangle$ for $1 \le j \le m$ by running the above algorithm for responding to H_1, H_2-queries. If there is any $c_{i,j} = 1$ for $1 \le j \le \ell$, then $\mathcal{B}$ aborts. Otherwise, it selects a random value t_i in $\mathbb{Z}_p^*$ and outputs $T_{j,Q_i} = (T_{j,Q_{i,1}}, T_{j,Q_{i,2}}, T_{j,Q_{i,3}})$ where $T_{j,Q_{i,1}} = g_1^{t_i}$, $T_{j,Q_{i,2}} = g_1^{t_i(\sum_{j=1}^m d_{i,j})}$, and $T_{j,Q_{i,3}} = g_1^{t_i(\sum_{j=1}^m e_{i,j})/\pi_j}$.
- **Challenge:** $\mathcal{A}$ selects a target document W^* and sends it to $\mathcal{B}$. It selects a random document R and sets $W_0 = W^*$ and $W_1 = R$ where $W_0 = \{w_{0,1}, \ldots, w_{0,\ell}\}$ and $W_1 = \{w_{1,1}, \ldots, w_{1,\ell}\}$. The only restriction is that W_0 and W_1 should not be distinguished from trapdoors issued in previous phase. And then it selects a random bit $\beta \in \{0, 1\}$. $\mathcal{B}$ queries all keywords of W_β to H_1, H_2-oracles and obtains lists $\langle w_{\beta,i}, c_{\beta,i}, h_{\beta,i}, d_{\beta,i}\rangle$ and $\langle w_{\beta,i}, c_{\beta,i}, f_{\beta,i}, e_{\beta,i}\rangle$ for all i from oracles. If there is no $c_{\beta,i} = 1$ for all i, it aborts. Otherwise, it computes a challenge ciphertext $S_\beta = \langle A, B_1, \ldots, B_n, C_{\beta,1}, \ldots, C_{\beta,\ell}\rangle$ where $A = v_1$, $B_j = v_2^{\pi_j\eta}$, and $C_{\beta,i} = v_1^{d_{\beta,i}} v_2^{e_{\beta,i}\eta}$ (in case that $c_{\beta,i} = 0$) or $v_3^{d_{\beta,i}}$ (in case that $c_{\beta,i} = 1$). $\mathcal{A}$ is given $\langle S_\beta, W_0, W_1\rangle$.
- **Phase 2:** $\mathcal{A}$ continues to issue trapdoor queries which are not equal to W_0 and W_1. The only restriction is that it cannot issue trapdoor query distinguishing W_0 for W_1. $\mathcal{B}$ responds as in **Phase 1**.
- **Guess:** Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$. If $\beta = \beta'$, then $\mathcal{B}$ outputs 1 meaning $v_3 = g_3^{a+b}$. Otherwise, it outputs 0 meaning $v_3 = z$.

The adversary $\mathcal{B}$ should not abort in Trapdoor Queries and **Challenge** phase for the success. The probability that it does not abort in Trapdoor Queries is $(1 - 1/\ell q_t)^{\ell q_t}$ for q_t queries. And the probability that it does not abort in **Challenge** phase is $(1-(1-\frac{1}{\ell q_t})^\ell)$. Therefore, the probability that it did not abort during the simulation is greater than $1/(4 \cdot \ell q_t)$ because $(1 - 1/\ell q_t)^{\ell q_t} \ge 1/4$ for any

$\ell q_t \geq 2$ and $(1-(1-\frac{1}{\ell q_t})^{\ell}) \geq (1-(1-\frac{1}{\ell q_t})) = \frac{1}{\ell q_t}$. If $v_3 = g_3^{a+b}$, then the challenge ciphertext is a valid encryption of the keyword set W_β.

$$\begin{aligned} A^* &= v_1 = g_1^a = g^a \\ B_j^* &= v_2^{\pi_j \eta} = g_2^{\pi_j b\eta} = y_j^{b\eta} \\ \text{(If } c_{\beta,i}=0\text{,) } C_{\beta,i} &= v_1^{d_{\beta,i}} v_2^{e_{\beta,i}\eta} = g_1^{d_{\beta,i}a} g_2^{e_{\beta,i}b\eta} = h_{\beta,i}^a f_{\beta,i}^{b\eta} \\ \text{(Otherwise,) } C_{\beta,i} &= v_3^{d_{\beta,i}} = g_3^{(a+b)d_{\beta,i}} = (g_3^{d_{\beta,i}})^a (g_3^{d_{\beta,i}/\eta})^{b\eta} \\ &= (h_{\beta,i})^a (g_3^{e_{\beta,i}})^{b\eta} = h_{\beta,i}^a f_{\beta,i}^{b\eta} \end{aligned}$$

In this case, $\mathcal{A}$'s view is identical to its view in a real attack game and it must satisfy $|\Pr[\beta = \beta'] - 1/2| \geq \epsilon'$. If $v_3 = z$ and $\mathcal{B}$ does not abort, then $\Pr[\beta = \beta'] = 1/2$. Therefore, we have that

$$\begin{aligned} |\Pr[\mathcal{B}(g_1, g_2, g_3, g_1^a, g_2^b, g_3^{a+b}) = 0] - \Pr[\mathcal{B}(g_1, g_2, g_3, g_1^a, g_2^b, z) = 0]| \\ \geq \tfrac{1}{4\ell q_t} |(1/2 \pm \epsilon') - 1/2| = \tfrac{\epsilon'}{4\ell q_t} \end{aligned}$$

We complete the proof of the theorem. □

5.3 Applications

In this section, we describe how the mPECK system can be used for specific applications. The first useful application is an encrypted file sharing system in a remote server for a number of users and the second application is an email routing system for a multi-recipient.

Encrypted File Sharing System. An encrypted file system allows users to store encrypted files on an untrusted remote server. If an encryptor wants a file to be shared to n users, it would be encrypted with the public keys of all n users. In this case, a large storage in the server and computation overhead for users might be required. To remedy this limitation, public key broadcast encryption [14] and multi-user systems [8,9] can be good solutions. However, users cannot download select files they want to retrieve if searches on encrypted data are not provided. A user would receive all of his data from the server and decrypt them. This procedure is very cumbersome. Therefore, an efficient conjunctive keyword search on the encrypted data in a multi-user system is a very useful technique for an encrypted file sharing system.

Users may want to share data only with particular users in many cases. Our mPECK scheme can be efficiently used for this application. When storing data, a user first selects the users with whom he wants to share his data and encrypts the data using the mPECK scheme with their public keys. To search the encrypted files that he wants to retrieve, a user makes a trapdoor which includes some keywords and then queries it to the server. The server can return the encrypted files that the user wants to retrieve. This system has an advantage in managing the storage of a server in comparison to an ordinary PECK scheme. Our mPECK system requires only $\omega|\mathtt{C}_n| + \omega(n+\ell+1)\cdot L_1$ storage of a server for ω documents of n users, while the `PKL I` and `PKL II` schemes respectively need $\omega n(|\mathtt{C}_n| + 2L_1 + \ell \cdot L_2)$ and $\omega n(|\mathtt{C}_n| + (2\ell+3)\cdot L_1)$ storage.

Email Routing in Email Gateway. Boneh *et al.* introduced that a public key encryption with keyword search can be efficiently applied to a routing service in an email gateway [14] and Park *et al.* stated that the conjunctive keyword searchable encryption is better suited for an email routing system [26]. An email routing system is a good example for using a multi-recipient since we often need to email a large number of recipients. To send an email to a number of users, the sender should encrypt the same email many times and append different searchable encryptions for routing the email to each user in email gateway. It requires not only a large computation overhead, but also a large communication overhead. If our example introduced in Section 5.1 is applied to this application, we could achieve very efficient system complexity. The email gateway implements a searching algorithm with $(A, B_i, C_1, \ldots, C_\ell)$ for a user u_i and route an encrypted email (A, B_i, E) for u_i. In this system, we can share an encryption part E of an email body m and searchable keyword encryption parts $A, C_1, \ldots, C_\ell$. Therefore, the length of the encrypted email with searchable keywords in our system is $|\mathtt{C}_n| + |\mathtt{S}| + (n-1) \cdot L_1$ for n users, while that in a PECK scheme is $n(|\mathtt{C}_1| + |\mathtt{S}|)$, where $\mathtt{S}$ is a ciphertext of searchable keyword encryption in a single-user system.

6 Concluding Remarks

We have presented the efficient and secure PECK scheme with a short ciphertext size and one private key. In addition, we have shown that our scheme has a concrete security proof under the IND-CR-CKA game, while other previous works have a weakness in security proof. Our PECK scheme was extended to a PECK system for supporting a multi-user. In [19], Curtmola *et al.* introduced a searchable symmetric encryption for a multi-user, which makes use of a single-user searchable encryption and a broadcast encryption. Their concept is slightly different from ours since their scheme was constructed in a symmetric key setting. The owner of the documents in their scheme gives users permission to search the documents and manage a group of authorized users. Whenever the group of users changes, the owner would have to update the documents and broadcast a new key for the group. Their scheme cannot support a conjunctive keyword search. We have provided specific applications for our mPECK system.

Acknowledgement

The authors would like to thank Dong Jin Park for helpful discussions at early stages of this paper and Yeon-Hyeong Yang for his help in preparing the final version of this paper. In addition, this research was supported by the KT R&D, grant No. R01-2005-000-10713-0 from the Basic Research Program of the KOSEF, the Brain Korea 21 Project in 2007, and the MIC of Korea under the ITRC support program supervised by the IITA(IITA-2006-C1090-0603-0026).

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searhable 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.: Efficient Multi-receiver Identity-Based Encryption and Its Application to Broadcast Encryption. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 380–397. Springer, Heidelberg (2005)
3. Ballard, L., Green, M., De Medeiros, B., Monrose, F.: Correlation Resistant Storage. Technical Report TR-SP-BGMM-050507, Johns Hopkins University Department of Computer Science (2005)
4. Ballard, L., Kamara, S., Monrose, F.: Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. In: Qing, S., Mao, W., Lopez, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2004)
5. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithm 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., 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 (2003)
7. Baudron, O., Pointcheval, D., Stern, J.: Extended Notions of Security for Multicast Public Key Cryptosystems. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 499–511. Springer, Heidelberg (2000)
8. Bellare, M., Boldyreva, A., Micali, S.: Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (2000)
9. Bellare, M., Boldyreva, A., Staddon, J.: Randomness Re-use in Multi-recepient Encryption Schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2003)
10. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatrues. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
11. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keywrod Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
12. 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)
13. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–342. Springer, Heidelberg (2005)
14. 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)
15. Boneh, D., Shacham, H., Lynn, B.: Short Signatures from the Weil Pairing. Journal of Cryptology 17(4), 297–319 (2004) (Extended abstract in ASIACRYPT 2001, LNCS, vol. 2248, pp. 514–532, Springer, Heidelberg, 2001)

16. 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)
17. Byun, J.W., Lee, D.H., Lim, J.: Efficient Conjunctive Keyword Searches on Encrypted Data Storage System. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 184–196. Springer, Heidelberg (2004)
18. Chang, Y.C., Mitzenmacher, M.: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
19. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: ACM CCS 2006, pp. 79–88. ACM Press, New York (2007)
20. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information on Information Theory 31, 469–472 (1985)
21. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003)
22. Golle, P., Staddon, J., Waters, B.: Secure Conjunctive Keyword Search Over Encrypted Data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
23. 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)
24. Kurosawa, K.: Multi-Recepient Public-Key Encryption with Shortend Ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Heidelberg (2002)
25. Lenstra, A., Verheul, E.R.: Selecting cryptographic key sizes. Journal of Cryptology 14(4), 255–293 (2001)
26. Park, D.J., Kim, K., Lee, P.J.: Public Key Encryption with Conjunctive Field Keyword Search. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 73–86. Springer, Heidelberg (2004)
27. Joux, A.: The Weil and Tate Pairing as Building Blocks for Public Key Cryptosystems. In: Fieker, C., Kohel, D.R. (eds.) Algorithmic Number Theory. LNCS, vol. 2369, pp. 20–32. Springer, Heidelberg (2002)
28. Miyaji, A., Nakabayashi, M., Takano, S.: New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. IEICE Trans. Fundamentals E84-A(5), 1234–1243 (2001)
29. Shamir, A.: How to Share a Secret. Communications of the ACM 22, 612–613 (1979)
30. Song, D., Wagner, D., Perrig, A.: Practical Techniques for Searching on Encrypted Data. In: IEEE Symposium on Research in Security and Privacy 2000, pp. 44–55. IEEE Computer Society Press, Los Alamitos (2000)
31. Waters, B., Balfanz, D., Durfee, G., Smetters, D.: Building an Encrypted and Searchable Audit Log. In: NDSS 2004, The internet society (2004)

A Real Attack on PKL I

We first review the PKL I scheme. Let $H_1{:}\{0,1\}^{\log w} \rightarrow \mathbb{G}_1$ be a collision-resistance hash function. Then the PKL I scheme is as follows;

- KeyGen(1^k): Given the a security parameter 1^k, it selects a random generator g in $\mathbb{G}_1$ and two random values x_1 and x_2 in $\mathbb{Z}_p^*$. Then the public/private key pair $(pk,\ sk)$ are given by

$$pk = (y_1, y_2),\ sk = (x_1, x_2)$$

  where $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$.
- PECK$(pk,\ W)$: The sender selects a keyword set $W = \{w_1, \ldots, w_\ell\}$ and a random value r in $\mathbb{Z}_p^*$, and computes $A = g^r, B = y_2^r, C_i = \hat{e}(h_i^r, y_1)$ for $1 \leq i \leq \ell$ where $h_i = H_1(w_i)$. Then it outputs $S = \langle A, B, C_1, \ldots, C_\ell \rangle$.
- Trapdoor(sk, Q): It selects a random value $t \in \mathbb{Z}_p^*$ and computes $T_{Q,1} = t$ and $T_{Q,2} = (\prod_1^m h_{I_i})^{x_1/(x_2+t)}$. Then output $T_Q = (T_{Q,1}, T_{Q,2}, T_{Q,3}, I_1, \ldots, I_m)$.
- Test(pk, S, T_Q): Check $\prod_{i=1}^m C_{I_i} = \hat{e}(B + A^{T_{Q,1}}, T_{Q,2})$.

We recall the example of Section 4.1. Assume that the adversary $\mathcal{A}$ received target ciphertext $\langle S^*, W_0 = \{w_1, w_3\}, W_1 = \{w_4, w_5\} \rangle$. $\mathcal{A}$ issues trapdoor queries for $\{w_1, w_2\}$ and $\{w_2\}$, and then obtains the trapdoors $T_{w_1,w_2} = (t, (H_1(w_1)\ H_1(w_2))^{x_1/(x_2+t)})$ and $T_{w_2} = (t', H_1(w_2)^{x_1/(x_2+t')})$. As mentioned in Section 4.1, the trapdoor queries are appropriate because they cannot distinguish between W_0 and W_1. Let S^* be $\langle A^*, B^*, C_1^*, C_2^* \rangle$. Then we can check if S^* is a searchable keyword encryption of W_0 or not by the equality check of the following equation.

$$\hat{e}((H_1(w_1)H_1(w_2))^{x_1/(x_2+t)}, B^* + A^{*t}) / \hat{e}(H_1(w_2)^{x_1/(x_2+t')}, B^* + A^{*t'}) = C_1^* \quad (1)$$

We assume that S^* is a searchable keyword encryption of W_0. Then we can represent it as $A^* = g^r, B^* = y_2^r$, $C_1^* = \hat{e}(H_1(w_1)^r, y_1)$, $C_2^* = \hat{e}(H(w_3)^r, y_1)$. We show that the equality check is correct;

$$\begin{aligned}
&\hat{e}((H_1(w_1)H_1(w_2))^{x_1/(x_2+t)}, B^* + A^{*t}) / \hat{e}(H_1(w_2)^{x_1/(x_2+t')}, B^* + A^{*t'}) \\
&= \hat{e}((H_1(w_1)H_1(w_2))^{x_1/(x_2+t)}, g^{r(x_2+t)}) / \hat{e}(H_1(w_2)^{x_1/(x_2+t')}, g^{r(x_2+t')}) \\
&= \hat{e}((H_1(w_1)H_1(w_2))^{x_1}, g^r) / \hat{e}(H_1(w_2)^{x_1}, g^r) \\
&= \hat{e}(H_1(w_1)^{x_1}, g^r) \\
&= \hat{e}(H_1(w_1)^r, y_1)
\end{aligned}$$

Therefore, S^* is a searchable keyword encryption of W_0 if (1) is correct and that of W_1 otherwise.

Practical Time Capsule Signatures in the Standard Model from Bilinear Maps

Benoît Libert* and Jean-Jacques Quisquater

UCL, Microelectronics Laboratory, Crypto Group
Place du Levant, 3, B-1348, Louvain-La-Neuve, Belgium
{benoit.libert,jean-jacques.quisquater}@uclouvain.be

Abstract. At FC'05, Dodis and Yum introduced a new cryptographic tool called *time capsule signature* (TCS) which allows signers to generate "future signatures" that only become valid from a specific future time t (chosen at signature generation) when a trusted entity (called *Time Server*) discloses some trapdoor information for period t. In addition, time capsule signatures endow signers with the ability to make their signatures valid before the pre-determined time t. Full signatures that were completed by their original issuer should be indistinguishable from those that automatically became valid after the release of the time-specific trapdoor. Time capsule signatures were showed to be generically constructible from another primitive called *identity-based trapdoor hard-to-invert relation* (ID-THIR). The only known instantiations of the latter either rely on the idealized random oracle model or are too inefficient for real-world applications. In this paper, we devise the first efficient ID-THIR (and thus TCS) construction which is secure in the standard model (i.e. without the random oracle heuristic).

Keywords: time capsule signatures, standard model, bilinear maps.

1 Introduction

In 2005, Dodis and Yum introduced the concept of time capsule signatures [17]. Such a primitive allows signers to generate signatures that only become valid from a future moment t when a trusted party (called Time Server) discloses a trapdoor information associated with period t. This is accomplished in such a way that:

- Anyone can directly ascertain that a "future signature" will indeed become effective at time t.
- In a "pre-hatching operation", the legal signer can decide to make her future signature valid at any time before the pre-determined moment t.
- A signature that was not opened by the signer automatically becomes valid (which is called "hatching" as opposed to "pre-hatching") at time t when

* This author acknowledges the DGTRE's First Europe Program of the Walloon Region in Belgium for his financial support.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 23–38, 2007.

the Time Server publishes the relevant trapdoor information Z_t allowing signature holders to complete future signatures generated for that period.
– The Time Server does not have to interact with any user at any time or know anything about the PKI employed by signers.

Regardless of whether a signature was previously opened by the signer or if it was automatically completed after the release of the trapdoor Z_t at time t, no one can tell how it became valid: "pre-hatched" signatures should be indistinguishable from "hatched" signatures.

Similarly to time release primitives described in [5,15,31], time capsule signatures (TCS) follow the server-based approach which allows preparing messages for a definite future and departs from "time-lock puzzle" methods addressing related problems [33,2,10,30,20,21]. They imply a minimal assumption on the Time Server that only has to publish some piece of information at the beginning of each time period and never has to contact users.

In [17], Dodis and Yum gave proper security definitions for TCS schemes and showed how to generically construct them using newly defined primitives called *identity-based trapdoor hard-to-invert relations* (ID-THIRs). They also described a generic construction of ID-THIR which yields very efficient implementations in the random oracle model [4] but is much less efficient in the standard model. These results proved the existence of time capsule signatures in the random oracle model assuming the availability of one-way functions and their existence in the standard model if trapdoor one-way permutations exist.

Our contribution. The generic construction of ID-THIR given in [17] relies on non-interactive witness-indistinguishable [18] proofs of knowledge. Before the recent advances of Groth, Ostrovsky and Sahai [26,27] in NIZK and witness indistinguishable proofs, the best known methods [34] for constructing such proofs in the standard model were very inefficient. Those dramatic improvements were adapted [28] so as to provide constant-size - though impractical - group signatures in the standard model. They could be applied to the present context as well, but resulting implementations would remain too inefficient for practical use. To date, the only practical examples of TCS schemes resort to the random oracle methodology [4] which is known [13] to *only* provide heuristic arguments.

The achievement of this paper is to describe a very simple and efficient identity-based trapdoor hard-to-invert relation which is not generic but is secure in the standard model. It utilizes the Waters signature [36] which is known to be secure in the standard model assuming the hardness of the Diffie-Hellman problem in groups equipped with bilinear maps. More precisely, our ID-THIR turns out to be somehow related to identity-based [35] extensions [12,32] of Waters signatures. This is not very surprising since the generic ID-THIR of [17] was already making use of proofs of knowledge of signatures (which are nothing but identity-based signatures). The technical difficulty was here to avoid witness indistinguishable proofs. To do so, our implementation takes advantage of tricks which date back to [7] and that were used to prove the security of the signature in [36]. Thanks to the generic transformation of [17], our ID-THIR gives rise to

the first practical time capsule signature scheme which is secure in the standard model (under a well-studied computational assumption).

Organization. In the forthcoming sections, we first recall functional definitions and security models for identity-based trapdoor hard-to-invert relations and time capsule signatures. Section 3 then describes our practical construction of ID-THIR. Its possible optimizations are discussed in section 4 and the resulting concrete TCS scheme is analyzed in section 5.

2 Preliminaries

2.1 Identity-Based Trapdoor Hard-to-Invert Relations

A binary relation R is a subset of $\{0,1\}^* \times \{0,1\}^*$ and the language $\mathcal{L}_R$ is the set of elements α for which there exist β such that $(\alpha, \beta) \in R$. The relation R must be completely specified by a short description D_R. Besides, for all pairs $(\alpha, \beta) \in R$, the length $|\beta|$ of β has to be bounded by a polynomial in $|\alpha|$. Lastly, it should be easy to decide whether a given α lies in $\mathcal{L}_R$.

Definition 1. *An identity-based trapdoor hard-to-invert relation* (ID-THIR) *is a family of binary relations $\mathcal{R} = \{R_{id} | id \in I_{\mathcal{R}}\}$, where $I_{\mathcal{R}}$ is a finite set of indices, that are all trapdoor hard-to-invert relations. Namely, for each $id \in I_{\mathcal{R}}$, sampling a lock/proof pair $(c, d) \in R_{id}$ is easy but finding a proof for a given lock is hard without knowing the specific trapdoor td_{id}. A master trapdoor $\mathsf{mtd}_{\mathcal{R}}$ allows extracting a trapdoor td_{id} for each relation $R_{id} \in \mathcal{R}$. An* ID-THIR *is entirely specified by a 5-uple of algorithms* (Gen, Sample, Check, Extract, Invert) *such that:*

- Gen*: given a security parameter k, this algorithm generates $\mathcal{R} = \{R_{id} | id \in I_{\mathcal{R}}\}$ and returns its description $D_{\mathcal{R}}$ and its master trapdoor $\mathsf{mtd}_{\mathcal{R}}$.*
- Sample*: takes as input $(D_{\mathcal{R}}, id)$ and returns a randomly sampled lock/proof pair $(c, d) \in R_{id}$.*
- Check*: verifies the validity of a lock/proof pair (c, d). It returns 1 (accept) if $(c, d) \in R_{id}$ and 0 (reject) otherwise.*
- Extract*: is used to extract the trapdoor of each relation. Given $id \in I_{\mathcal{R}}$ and the master trapdoor $\mathsf{mtd}_{\mathcal{R}}$, it returns the trapdoor td_{id} for the relation R_{id}.*
- Invert*: allows finding a proof d for a given lock $c \in \mathcal{L}_{R_{id}}$ using the trapdoor td_{id}. If $c \in \mathcal{L}_{R_{id}}$, $\mathsf{Invert}_{\mathsf{td}_{id}}(c)$ outputs a proof d such that $(c, d) \in R_{id}$.*

Let $(c, d) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(id)$ and $\tilde{d} \leftarrow \mathsf{Invert}_{\mathsf{td}_{id}}(c)$. The correctness property imposes that $\mathsf{Check}_{D_{\mathcal{R}}, id}(c, d) = \mathsf{Check}_{D_{\mathcal{R}}, id}(c, \tilde{d}) = 1$. The ambiguity is the computational indistinguishability of (c, d) and $(c, \tilde{d})$ even knowing $\mathsf{mtd}_{\mathcal{R}}$. Besides, an ID-THIR *is said one-way if the following probability is negligible for any PPT algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:*

$$\Pr[\mathsf{Check}_{D_{\mathcal{R}}, id^\star}(c, \hat{d}) = 1 \ \wedge \ id^\star \notin Query(\mathcal{A}, O_{\mathsf{Extract}}) \mid (D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}}) \leftarrow \mathsf{Gen}(k); \\ (id^\star, st) \leftarrow \mathcal{A}_1^{O_{\mathsf{Extract}}}(D_{\mathcal{R}}); \ (c, d) \leftarrow \mathsf{Sample}_{D_{\mathcal{R}}}(id^\star); \ \hat{d} \leftarrow \mathcal{A}_2^{O_{\mathsf{Extract}}}(D_{\mathcal{R}}, c, st)]$$

where O_{Extract} *is an oracle simulating the trapdoor extraction algorithm* Extract, $Query(\mathcal{A}, O_{\mathsf{Extract}})$ *is the set of queries made by* $\mathcal{A}$ *to the latter oracle and st stands for the state information passed by* $\mathcal{A}_1$ *to* $\mathcal{A}_2$*. The soundness property states that the following property is negligible for any algorithm* $\mathcal{B}$*:*

$$\Pr[\mathsf{Check}_{D_{\mathcal{R},id^\star}}(c,\tilde{d}) = 0 \ \wedge \ R_{id^\star} \in \mathcal{R} \ \wedge \ c \in \mathcal{L}_{R_{id^\star}} \mid (D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}}) \leftarrow \mathsf{Gen}(k);$$
$$(c, id^\star) \leftarrow \mathcal{B}(D_{\mathcal{R}}); \ \mathsf{td}_{id^\star} \leftarrow \mathsf{Extract}_{\mathsf{mtd}_{\mathcal{R}}}(id^\star); \ \tilde{d} \leftarrow \mathsf{Invert}_{\mathsf{td}_{id^\star}}(c)]$$

An ID-THIR *is said secure if it meets the above four requirements.*

Intuitively, the one-wayness property captures that it should be computationally infeasible to open a given lock without the trapdoor of the corresponding relation even after having seen trapdoors for polynomially-many other relations. The soundness is the impossibility of coming up with a lock (for some relation) that cannot be opened into a valid lock/proof pair using the relevant trapdoor.

Dodis and Yum showed in [17] that an ID-THIR exists in the random oracle model if a one-way function exists. Their construction relies on the Fiat-Shamir heuristic [19] and non-interactive witness indistinguishable [18] proofs of knowledge. Instead of a Fiat-Shamir like proof, their method can be implemented with non-interactive witness indistinguishable proofs of knowledge (with a common reference string) that do not involve random oracles. However, the best known technique [34] for constructing such proofs uses trapdoor one-way permutations and is very inefficient. Therefore the existence of identity-based trapdoor hard-to-invert relations in the standard model, which requires the existence of trapdoor one-way permutations [17], is currently mainly of theoretical interest.

2.2 Time Capsule Signatures

Definition 2. *A time capsule signature (*TCS*) consists of a* 8*-uple of PPT algorithms* $(\mathsf{Setup}^{\mathsf{TS}}, \mathsf{Setup}^{\mathsf{User}}, \mathsf{TSig}, \mathsf{TVer}, \mathsf{TRelease}, \mathsf{Hatch}, \mathsf{PreHatch}, \mathsf{Ver})$.

- $\mathsf{Setup}^{\mathsf{TS}}$*: is an algorithm run by the Time Server. Given a security parameter* k*, it returns a public/private key pair* $(\mathsf{TPK}, \mathsf{TSK})$.
- $\mathsf{Setup}^{\mathsf{User}}$*: is run by each signer. Given a security parameter* k*, it returns a public/private key pair for the signer* $(\mathsf{PK}, \mathsf{SK})$.
- TSig*: is the time capsule signature generation algorithm. It takes as input* $(m, \mathsf{SK}, \mathsf{TPK}, t)$*, where* t *is the time from which the signature becomes valid. It produces a future signature* σ'_t.
- TVer*: is the time capsule signature verification algorithm. It takes as input a* 5*-uple* $(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t)$ *and returns either* 1 *(accept) or* 0 *(reject).*
- TRelease*: is the time release algorithm run by the Time Server. At the beginning of period* t*, it uses* TSK *to compute and publish* $Z_t = \mathsf{TRelease}(t, \mathsf{TSK})$*. Note that the Time Server never interacts with any user at any time.*
- Hatch*: is run by any party to open a valid time capsule signature that became mature. Given* $(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t)$ *and the time-specific trapdoor* Z_t *as inputs, it returns a hatched signature* σ_t.

$\mathsf{PreHatch}$*: is run by the signer to open a valid time capsule signature which is not mature yet. It takes as input* $(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t)$ *and the signer's private key* SK *as inputs and outputs a pre-hatched signature* σ_t.

Ver*: is used to verify hatched or pre-hatched signatures. Given* $(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t)$*, it returns* 1 *(accept) or* 0 *(reject).*

The correctness imposes that $\mathsf{TVer}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{PK}, \mathsf{TPK}, t) = 1$ *and* $\mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t) = 1$ *if* $\sigma_t = \mathsf{Hatch}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{PK}, \mathsf{TPK}, Z_t)$ *or* $\sigma_t = \mathsf{PreHatch}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{SK}, \mathsf{TPK})$*. Ambiguity requires the distribution of "hatched signatures"* $\sigma_t = \mathsf{Hatch}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{PK}, \mathsf{TPK}, Z_t)$ *to be computationally indistinguishable from that of "pre-hatched signatures"* $\sigma_t = \mathsf{PreHatch}(m, \mathsf{TSig}(m, \mathsf{SK}, \mathsf{TPK}, t), \mathsf{SK}, \mathsf{TPK})$ *even knowing* TSK*.*

As explained in [17], the security of time capsule signatures is defined in three aspects: security against the signer, the verifier and the Time Server. In the following notation O_{TSig} is an oracle simulating the time capsule signature generation algorithm TSig, O_{TR} denotes an oracle simulating the time release algorithm $\mathsf{TRelease}$ and O_{PreH} stands for the pre-hatching oracle emulating $\mathsf{PreHatch}$. Given (m, t) as input, O_{TSig} returns a time capsule signature σ'_t generated on behalf of the signer. Oracle O_{PreH} takes (m, t, σ'_t) as input and outputs the signer's pre-hatched signature σ_t.

Security against the signer. This definition means that the signer should be unable to produce a time capsule signature which looks good to the verifier but cannot be hatched into a full signature by the Time Server. More formally, any PPT adversary $\mathcal{A}$ should have negligible advantage in this experiment.

$$\begin{aligned}
\mathsf{Setup}^{\mathsf{TS}}(k) &\rightarrow (\mathsf{TPK}, \mathsf{TSK}) \\
(m, t, \sigma'_t, \mathsf{PK}) &\leftarrow \mathcal{A}^{O_{\mathsf{TR}}}(\mathsf{TPK}) \\
Z_t &\leftarrow \mathsf{TRelease}(t, \mathsf{TSK}) \\
\sigma_t &\leftarrow \mathsf{Hatch}(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, Z_t) \\
Adv(\mathcal{A}) &= \Pr[\mathsf{TVer}(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t) = 1 \ \wedge \ \mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t) = 0]
\end{aligned}$$

Security against the verifier. Informally, the verifier must be unable to open a future signature without the help of the signer or the Time Server. We require any PPT adversary $\mathcal{B}$ to have negligible advantage in the next experiment.

$$\begin{aligned}
\mathsf{Setup}^{\mathsf{TS}}(k) &\rightarrow (\mathsf{TPK}, \mathsf{TSK}) \\
\mathsf{Setup}^{\mathsf{User}}(k) &\rightarrow (\mathsf{PK}, \mathsf{SK}) \\
(m, t, \sigma_t) &\leftarrow \mathcal{B}^{O_{\mathsf{TR}}, O_{\mathsf{TSig}}, O_{\mathsf{PreH}}}(\mathsf{TPK}, \mathsf{PK}) \\
Adv(\mathcal{A}) &= \Pr[\mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t) = 1 \ \wedge \ t \notin Query(\mathcal{B}, O_{\mathsf{TR}}) \\
&\qquad \wedge \ (m, t, .) \notin Query(\mathcal{B}, O_{\mathsf{PreH}})]
\end{aligned}$$

where $Query(\mathcal{B}, O_{\mathsf{TR}})$ is the set of queries made to the time release oracle O_{TR} and $Query(\mathcal{B}, O_{\mathsf{PreH}})$ denotes the set of *valid* queries to O_{PreH} (i.e. queries (m, t, σ'_t) for which $\mathsf{TVer}(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t) = 1$).

Security against the Time Server. Obviously, the Time Server should not be able to produce a valid hatched or pre-hatched signature full signature on a message m without obtaining a time capsule signature on m from the signer. Any PPT adversary $\mathcal{C}$ must have negligible advantage in the following experiment.

$$\begin{aligned}
&\mathsf{Setup}^{\mathsf{TS}^*}(k) \rightarrow (\mathsf{TPK}, \mathsf{TSK}^*)\\
&\mathsf{Setup}^{\mathsf{User}}(k) \rightarrow (\mathsf{PK}, \mathsf{SK})\\
&\quad (m, t, \sigma_t) \leftarrow \mathcal{C}^{O_{\mathsf{TSig}}, O_{\mathsf{PreH}}}(\mathsf{PK}, \mathsf{TPK}, \mathsf{TPK}^*)\\
&\qquad Adv(\mathcal{C}) = \Pr[\mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t) = 1 \ \wedge \ (m, .) \notin Query(\mathcal{C}, O_{\mathsf{TSig}})]
\end{aligned}$$

where $\mathsf{Setup}^{\mathsf{TS}^*}$ denotes a run of $\mathsf{Setup}^{\mathsf{TS}}$ by a dishonest Time Server, TSK^* is $\mathcal{C}$'s state after this malicious key generation and $Query(\mathcal{C}, O_{\mathsf{TSig}})$ stands for the set of queries to the time capsule signature oracle O_{TSig}.

2.3 Bilinear Maps

Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear map groups* if there is a mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

1. bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}$;
2. efficient computability for any input pair;
3. non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

The protocol that we have in mind relies on the intractability of the following well-studied problem in bilinear map groups.

Definition 3. *The* **Computational Diffie-Hellman Problem** *(CDH) in a group $\mathbb{G} = \langle g \rangle$ is to compute g^{ab} given (g^a, g^b). An algorithm (τ, ε)-breaks the CDH assumption if it solves a CDH instance with probability ε in time τ.*

2.4 The Waters Signature

We recall the description of the signature scheme of [36] which is existentially unforgeable in the standard model under the CDH assumption in bilinear map groups. In the description hereafter, messages are assumed to be encoded as bitstrings of length n. In practice however, a collision-resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ can be applied to sign longer messages.

$\mathsf{Keygen}(k, n)$: choose bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^k$. Randomly pick $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, as well as $g, g_2 \stackrel{R}{\leftarrow} \mathbb{G}$ and a vector $\overline{u} = (u', u_1, \ldots, u_n) \in \mathbb{G}^{n+1}$ of random group elements. The public key is $\mathsf{PK} = (n, \mathbb{G}, \mathbb{G}_T, g, g_1, g_2, \overline{u}, W)$ with $g_1 = g^\alpha$ and $W = e(g_1, g_2)$. The private key is $\mathsf{SK} = \alpha$.

$\mathsf{Sign}(\mathsf{m}, \alpha)$: parse m as $m_1 \ldots m_n$ with $m_i \in \{0,1\}$ for all $i \in \{1, \ldots, n\}$. A signature of m is produced by picking $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and setting $\sigma = (\sigma_1, \sigma_2)$ with $\sigma_1 = g_2^\alpha \cdot (u' \cdot \prod_{i=1}^n u_i^{m_i})^r$ and $\sigma_2 = g^r$.

$\mathsf{Verify}(\mathsf{m}, \sigma, \mathsf{PK})$: a purported signature $\sigma = (\sigma_1, \sigma_2)$ on $\mathsf{m} = m_1 \dots m_n$ is accepted if

$$e(\sigma_1, g) = W \cdot e\big(u' \cdot \prod_{i=1}^{n} u_i^{m_i}, \sigma_2\big).$$

3 An Efficient ID-THIR in the Standard Model

In this section, we present an identity-based trapdoor hard-to-invert relation based on the Waters signature. More precisely, it uses a 2-level hierarchical extension [22,29] of the latter independently described in [12,32] and which is intentionally made existentially (but not universally) forgeable here.

In a nutshell, sampling a random lock/proof pair for some relation R_{id} is done by generating a signature (d_1, d_2, d_3) on some artificial random "message" c in the name of the identity id. The sampling algorithm uses the technique of the simulator in the security proof of [36] to handle signing queries without knowing the private key. Generating a proof for any given lock c is easily achieved using the private key for the identity id.

$\mathsf{Gen}(k, n)$: this algorithm chooses bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^k$ and a generator $g \in \mathbb{G}$. It computes $g_1 = g^\alpha$ for a random $\alpha \xleftarrow{R} \mathbb{Z}_p^*$. Next, it chooses $g_2 \xleftarrow{R} \mathbb{G}$, computes $W = e(g_1, g_2)$ and picks a random vector $\overline{u} = (u', u_1, \dots, u_n) \xleftarrow{R} \mathbb{G}^{n+1}$ which allows defining a function $F : \{0,1\}^n \to \mathbb{G}$ as

$$F(id) = u' \cdot \prod_{j=1}^{n} u_j^{i_j}$$

where $id = i_1 \dots i_n$ and $i_j \in \{0,1\}$ for all j. For an identity $id \in I_{\mathcal{R}} = \{0,1\}^n$, the relation $\mathcal{R}_{id}$ is defined as the set of pairs $\big(c, (d_1, d_2, d_3)\big) \in \mathbb{G} \times \mathbb{G}^3$ such that

$$e(d_1, g) = W \cdot e(F(id), d_2) \cdot e(c, d_3) \tag{1}$$

The master trapdoor is $\mathsf{mtd}_{\mathcal{R}} = g_2^\alpha$ and the family of relations $\mathcal{R}$ is entirely described by

$$D_{\mathcal{R}} = \{n, \mathbb{G}, \mathbb{G}_T, g, g_1, g_2, \overline{u}, W, \mathcal{R}_{id}, I_{\mathcal{R}}\}.$$

$\mathsf{Sample}(D_{\mathcal{R}}, id)$: to generate a random lock/proof pair $\big(c, (d_1, d_2, d_3)\big) \in \mathbb{G} \times \mathbb{G}^3$, this algorithm conducts the following steps.

1. Choose $j_1, j_2 \xleftarrow{R} \mathbb{Z}_p^*$ and compute $c = g_2^{j_1} g^{j_2}$.
2. Pick $r, s \xleftarrow{R} \mathbb{Z}_p^*$ and compute $d_1 = c^s \cdot g_1^{-j_2/j_1} \cdot F(id)^r$.
3. Set $d_2 = g^r$ and $d_3 = g^s \cdot g_1^{-1/j_1}$.

If we define $\tilde{s} = s - \frac{\alpha}{j_1}$, we observe that

$$d_1 = g_2^\alpha \cdot F(id)^r \cdot c^{\tilde{s}}, \quad d_2 = g^r, \quad d_3 = g^{\tilde{s}}. \tag{2}$$

$\mathsf{Check}_{D_{\mathcal{R}},id}(c,d)$: parse d as (d_1,d_2,d_3). Return 1 if

$$e(d_1,g) = W \cdot e(F(id),d_2) \cdot e(c,d_3)$$

and 0 otherwise.

$\mathsf{Extract}_{\mathsf{mtd}_{\mathcal{R}}}(id)$: given $\mathsf{mdt}_{\mathcal{R}} = g_2^{\alpha}$, a trapdoor for $id \in \{0,1\}^n$ is extracted by randomly choosing $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and returning $\mathsf{td}_{id} = (t_1,t_2) = (g_2^{\alpha} \cdot F(id)^r, g^r)$.

$\mathsf{Invert}_{\mathsf{td}_{id}}(c)$: parse td_{id} as (t_1,t_2). Choose random $r', s \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and return

$$(d_1,d_2,d_3) = (t_1 \cdot F(id)^{r'} \cdot c^s, t_2 \cdot g^{r'}, g^s) = (g_2^{\alpha} \cdot F(id)^{r''} \cdot c^s, g^{r''}, g^s).$$

with $r'' = r + r'$.

We now analyze the four security properties of the above scheme.

Correctness. It is clear that lock/proof pairs $(c,\tilde{d})$ where $\tilde{d} = \mathsf{Invert}_{\mathsf{td}_{id}}(c)$ satisfy equation (1) since $e(t_1,g) = W \cdot e(F(id),t_2)$ for all trapdoors $\mathsf{td}_{id} = (t_1,t_2)$ produced by $\mathsf{Extract}$. From (2), it follows that equation (1) is also satisfied by all pairs (c,d) produced by $\mathsf{Sample}(D_{\mathcal{R}},id)$. Now, we check that elements $\big(c,(d_1,d_2,d_3)\big)$ generated by Sample are actually distributed according to (2). Indeed, since $c = g_2^{j_1} g^{j_2}$, we have

$$d_1 = c^s \cdot g_1^{-j_2/j_1} \cdot F(id)^r = c^{\tilde{s}} \cdot \big(g_2^{j_1} g^{j_2}\big)^{\alpha/j_1} \cdot g_1^{-j_2/j_1} \cdot F(id)^r = g_2^{\alpha} \cdot c^{\tilde{s}} \cdot F(id)^r$$
$$d_3 = g^s \cdot g_1^{-1/j_1} = g^{\tilde{s}}.$$

The sampling algorithm uses the strategy (borrowed from the Boneh-Boyen framework [7]) of the simulator answering signing queries in the proof of the Waters scheme [36].

Ambiguity. Sampled pairs $\big(c,(d_1,d_2,d_3)\big)$ clearly have exactly the same distribution as pairs $\big(c,(\tilde{d}_1,\tilde{d}_2,\tilde{d}_3)\big)$ when $(\tilde{d}_1,\tilde{d}_2,\tilde{d}_3) = \mathsf{Invert}_{\mathsf{td}_{id}}(c)$.

Soundness. It directly derives from the fact that any given $c \in \mathbb{G}$ can be "signed" using the trapdoor for the relation R_{id} (which is a private key for the identity id in [12,32]).

One-wayness. The next theorem shows that our ID-THIR is one-way if Waters signatures are existentially unforgeable under chosen-message attacks [24].

Theorem 1. *An attacker breaking the one-wayness property of our* ID-THIR *in the sense of definition 1 implies a chosen-message attacker with the same advantage and running in comparable time for Waters signatures.*

Proof. Let $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$ be an adversary with advantage ε against the one-wayness property. We construct a forger $\mathcal{F}$ using $\mathcal{A}$ to forge a signature using a challenger $\mathcal{CH}$ answering signing queries.

Algorithm $\mathcal{F}$ first obtains a public key $\mathsf{PK} = (n,\mathbb{G},\mathbb{G}_T,g,g_1,g_2,\overline{u},W)$ from

its challenger $\mathcal{CH}$ and sends $\mathcal{A}$ an input $D_{\mathcal{R}}$ consisting of PK, $I_{\mathcal{R}} = \{0,1\}^n$ and a description of R_{id} for $id \in I_{\mathcal{R}}$.

Whenever $\mathcal{A}_1$ asks O_{Extract} for the trapdoor of a relation R_{id} for some identity $id \in I_{\mathcal{R}}$, $\mathcal{F}$ asks its challenger $\mathcal{CH}$ for a signature of the message id and relays the answer to $\mathcal{A}_1$. After polynomially-many queries to O_{Extract}, $\mathcal{A}_1$ comes up with an identity $id^\star$ that was never queried to O_{Extract}. At this stage, $\mathcal{F}$ generates a uniformly distributed lock $c = g^\omega$ for a random $\omega \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. In particular c has the same distribution as locks generated by Sample. On input of c and the state information transmitted by $\mathcal{A}_1$, $\mathcal{A}_2$ issues new queries to O_{Extract} which all trigger a signing query from $\mathcal{F}$ to $\mathcal{CH}$. Eventually, $\mathcal{A}_2$ is expected to output a proof (d_1, d_2, d_3) such that

$$e(d_1, g) = W \cdot e(F(id^\star), d_2) \cdot e(g^\omega, d_3)$$

which can be re-written as

$$e(d_1 \cdot d_3^{-\omega}, g) = W \cdot e(F(id^\star), d_2).$$

Hence, the pair $(\sigma_1 = d_1 \cdot d_3^{-\omega}, \sigma_2 = d_2)$ passes the verification test of Waters signatures. It is thus a valid forgery since $id^\star$ was not queried for signature by $\mathcal{F}$ as it may not have been queried to O_{Extract} by $\mathcal{A}_1$ or $\mathcal{A}_2$ at any time. □

Together with security results of [36], theorem 1 implies the following corollary.

Corollary 1. *Assuming that an adversary $\mathcal{A}$ breaks the one-wayness of our* ID-THIR *with advantage ε when running in time τ and making q_{td} trapdoor queries, there is an algorithm $\mathcal{B}$ that (τ', ε')-breaks the CDH assumption where*

$$\varepsilon' \geq \frac{\varepsilon}{4q_{\mathsf{td}}(n+1)} \qquad \tau' \leq t + O(q_{\mathsf{td}}\tau_{exp}),$$

τ_{exp} denoting the time complexity of an exponentiation in $\mathbb{G}$.

4 Shorter Public Keys for Small Identity Spaces

The ID-THIR construction of section 3 assumes a space of identities $I_{\mathcal{R}} = \{0,1\}^n$ where n can be as large as 160. In some applications, this space is quite likely to be much smaller. With time capsule signatures for instance, it is reasonable to settle for initializing the scheme in expectation of 2^{30} time periods.

In this case, the function $F : \{0,1\}^n \to \mathbb{G}$ can be replaced with Boneh and Boyen's selective-ID secure "hash" $F(id) = g_2^{H(id)} h$ [7] where $h \in_R \mathbb{G}$ and $H : \{0,1\} \to \mathbb{Z}_p^*$ is a collision-resistant hash function. This modification results in much shorter public parameters as a single group element $h \in \mathbb{G}$ supersedes the vector $\overline{u}$. The resulting ID-THIR remains one-way under the Diffie-Hellman assumption but the proof of one-wayness requires the Diffie-Hellman solver to guess which identity $id^\star$ will be attacked by $\mathcal{A}$ beforehand.

Theorem 2. *If an adversary $\mathcal{A}$ breaks the one-wayness of the modified* ID-THIR *with probability ϵ in time τ, the CDH problem can be (τ', ϵ')-solved where $\tau' \approx \tau$ and $\epsilon' = \epsilon/|I_{\mathcal{R}}|$.*

Proof. We outline an algorithm $\mathcal{B}$ solving a CDH instance (g^a, g^b) using $\mathcal{A}$ as a subroutine. To do so, $\mathcal{B}$ first picks $\rho \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and chooses $id^\star \stackrel{R}{\leftarrow} I_{\mathcal{R}}$ as a guess for the identity to be attacked by $\mathcal{A}$. Public parameters are defined as $g_1 = g^a$, $g_2 = g^b$ and $h = g_2^{-I^\star} g^\rho$, where $I^\star = H(id^\star) \in \mathbb{Z}_p^*$, so that $F(id) = g_2^{H(id)-I^\star} g^\rho$.

Trapdoor queries for identities $id \neq id^\star \in I_{\mathcal{R}}$ can be answered by choosing $s \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and returning

$$(t_1, t_2) = (F(id)^s \cdot g_1^{-\rho/(I-I^\star)}, g^s \cdot g_1^{-1/(I-I^\star)})$$

with $I = H(id) \in \mathbb{Z}_p^*$. The pair (t_1, t_2) has the correct distribution since

$$(t_1, t_2) = (g_2^a \cdot F(id)^{\tilde{s}}, g^{\tilde{s}})$$

with $\tilde{s} = s - a/(I - I^\star)$.

When $\mathcal{A}$ issues her challenge query, $\mathcal{B}$ fails if the target identity is not $id^\star$. Otherwise, it picks a random $\omega \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and responds with the challenge $c = g^\omega$. A successful attacker $\mathcal{A}$ is then expected to output a triple (d_1, d_2, d_3) satisfying

$$e(d_1, g) = W \cdot e(g^\rho, d_2) \cdot e(g^\omega, d_3)$$

which implies $e(d_1 \cdot d_2^{-\rho} \cdot d_3^{-\omega}, g) = e(g_1, g_2)$ and yields the solution $d_1 \cdot d_2^{-\rho} \cdot d_3^{-\omega}$ that $\mathcal{B}$ was after. □

Since $q_{td} < 2^{30}$ is a reasonable upper bound frequently encountered in the literature, the modified scheme should be preferred whenever $|I_{\mathcal{R}}| < 2^{30}$.

5 Efficient TCS Schemes in the Standard Model

The generic construction [17] of secure TCS from any ID-THIR is very simple and does not involve random oracles. It requires an ordinary digital signature scheme $\Sigma = (\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ and an ID-THIR $(\mathsf{Gen}, \mathsf{Sample}, \mathsf{Check}, \mathsf{Extract}, \mathsf{Invert})$. The signer generates a key pair $(\mathsf{PK}, \mathsf{SK}) \leftarrow \Sigma.\mathsf{Keygen}(k)$ while the Time Server runs $\mathsf{Gen}(k)$ to produce $(D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}})$ and sets $(\mathsf{TPK}, \mathsf{TSK}) = (D_{\mathcal{R}}, \mathsf{mtd}_{\mathcal{R}})$.

To produce a time capsule signature on a message m for time t, the signer samples a random lock/proof pair (c, d) for the relation R_t corresponding to the "identity" $t \in I_{\mathcal{R}}$. The future signature consists of c and the output σ of $\Sigma.\mathsf{Sign}_{\mathsf{SK}}(m||c||t)$ which can be verified by running $\Sigma.\mathsf{Verify}_{\mathsf{PK}}(m||c||t, \sigma)$. The signer also remembers d which is used for pre-hatching. The time release algorithm simply uses the master trapdoor $\mathsf{TSK} = \mathsf{mtd}_{\mathcal{R}}$ to generate a trapdoor $Z_t = \mathsf{td}_{R_t} = \mathsf{Extract}_{\mathsf{mtd}_{\mathcal{R}}}(t)$ for the "identity" t. Given a future signature $\langle c, \sigma \rangle$, the hatching algorithm uses $Z_t = \mathsf{td}_{R_t}$ to compute a proof $\tilde{d}$ for the lock c. Upon verification of a hatched or pre-hatched signature $\langle (c, d), \sigma \rangle$, the verifier accepts if $\Sigma.\mathsf{Verify}_{\mathsf{PK}}(m||c||t, \sigma)$ and $\mathsf{Check}_{D_{\mathcal{R}},t}(c, d)$ both return 1 and rejects otherwise.

5.1 A Concrete Scheme

The scheme described below is an example of concrete TCS in the standard model. It combines our ID-THIR scheme with Waters signatures. That is why all parties use common public parameters including the description of bilinear map groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^k$. In practice however, signers are free to choose their own parameters independently of the Time Server: they can use any secure digital signature in the standard model such as Cramer-Shoup [16].

$\mathsf{Setup}^{\mathsf{TS}}(k, n)$: the Time Server chooses a generator $g \in \mathbb{G}$. It computes $g_v = g^{\alpha_v}$ for a random $\alpha_v \xleftarrow{R} \mathbb{Z}_p^*$. Next, it chooses $g'_v \xleftarrow{R} \mathbb{G}$, computes $W_v = e(g_v, g'_v)$ and selects a random vector $\overline{v} = (v', v_1, \ldots, v_n) \xleftarrow{R} \mathbb{G}^{n+1}$ defining a function $F_v : \{0,1\}^n \to \mathbb{G} : t \to F_v(t) = v' \cdot \prod_{j=1}^{n} v_j^{t_j}$ where $t = t_1 \ldots t_n$ and $t_j \in \{0,1\}$ for all j. The Time Server's private key is $\mathsf{TSK} = {g'_v}^{\alpha_v}$ and the public key is

$$\mathsf{TPK} = \{n, \mathbb{G}, \mathbb{G}_T, g, g_v, g'_v, \overline{v}, W_v\}.$$

$\mathsf{Setup}^{\mathsf{User}}(k, n)$: the user picks $\alpha_u \xleftarrow{R} \mathbb{Z}_p^*$, $g'_u \xleftarrow{R} \mathbb{G}$ and a random $(n+1)$-vector $\overline{u} = (u', u_1, \ldots, u_n) \in \mathbb{G}^{n+1}$ which defines the function $F_u : \{0,1\}^n \to \mathbb{G}$ as $F_u(\mathsf{m}) = u' \cdot \prod_{j=1}^{n} u_j^{m_j}$ where $\mathsf{m} = m_1 \ldots m_n$ and $m_j \in \{0,1\}$ for all j. A collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^n$ is also chosen. The private key is $\mathsf{SK} = {g'_u}^{\alpha_u}$. The public key is $\mathsf{PK} = (n, g, g_u, g'_u, \overline{u}, W_u, H)$ with $g_u = g^{\alpha_u}$ and $W_u = e(g_u, g'_u)$.

$\mathsf{TSig}(m, t)$: the signer first generates a pair $\big(c, (d_1, d_2, d_3)\big) \in \mathbb{G} \times \mathbb{G}^3$ following these steps.

1. Choose $j_1, j_2 \xleftarrow{R} \mathbb{Z}_p^*$ and compute $c = {g'_v}^{j_1} g^{j_2}$.
2. Pick $r, s \xleftarrow{R} \mathbb{Z}_p^*$ and compute $d_1 = c^s \cdot g_v^{-j_2/j_1} \cdot F_v(t)^r$.
3. Set $d_2 = g^r$ and $d_3 = g^s \cdot g_v^{-1/j_1}$.

Then, he computes $\mathsf{m} = H(m||c||t) \in \{0,1\}^n$ and

$$\sigma = (\sigma_1, \sigma_2) = \big({g'_u}^{\alpha_u} \cdot F_u(\mathsf{m})^{r_u}, g^{r_u}\big)$$

for a randomly chosen $r_u \xleftarrow{R} \mathbb{Z}_p^*$. He outputs $\sigma'_t = \langle(\sigma_1, \sigma_2), c\rangle$ and stores the triple (d_1, d_2, d_3) for later use.

$\mathsf{TVer}(m, \sigma'_t, \mathsf{PK}, \mathsf{TPK}, t)$: parse σ'_t as $\langle(\sigma_1, \sigma_2), c\rangle$ and PK as $(n, g, g_u, g'_u, \overline{u}, W_u, H)$. Check that $c \in \mathbb{G}$ and return 0 otherwise. Return 1 if

$$e(\sigma_1, g) = W_u \cdot e(F_u(\mathsf{m}), \sigma_2)$$

with $\mathsf{m} = H(m||c||t) \in \{0,1\}^n$.

$\mathsf{TRelease}(t, \mathsf{TSK})$: given $\mathsf{TSK} = {g'_v}^{\alpha_v}$, the Time Server picks $r_v \xleftarrow{R} \mathbb{Z}_p^*$ and returns $Z_t = ({g'_v}^{\alpha_v} \cdot F_v(t)^{r_v}, g^{r_v})$.

$\mathsf{Hatch}(\sigma'_t, Z_t)$: parse σ'_t as $\langle(\sigma_1, \sigma_2), c\rangle$ and Z_t as $(z_1, z_2) = ({g'_v}^{\alpha_v} \cdot F_v(t)^{r_v}, g^{r_v})$. Pick r'_v, s and compute

$$(\tilde{d}_1, \tilde{d}_2, \tilde{d}_3) = (z_1 \cdot F_v(t)^{r'_v} \cdot c^s, z_2 \cdot g^{r'_v}, g^s) = ({g'_v}^{\alpha_v} \cdot F_v(t)^{r''_v} \cdot c^s, g^{r''_v}, g^s)$$

where $r''_v = r_v + r'_v$. The hatched signature is

$$\sigma_t = \langle(\sigma_1, \sigma_2), c, (\tilde{d}_1, \tilde{d}_2, \tilde{d}_3)\rangle$$

$\mathsf{PreHatch}(\sigma'_t, d)$: parse σ'_t as $\langle(\sigma_1, \sigma_2), c\rangle$ and d as (d_1, d_2, d_3), return the opened signature $\sigma_t = \langle(\sigma_1, \sigma_2), c, (d_1, d_2, d_3)\rangle$.

$\mathsf{Ver}(m, \sigma_t, \mathsf{PK}, \mathsf{TPK}, t)$: parse σ_t as $\langle(\sigma_1, \sigma_2), c, (d_1, d_2, d_3)\rangle$, the signer's public key PK as $(n, g, g_u, g'_u, \overline{u}, W_u, H)$ and TPK as $(n, g, g_v, g'_v, \overline{v}, W_v)$. Return 1 if

$$e(d_1, g) = W_v \cdot e(F_v(t), d_2) \cdot e(c, d_3) \tag{3}$$
$$e(\sigma_1, g) = W_u \cdot e(F_u(\mathsf{m}), \sigma_2) \tag{4}$$

where $\mathsf{m} = H(m||c||t) \in \{0,1\}^n$.

We note that the latter verification algorithm can be optimized as follows. Instead of sequentially verifying relations (3) and (4), the verifier can randomly choose $\beta_1, \beta_2 \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and accept the signature if

$$\frac{1}{W_v^{\beta_1} \cdot W_u^{\beta_2}} \cdot \frac{e(g, d_1^{\beta_1} \cdot \sigma_1^{\beta_2})}{e(F_v(t), d_2^{\beta_1}) \cdot e(c, d_3^{\beta_1}) \cdot e(F_u(\mathsf{m}), \sigma_2^{\beta_2})} = 1_{\mathbb{G}_T}.$$

Indeed, if we raise both members of (3) and (4) to the powers β_1 and β_2 respectively, we observe that the above verification test fails with overwhelming probability if either (3) or (4) does not hold. A product of four pairings (which is much faster to compute than a sequence of 4 independent pairings as discussed in [25]) suffices to check both conditions.

As explained in [17], the unconditional security against the signer follows from the correctness and soundness properties of the ID-THIR scheme. Theorem 2 in [17] shows that a successful cheating verifier obtaining a full signature without the help of the Time Server or the signer implies a successful inverter for the underlying ID-THIR scheme. The proof of this fact entails a degradation factor of q_{TSig} which is the number of queries to $\mathsf{O}_{\mathsf{TSig}}$.

Corollary 2. *If a cheating verifier $\mathcal{B}$ has advantage ε within running time τ when making q_{TR} queries to O_{TR} and q_{TSig} queries to O_{TSig}, there is an algorithm that (τ', ε')-breaks the CDH assumption where*

$$\varepsilon' \geq \frac{\varepsilon}{4q_{\mathsf{TR}}q_{\mathsf{TSig}}(n+1)} \qquad \tau' \leq t + O\big((q_{\mathsf{TR}} + q_{\mathsf{TSig}})\tau_{exp}\big),$$

where τ_{exp} is the time complexity of an exponentiation in $\mathbb{G}$.

It was also proved in [17] that a successful dishonest Time Server implies a chosen-message attacker breaking the underlying signature scheme with the same advantage. Together with results from [36], this yields the following corollary which completes the proof that a secure and efficient time capsule signature exists in the standard model under the Diffie-Hellman assumption.

Corollary 3. *If a cheating Time Server $\mathcal{C}$ has advantage ε within running time τ when making q_{TSig} queries to O_{TSig}, there is an algorithm that (τ', ε')-breaks the CDH assumption where*

$$\varepsilon' \geq \frac{\varepsilon}{4q_{\mathsf{TSig}}(n+1)} \qquad \tau' \leq t + O(q_{\mathsf{TSig}}\tau_{exp}),$$

where τ_{exp} is the same quantity as in corollary 2.

5.2 Efficiency Improvements for Smaller Number of Periods

In section 5.1, the Time Server performs the setup for a large number of time periods. As discussed in section 4, $N < 2^{30}$ is a smaller but quite realistic[1] number of time periods. In this case, the Server's public key can be shortened by replacing the Waters "hash" $F_v(t) = v' \prod_{j=1}^{n} v_j^{t_j}$ with $F_v(t) = g_2^{H(t)}h$, for a random element $h \in_R \mathbb{G}$ and a collision-resistant hash function $H : \{0,1\}^{\lceil \log_2 N \rceil} \to \mathbb{Z}_p^*$. The degradation factor of corollary 2 becomes $O(N \cdot q_{\mathsf{TSig}})$ instead of $O(q_{\mathsf{TR}} \cdot q_{\mathsf{TSig}})$.

We note that signers are free to implement the scheme with their favourite signing algorithm and they may prefer using short public keys. In this case, they can use the same common public parameters $(\mathbb{G}, \mathbb{G}_T)$ with other pairing-based signatures in the standard model. For instance, combining the selective-message secure signature of [7] at the Time Server with Strong Diffie-Hellman-based signatures [6,23] at the signer provides an efficient TCS scheme under the Strong Diffie-Hellman assumption. In this case, we have a tight reduction under a stronger assumption in corollary 3.

5.3 Reducing the Public Storage for the Time Server

A shortcoming of time capsule signatures considered in sections 5.1 and 5.2 is that Time Servers have to publish and store a number of group elements which is linear in the number of past time periods at any time. After n periods have passed, the server has to publish a bulletin board with $O(n)$ trapdoors.

To overcome this limitation also present in some time release primitives [31,5,15], Boneh et al. [8] proposed to use forward-secure primitives [1,3,14] backwards. Roughly said, forward-secure schemes protect the confidentiality or the authenticity of past communications by preventing past (but obviously not future) private keys to be computable from current ones. Hence, to encrypt a message for period t in the future, one can simply encipher it for period $N - t$

[1] For instance, a scheme could be used over more than 2000 years with 2^{30} periods of one minute.

using a forward-secure public key encryption scheme [14,8] prepared for N stages using the tree-like structure of [14]. Thanks to the latter, a private key for period $N-t$ allows anyone to derive keys for stages $N-t+1,\ldots,N$. In terms of time release primitives, the current private key allows recovering keys for past periods so that the public storage of the server never exceeds $O(\log^2 N)$ group elements.

It is not hard to see that aforementioned tricks apply to our context for a reasonably small number of time periods. At the server, we simply have to replace the selective-message secure signature of Boneh-Boyen [7] by the hierarchical selective-message secure signature suggested by the hierarchical IBE of [8]. It amounts to use the keying technique of a recently proposed forward-secure signature [11] in reverse. To generate a future signature for period t, the signer actually prepares it for period $N-t$. At period t, the Time Server only stores the trapdoor for period t (which is the "forward-secure private key" of period $N-t$) that allows deriving trapdoors for stages $1,\ldots,t-1$.

In this case, the security against cheating verifiers relies on a variant of the Diffie-Hellman problem which is to compute $g^{a^{\ell+1}}$ given $(g, g^a, \ldots, g^{a^\ell})$ where $\ell = \log_2 N$.

6 Conclusion

In this paper, we put forth the first practical construction of time capsule signature that provably fits the security definitions of [17] without using the random oracle heuristic. It stems from an efficient example of a recently introduced primitive which is of independent interest and in turn builds on Waters signatures and the Diffie-Hellman assumption.

We note that time capsule signatures with tight reductions remain elusive (even in the random oracle model). Solving this problem would require a new approach for constructing them since the generic construction of ID-THIRs entails a loss of $O(\mathsf{TSig})$ in the security bound against verifiers.

References

1. Anderson, R.: Two Remarks on Public Key Cryptology. In: Invited lecture, ACM Conference on Computer and Communications Security (1997)
2. Bellare, M., Goldwasser, S.: Encapsulated key-escrow. In: 4th ACM Conference on Computer and Communications Security, pp. 78–91. ACM Press, New York (1997)
3. Bellare, M., Miner, S.: A Forward-Secure Digital Signature Scheme. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
5. Blake, I., Chan, A.-C.-F.: Scalable, Server-Passive, User-Anonymous Timed Release Public Key Encryption from Bilinear Pairing. In: ICDCS'05, pp. 504–513. IEEE Computer Society, Los Alamitos (2005)

6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
7. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
8. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
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. Boneh, D., Naor, M.: Timed Commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
11. Boyen, X., Shacham, H., Shen, E., Waters, B.: Forward-Secure Signatures with Untrusted Update. In: ACM CCS'06, ACM Press, New York (2006)
12. 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)
13. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. Journal of the ACM 51(4), 557–594 (2004)
14. Canetti, R., Halevi, S., Katz, J.: A forward secure public key encryption scheme. In: Biham, E. (ed.) Advances in Cryptology – EUROCRPYT 2003. LNCS, vol. 2656, pp. 254–271. Springer, Heidelberg (2003)
15. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Timed-Release and Key-Insulated Public Key Encryption. In: Financial Cryptography 2006. LNCS, Springer, Heidelberg (to appear) Available from http://eprint.iacr.org/2004/231
16. Cramer, R., Shoup, V.: Signature schemes based on the strong rsa assumption. In: 7th ACM Conference on Computer and Communications Security, pp. 46–51. ACM Press, New York (1999)
17. Dodis, Y., Yum, D.-H.: Time Capsule Signature. In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 57–71. Springer, Heidelberg (2005)
18. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC'90, pp. 416–426. ACM Press, New York (1990)
19. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1986)
20. Garay, J., Jakobsson, M.: Timed-Release of Standard Digital Signatures. In: FC 2002. LNCS, vol. 2357, pp. 168–182. Springer, Heidelberg (2002)
21. Garay, J., Pomerance, C.: Timed Fair Exchange of Standard Signatures. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 190–207. Springer, Heidelberg (2003)
22. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
23. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
24. Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. 17(2), 281–308 (1988)
25. Granger, R., Smart, N.P.: On Computing Products of Pairings. Cryptology ePrint Archive: Report 2006/172 (2006)

26. Groth, J., Ostrovsky, R., Sahai, A.: Perfect Non-interactive Zero Knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
27. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive Zaps and New Techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
28. 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)
29. Kiltz, E., Mityagin, A., Panjwani, S., Raghavan, B.: Append-Only Signatures. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 434–445. Springer, Heidelberg (2005)
30. Mao, W.: Timed-Release Cryptography. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 342–357. Springer, Heidelberg (2001)
31. Mont, M.C., Harrison, K., Sadler, M.: The HP time vault service: Innovating the way confidential information is disclosed at the right time, in HP Lab. Report HPL-2002-243 (2002)
32. Paterson, K.G., Schuldt, J.C.N.: Efficient Identity-based Signatures Secure in the Standard Model. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 207–222. Springer, Heidelberg (2006)
33. Rivest, R., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. MIT LCS Tech. Report MIT/LCS/TR-684 (1996)
34. De Santis, A., Persiano, G.: Zero-knowledge proofs of knowledge without interaction. In: FOCS'92, pp. 427–436 (1992)
35. 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 (1984)
36. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys

Cécile Delerablée[1,3], Pascal Paillier[2], and David Pointcheval[3]

[1] France Telecom Division R&D
cecile.delerablee@orange-ftgroup.com
[2] Gemalto Security Labs
pascal.paillier@gemalto.com
[3] ENS-CNRS
david.pointcheval@ens.fr

Abstract. This paper puts forward new efficient constructions for public-key broadcast encryption that simultaneously enjoy the following properties: receivers are stateless; encryption is collusion-secure for arbitrarily large collusions of users and security is tight in the standard model; new users can join dynamically i.e. without modification of user decryption keys nor ciphertext size and little or no alteration of the encryption key. We also show how to permanently revoke any subgroup of users. Most importantly, our constructions achieve the optimal bound of $O(1)$-size either for ciphertexts or decryption keys, where the hidden constant relates to a couple of elements of a pairing-friendly group. Our broadcast-KEM trapdoor technique, which has independent interest, also provides a dynamic broadcast encryption system improving all previous efficiency measures (for both execution time and sizes) in the private-key setting.

1 Introduction

Broadcast Encryption. The concept of stateless broadcast encryption was introduced by Fiat and Naor in [5]. In this paradigm, a broadcaster encrypts messages and transmits these to a group of users $\mathcal{U}$ who are listening to a broadcast channel and use their private keys to decrypt transmissions. The broadcaster may exclude any subset of users $\mathcal{R} \subseteq \mathcal{U}$ from being able to decrypt the contents of the broadcast thanks to a one-time exclusion or revocation mechanism. The subset of revoked users $\mathcal{R}$ is chosen at encryption time and may change from one encryption to the next. A broadcast encryption scheme is said to be (t, n)-collusion secure if for any r-subset $\mathcal{R} \subseteq \mathcal{U}$ with $r \leq t$ and $|\mathcal{U}| = n$, users in $\mathcal{R}$ can by no means infer information about the broadcast message. It is said to be fully collusion secure when it is (n, n)-collusion secure. There are mainly two categories of broadcast encryption systems:

$r \ll n$: meaning that we broadcast to all but a small set of revoked users. A number of systems [14,9,8] have been suggested that achieve at best $O(r)$-size ciphertexts and $O(\log n)$ private key size. This paper specifically focuses

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 39–59, 2007.

on this case and puts forward constructions which improve these bounds in the public-key setting and even reach their information-theoretic values in the private-key setting.

$n - r \ll n$: we broadcast only to a few users in the group. The best known systems are the scheme of Boneh, Gentry and Waters [3] which achieves $O(\sqrt{n})$-size ciphertexts and private keys, and the trivial scheme where users hold independent keys pairs and the message is sequentially encrypted under the $n - r$ public keys of non-revoked users. This trivial scheme achieves $O(n - r)$-size ciphertexts and $O(1)$ private keys and is more efficient than [3] when $n - r < O(\sqrt{n})$.

Although previous works often use as an efficiency measure the size of public and private keys, we choose to rigorously separate encryption from decryption key material by taking into account all elements required to perform encryption or decryption. We will denote by λ_c the size of the broadcast ciphertext, λ_{dk} the (maximal) size of a user decryption key (which may then contain private and public parts) and λ_{ek} the size of the encryption key (which may be public or private). τ_{ek} and τ_{dk} denote the execution time of encryption and decryption respectively.

Related work in the $r \ll n$ case. Naor et al. [14] suggested two fully collusion secure broadcast systems: NNL_1 (based on the Complete-Subtree method) which achieves $\lambda_c = O(r \log n/r)$ and $\lambda_{\mathsf{dk}} = O(\log n)$ and NNL_2 (Subtree-Difference method) where $\lambda_c = O(r)$ and $\lambda_{\mathsf{dk}} = O(\log^2 n)$. Originally the (private) encryption key has size linear in n, but by using a PRF to generate user decryption keys, the size of the encryption key can be reduced to $O(1)$ in both NNL_1 and NNL_2. Note however that these two systems do not support public-key encryption. Dodis and Fazio [4] later refined NNL_2 into a public key broadcast encryption scheme with $O(1)$-size encryption key. We also mention the work of Dodis and Fazio [4] which by using parallelized schemes lead to a broadcast system that has essentially the same characteristics as those of NNL_2.

Related work in the $n - r \ll n$ case. The so-called trivial broadcast system consists in multiple encryptions of the message under individual and unrelated public keys. It is easily seen that this gives a public-key broadcast encryption scheme with $\lambda_c = O(n - r)$, $\lambda_{\mathsf{dk}} = O(1)$ and $\lambda_{\mathsf{ek}} = O(n)$. Recently, Boneh, Gentry and Waters [3] proposed a very efficient public-key broadcast encryption system (called BGW_1 hereafter) where both ciphertexts and private keys are of constant size while the public key has size $O(n)$. However, in order to decrypt ciphertexts, users need to store this public key in addition to their $O(1)$-size private keys. Thus each user has to store an actual decryption key of size $\lambda_{\mathsf{dk}} = O(n)$. The same authors suggested a second system (BGW_2) that achieves a trade-off between the key and ciphertext sizes and gives $\lambda_c = \lambda_{\mathsf{dk}} = \lambda_{\mathsf{ek}} = O(\sqrt{n})$. Overall, BGW_2 provides the best broadcast system known so far for general r's i.e. when $r \in [O(\sqrt{n}), n - O(\sqrt{n})]$.

All the above systems make use of the hybrid (KEM-DEM) encryption paradigm where the broadcast ciphertext only encrypts a symmetric key used to encrypt the broadcast contents. We mention that a number of other systems rely on an asymmetric encryption of the whole broadcast data. The encryption rate may asymptotically tend to one in these schemes, thereby reaching transmission sizes similar to hybrid encryption. However, asymmetrically encrypting the whole contents is often unrealistic in practice for performance reasons.

Dynamic broadcast encryption. A basic property very much desired in broadcast encryption (and other group-based protocols) is that the group should be dynamic in the sense that the group manager can invite new members to join or permanently revoke undesired members in a very efficient way. Although long-term revocation necessarily implies a modification of the keys, there is no such theoretical requirement when a new member joins the group. In this respect, we say that a broadcast system is *dynamic* when

i) the system setup as well as the ciphertext size are fully independent from the expected number of users or an upper bound thereof,

ii) a new user can join anytime without implying a modification of preexisting user decryption keys,

iii) the encryption key is *unchanged* in the private-key setting or *incrementally updated* in the public-key setting, meaning that this operation must be of complexity at most $O(1)$.

Hence, by definition, dynamic systems support arbitrarily many users. In [3] as well as in NNL_1 and NNL_2, either a large upper bound on the number of possible users is chosen at initialization time or the decryption keys have to be recomputed when a user joins the group, resulting in that those systems are not dynamic. Similarly, the trivial broadcast system is not dynamic since the ciphertext must include one additional element per new user, irremediably altering its size. As discussed in [14, p. 56], the property of being dynamic is incompatible with forward-secrecy because new group members can actually decrypt all previously encrypted messages. This feature may however be desirable; a newly manufactured DVD player is expected to play any properly encrypted DVD issued in the past. Achieving forward-secrecy requires the long-term revocation and a re-keying of user decryption keys.

Our contributions. Introducing a new multi-receiver encryption trapdoor based on bilinear maps, we suggest broadcast encryption systems which improve the points discussed above. First, in all our schemes, either the broadcast ciphertext or the decryption key dk_i containing all the information required by the receiver to decrypt is of constant size. Second, the group manager can dynamically include new members while preserving previously computed information: in particular, user decryption keys need not be recomputed, the morphology and size of ciphertexts are unchanged and the group encryption key ek requires minimal or no modification. Thirdly, our constructions provably resist full collusions

relative to a bilinear map related computational problem denoted (t, n)-GDDHE. Our security reductions are *tight* and do not rely on random oracles. We also show that (t, n)-GDDHE has generic security. Finally, we introduce the most efficient private-key broadcast encryption scheme known so far which features constant-size encryption and decryption keys and information-theoretically minimal ciphertext size.

For the sake of completeness, Figure 1 compares our schemes with the previous proposals in terms of ciphertext and key sizes. We consider the private-key broadcast encryption schemes NNL_1 and NNL_2 proposed by Naor et al. [14] (where the group encryption key ek remains private) as well as the public-key broadcast encryption schemes BGW_1 and BGW_2 proposed by Boneh et al. [3]. We denote by $\mathsf{BGW'}_1$ a slightly modified version of BGW_1 where the public parameters needed by the decryption procedure are included in the ciphertext rather than in the decryption key. BGW_1 and $\mathsf{BGW'}_1$ are described in more detail in Appendix A.

Schemes	Key/Ciphertext Sizes			Time Complexity		Dynamic?
Public-key	λ_{ek}	λ_{dk}	λ_c	τ_{ek}	τ_{dk}	
Trivial	$O(n)$	$O(1)$	$O(n-r)$	$O(n-r)$	$O(1)$	no
BGW_1	$O(n)$	$O(n)$	$O(1)$	$O(n-r)$	$O(n-r)$	no
$\mathsf{BGW'}_1$	$O(n)$	$O(1)$	$O(n-r)$	$O(n-r)$	$O(n-r)$	no
BGW_2	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	no
Construction 1	$O(n)$	$O(1)$	$O(r)$	$O(r^2)$	$O(r)$	yes
Construction 2	$O(n)$	$O(n)$	$O(1)$	$O(r^2)$	$O(r^2)$	no
Private-key	λ_{ek}	λ_{dk}	λ_c	τ_{ek}	τ_{dk}	
NNL_1	$O(1)$	$O(\log n)$	$O(r \log \frac{n}{r})$	$O(r \log \frac{n}{r})$	$O(\log \log n)$	no
NNL_2	$O(1)$	$O(\log^2 n)$	$O(r)$	$O(r)$	$O(\log n)$	no
Construction 3	$O(1)$	$O(1)$	$O(r)$	$O(r)$	$O(r)$	yes

Fig. 1. Comparing the efficiency of fully collusion secure broadcast encryption schemes

Remark 1. It is a common practice in broadcast systems (and other group-oriented protocols as well) to ignore the part of the broadcast ciphertext that identifies the target subset of users (our $\mathcal{R}$, or the subset S of effective receivers in [3]). What is called ciphertext size usually refers to the size of the header alone, not the size of the *full* header. As discussed in [14], transmitting the r-subset $\mathcal{R} \subseteq \{1, \ldots, n\}$ requires an extra $O(r \log n/r)$ bits. We note however that transmitting $\mathcal{R}$ is actually not necessary in Constructions 1 and 3.

Roadmap. Section 2 provides a number of definitional facts about bilinear maps and the General Diffie-Hellman Exponent assumption. We define dynamic broadcast encryption schemes and related security notions in Section 3. We describe

our main construction in Section 4 and prove its security in Section 5. We suggest a number of related variants and discuss these in Section 6. We finally conclude on a number of open issues.

2 Preliminaries

2.1 Bilinear Maps

We briefly review the necessary facts about bilinear maps. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p. The group laws in $\mathbb{G}_1$ and $\mathbb{G}_2$ are noted additively using elliptic curve conventions, whereas the inner law of $\mathbb{G}_T$ is noted multiplicatively. A bilinear map $e(\cdot,\cdot)$ is a map $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that for any $G \in \mathbb{G}_1$, $H \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$,

$$e([a]\,G, [b]\,H) = e(G, H)^{ab}$$

and $e(G, H) \neq 1$ unless $G = 1$ or $H = 1$. A bilinear map group system $\mathbb{S}$ is a tuple

$$\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$$

composed of the objects as described above. $\mathbb{S}$ may also include group generators in its description. We impose all group operations as well as the bilinear map $e(\cdot,\cdot)$ to be efficiently computable i.e. in time $\mathsf{poly}(|p|)$. We know three categories of bilinear map group systems that are of interest in cryptography:

- the symmetric case $\mathbb{G}_1 = \mathbb{G}_2$ and by extension the one where an efficient and efficiently invertible isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is known [10,11,2],
- the asymmetric case where an efficient isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is known but no efficient algorithm is known to invert ψ or more generally to isomorphically map $\mathbb{G}_1$ onto $\mathbb{G}_2$ [12,13],
- the dissociate case where no efficient isomorphism $\mathbb{G}_2 \rightarrow \mathbb{G}_1$ or $\mathbb{G}_1 \rightarrow \mathbb{G}_2$ is known.

As seen later, we make use of an arbitrary bilinear map group system in our constructions. In particular, we do not need $\mathbb{G}_1$ and $\mathbb{G}_2$ to be distinct nor equal. Neither do we require the existence[1] of an efficient isomorphism going either way between $\mathbb{G}_1$ and $\mathbb{G}_2$. Practical implementations may therefore rely on any category of bilinear maps and select an $\mathbb{S}$ that optimizes the size of group elements or the performance of group operations.

2.2 The General Diffie-Hellman Exponent Assumption

We make use of a nice generalization of the Diffie-Hellman Exponent assumption due to Boneh, Boyen and Goh [1]. The framework suggested in [1] applies to symmetric and asymmetric bilinear map group systems but can easily be

[1] A one-way isomorphism is often necessary to prove the security of pairing-based systems.

extended to the dissociate case, adopting the notations of [1]. We give here a rough overview in the symmetric case. Let then $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ be a bilinear map group system such that $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. Let $G_0 \in \mathbb{G}$ be a generator of $\mathbb{G}$, and set $g = e(G_0, G_0) \in \mathbb{G}_T$. Let s, m be positive integers and $P, Q \in \mathbb{F}_p[X_1, \ldots, X_m]^s$ be two s-tuples of m-variate polynomials over $\mathbb{F}_p$. We write $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$ and impose that $p_1 = q_1 = 1$. For any function $h : \mathbb{F}_p \to \Omega$ and vector $(x_1, \ldots, x_m) \in \mathbb{F}_p^m$, $h(P(x_1, \ldots, x_m))$ stands for $(h(p_1(x_1, \ldots, x_m)), \ldots, h(p_s(x_1, \ldots, x_m))) \in \Omega^s$. We use a similar notation for the s-tuple Q. Let $F \in \mathbb{F}_p[X_1, \ldots, X_m]$. It is said that F *depends* on (P, Q), which we denote by $F \in \langle P, Q \rangle$, when there exists a linear decomposition

$$F = \sum_{1 \le i,j \le s} a_{i,j} p_i p_j + \sum_{1 \le i \le s} b_i q_i \ ,$$

with coefficients $a_{i,j}, b_i \in \mathbb{Z}_p$. Let P, Q be as above and $F \in \mathbb{F}_p[X_1, \ldots, X_m]$. The (P, Q, F)-General Diffie-Hellman Exponent problems are defined as follows.

Definition 1 (**(P, Q, F)-GDHE**). *Given the vector*

$$H(x_1, \ldots, x_m) = \left([P(x_1, \ldots, x_m)]\, G_0, g^{Q(x_1, \ldots, x_m)} \right) \in \mathbb{G}^s \times \mathbb{G}_T^s \ ,$$

compute $g^{F(x_1, \ldots, x_m)}$.

Definition 2 (**(P, Q, F)-GDDHE**). *Given* $H(x_1, \ldots, x_m) \in \mathbb{G}^s \times \mathbb{G}_T^s$ *as above and* $T \in \mathbb{G}_T$, *decide whether* $T = g^{F(x_1, \ldots, x_m)}$.

We refer to [1] for a proof that (P, Q, F)-GDHE and (P, Q, F)-GDDHE have generic security when $F \notin \langle P, Q \rangle$. We prove that our constructions are fully collusion secure based on the assumption that (P, Q, F)-GDDHE is intractable for some well-defined P, Q, F with $F \notin \langle P, Q \rangle$ and for polynomial parameters $s, m = \mathsf{poly}(\lambda)$ where λ is a security parameter.

3 Dynamic Public-Key Broadcast Encryption

We give a formal definition of a dynamic broadcast encryption scheme and discuss security notions that are associated to the concept. We basically refine the definition of [3] by including a join procedure.

3.1 Definition

A dynamic broadcast encryption scheme involves two authorities: a group manager and a broadcaster. The group manager grants new members access to the group[2] by providing to each new member a public label lab_i and a decryption

[2] Note that given our definition of dynamic broadcast encryption, the group manager cannot revoke users permanently since keys cannot be changed. See Section 6 for more detail.

key dk_i. The generation of $(\mathsf{lab}_i, \mathsf{dk}_i)$ is performed using a secret manager key mk. The broadcaster encrypts messages and transmits these to the whole group of users through the broadcast channel. In a public-key broadcast encryption scheme, the broadcaster does not hold any private information and encryption is performed with the help of a public group encryption key ek containing, possibly among other things, all user labels. When the broadcaster encrypts a message, some group members can be revoked temporarily from decrypting the broadcast content thanks to a one-time revocation mechanism. Following the KEM-DEM methodology, broadcast encryption is viewed as the combination of a specific key encapsulation mechanism (a Broadcast-KEM) with a symmetric encryption (DEM) that shall remain implicit throughout the paper. More formally, a dynamic public-key broadcast encryption scheme $\mathcal{DBE}$ with security parameter λ is a tuple of probabilistic algorithms $\mathcal{DBE} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Encrypt}, \mathsf{Decrypt})$ described as follows:

$\mathsf{Setup}(\lambda)$. Takes as input the security parameter λ and outputs a manager key mk and an initial group encryption key ek. The group manager is given mk, and ek is made public.

$\mathsf{Join}(\mathsf{mk}, i)$. Takes as input the manager key mk and a user counter i. Join generates a user label lab_i and a user decryption key dk_i. The user label lab_i is added to the group encryption key $\mathsf{ek} := \mathsf{ek} \cup \{\mathsf{lab}_i\}$ and the user decryption key dk_i is sent to the i-th user securely.

We denote by n the total number of users (evolving over time) and by $\mathcal{U} = \{1, \ldots, n\}$ the set of all users.

$\mathsf{Encrypt}(\mathsf{ek}, \mathcal{R})$. Takes as input the group encryption key ek and a set of revoked users $\mathcal{R} \subseteq \mathcal{U}$ and outputs a random pair (hdr, K).
When a message $M \in \{0,1\}^*$ is to be broadcast to users in $\mathcal{U} \setminus \mathcal{R}$, the broadcaster generates $(\mathsf{hdr}, K) \leftarrow \mathsf{Encrypt}(\mathsf{ek}, \mathcal{R})$, computes the encryption C_M of M under the symmetric key K and broadcasts $(\mathsf{hdr}, \mathcal{R}, C_M)$. We will refer to hdr as the header or broadcast ciphertext, $(\mathsf{hdr}, \mathcal{R})$ as the full header, K as the message encryption key and C_M as the broadcast body.

$\mathsf{Decrypt}(\mathsf{dk}_i, \mathcal{R}, \mathsf{hdr})$. Takes as input a header hdr, a subset $\mathcal{R} \subseteq \mathcal{U}$ and a user decryption key dk_i. If $i \in \mathcal{U} \setminus \mathcal{R}$, the algorithm outputs the message encryption key K which is then used to decrypt the broadcast body C_M and recover M.

3.2 Security Notions for Dynamic Public-Key Broadcast Encryption

Semantic Security against Static Adversaries. The standard security notion for broadcast encryption schemes is semantic security against static colluders. Since we consider dynamic public-key broadcast encryption, we extend the security definition to one that is a bit more general than in [3]. More specifically, we allow the adversary to see the group encryption key before choosing the corrupted users:

1. The challenger first runs $\mathsf{Setup}(\lambda)$ to generate a manager key mk and an initial group encryption key ek. The adversary $\mathcal{A}$ is given ek.

2. $\mathcal{A}$ runs exactly n times the Join procedure, but before each invocation $\mathcal{A}$ specifies whether the corresponding new group member is honest or corrupted. Following the i-th call to Join, a user label lab_i is created and added to ek and therefore given to the adversary. If user i is corrupted, $\mathcal{A}$ receives in addition the decryption key dk_i. The user counter i is then incremented and so forth. Eventually, $\mathcal{A}$ ends up with the decryption keys of all corrupted users $\mathcal{C} \subseteq \mathcal{U}$, that is $\{\mathsf{dk}_i\}_{i\in\mathcal{C}}$. Let $t = |\mathcal{C}|$.
3. The challenger runs algorithm $\mathsf{Encrypt}$ with $\mathcal{R} = \mathcal{C}$ i.e. by revoking all corrupted users to randomly generate $(\mathsf{hdr}, K) \leftarrow \mathsf{Encrypt}(\mathsf{ek}, \mathcal{C})$. The challenger randomly selects $b \leftarrow \{0,1\}$, sets $K_b = K$ and sets K_{1-b} to a random value in the appropriate range. The tuple (hdr, K_0, K_1) is returned to $\mathcal{A}$.
4. $\mathcal{A}$ eventually outputs a guess $b' \in \{0,1\}$.

The adversary wins the above game when $b' = b$. Viewing t, n as attack parameters, we denote by $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n, \mathcal{A})$ the advantage of $\mathcal{A}$ in winning the game:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n, \mathcal{A}) = |2 \times \Pr[b' = b] - 1| = |\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]|$$

where the probability is taken over the random coins of $\mathcal{A}$, the challenger and all probabilistic algorithms run by the challenger.

Definition 3 ((t, n)-Collusion Resistance). *Let*

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n) = \max_{\mathcal{A}} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n, \mathcal{A})$$

where the maximum is taken over all probabilistic algorithms $\mathcal{A}$ running in time $\mathsf{poly}(\lambda)$. A dynamic public-key broadcast encryption scheme $\mathcal{DBE}$ is said to be semantically secure against (t, n)-colluders if $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n) = \mathsf{negl}(\lambda)$.

Definition 4 (Full Collusion Resistance). *Note that for any integers t_1, t_2 such that $0 \le t_1 \le t_2 \le n$ one has $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t_1, n) \le \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t_2, n)$. $\mathcal{DBE}$ is said to be semantically secure against full collusions if $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(n, n) = \mathsf{negl}(\lambda)$ for $n = \mathsf{poly}(\lambda)$.*

Chosen-ciphertext security. Given a chosen-plaintext secure Broadcast-KEM, one can realize CCA secure public-key broadcast encryption using generic security enhancers such as Fujisaki-Okamoto [7,6] or REACT [15]. Applying the Fujisaki-Okamoto transform guarantees CCA security in the random oracle model assuming the one-wayness of the Broadcast-KEM. REACT gives the same guarantee (and a performance gain at decryption time) assuming one-wayness under plaintext-checking attacks. We therefore do not consider CCA security in our constructions, given that it can be realized at negligible cost[3].

[3] In the RO model. However combining the Broadcast-KEM with ID-based encryption as in [3] may allow to avoid random oracles when such a combination is possible.

Beyond static adversaries. We comment that the above definition captures adversaries that are less static than in previous schemes because the adversary may choose $\mathcal{C}$ somewhat adaptively while seeing how the group encryption key ek evolves while new users join the system. Up to our knowledge, no public-key broadcast encryption scheme is known to resist fully adaptive adversaries (i.e. where the adversary determines $\mathcal{C}$ after seing all the public information) in the standard model[4]. As commented in [3], any static adversary that has success probability ε in the (t, n)-collusion security game leads to an adaptive adversary with success probability $\varepsilon \cdot 2^{-n}$. However, in practice this reduction is only meaningful for small values of n and building systems resisting fully adaptive colluders is still an open problem in the field.

4 Public-Key DBE with Constant-Size Decryption Keys

4.1 A New Multi-receiver Encryption Trapdoor

We describe a public-key encryption scheme with multiple receivers featuring $O(1)$-size ciphertexts, decryption keys and encryption key. This system, which we call Construction 0, does not support user revocation and therefore is not a broadcast encryption system. We describe Construction 0 to show the basic trapdoor mechanism which we generalize later to achieve broadcast encryption.

Let $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ be a bilinear map group system with two randomly selected generators $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$. Assume one publishes $\mathsf{ek} = (H, W, V)$ where $W = [\gamma] G$ for some $\gamma \leftarrow \mathbb{Z}_p$ and $V = e(G, H)$ and keeps $\mathsf{mk} = (G, \gamma)$ secret. Now given $\mathsf{mk} = (G, \gamma)$ and a user counter i, we generate a unique decryption key by randomly selecting a fresh $x_i \leftarrow \mathbb{Z}_p$ and defining $\mathsf{dk}_i = (x_i, A_i, B_i)$ where

$$A_i = \left[\frac{x_i}{\gamma + x_i}\right] G\,, \qquad B_i = \left[\frac{1}{\gamma + x_i}\right] H\,.$$

To generate a random encryption key K given $\mathsf{ek} = (H, W, V)$, the broadcaster randomly picks $k \leftarrow \mathbb{Z}_p^\star$, computes

$$C_1 = [k] W\,, \qquad C_2 = [k] H\,, \qquad K = V^k$$

and broadcasts $\mathsf{hdr} = (C_1, C_2)$. To recover K from hdr with dk_i, the i-th user computes

$$e(C_1, B_i) \cdot e(A_i, C_2) = e(G, H)^{\frac{k \cdot \gamma}{\gamma + x_i}} \cdot e(G, H)^{\frac{k \cdot x_i}{\gamma + x_i}} = V^k = K\,.$$

It turns out that this encryption scheme achieves chosen-plaintext security under the co-DDH assumption. Namely, given $H, [a] H \in \mathbb{G}_2$ and $v, t \in \mathbb{G}_T$, it must be hard to decide whether $t = v^a$. Construction 0 may have applications on its own in contexts where user revocation is not required.

[4] This is known to be achievable in the random oracle and generic group models.

4.2 Achieving User Revocation: A Full Construction

We now proceed to describe our main dynamic public-key broadcast encryption scheme which we refer to as Construction 1 throughout the paper. Our scheme allows decoders to store constant-size decryption keys i.e. $\lambda_{\mathsf{dk}} = O(1)$ and features $O(r)$ ciphertext size where r is the number of revoked users. This is of particular interest when r is small (i.e. $r < \sqrt{n}$).

$\mathsf{Setup}(\lambda)$. Given the security parameter λ, a bilinear map group system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ is constructed such that $|p| = \lambda$. Also, two generators $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$ are randomly selected as well as a secret value $\gamma \leftarrow \mathbb{Z}_p^\star$. The manager key is defined as $\mathsf{mk} = (\mathbb{S}, G, H, \gamma)$. The initial group encryption key is $\mathsf{ek} = (\mathbb{S}, H, W, V)$ where $W = [\gamma]\, G$ and $V = e(G, H)$.

$\mathsf{Join}(\mathsf{mk}, i)$. Given $\mathsf{mk} = (\mathbb{S}, G, H, \gamma)$ and the user counter i, Join randomly selects a fresh $x_i \leftarrow \mathbb{Z}_p^\star$ (thus $x_i \neq x_j$ for $j < i$) and sets $\mathsf{dk}_i = (\mathbb{S}, x_i, A_i, B_i)$ and $\mathsf{lab}_i = (x_i, V_i, B_i)$ where

$$A_i = \left[\frac{x_i}{\gamma + x_i}\right] G\,, \qquad B_i = \left[\frac{1}{\gamma + x_i}\right] H \qquad \text{and} \qquad V_i = V^{\frac{1}{\gamma + x_i}}\,.$$

dk_i is securely given to user i and lab_i is appended to the group encryption key ek.

$\mathsf{Encrypt}(\mathsf{ek}, \mathcal{R})$. Assume for notational simplicity that $\mathcal{R} = \{1, \ldots, r\}$. Given $\mathsf{ek} = (\mathbb{S}, H, W, V, (x_1, V_1, B_1), \ldots, (x_n, V_n, B_n))$, the broadcaster computes

$$\begin{aligned} P_1 &= \left[\frac{1}{\gamma + x_1}\right] H, \\ P_2 &= \left[\frac{1}{(\gamma + x_1)(\gamma + x_2)}\right] H, \\ &\vdots \\ P_r &= \left[\frac{1}{(\gamma + x_1)\ldots(\gamma + x_r)}\right] H\,. \end{aligned}$$

We describe below a quadratic time algorithm $\mathsf{Aggregate}$ which $\mathsf{Encrypt}$ may use to compute

$$P_r = \mathsf{Aggregate}\left(\mathbb{G}_2, \left(x_1, \left[\frac{1}{\gamma + x_1}\right] H\right), \ldots, \left(x_r, \left[\frac{1}{\gamma + x_r}\right] H\right)\right)\,.$$

By running the $\mathsf{Aggregate}$ algorithm on $(x_1, B_1), \ldots, (x_r, B_r)$ and storing the intermediate variables $P_j = P_{j-1,j}$, $\mathsf{Encrypt}$ also computes P_j for $j < r$. The $\mathsf{Aggregate}$ algorithm given on Fig. 2 precisely goes through the successive evaluations of $P_1, \ldots, P_r$ and these can be stored at no extra cost. The broadcaster then picks $k \leftarrow \mathbb{Z}_p^\star$ at random and sets

$$C_1 = [k]\, W\,, \qquad C_2 = \left[\frac{k}{(\gamma + x_1)\ldots(\gamma + x_r)}\right] H = [k]\, P_r\,.$$

The same Aggregate algorithm can also be applied to compute

$$K' = V^{\frac{1}{(\gamma+x_1)\cdots(\gamma+x_r)}} = \mathsf{Aggregate}\left(\mathbb{G}_T, \left(x_1, V^{\frac{1}{\gamma+x_1}}\right), \ldots, \left(x_r, V^{\frac{1}{\gamma+x_r}}\right)\right) ,$$

wherefrom $K = K'^k$ is obtained. These two computations are performed in time $O(r^2)$, see below. Encrypt then defines

$$\mathsf{hdr} = (C_1, C_2, (x_1, P_1), \ldots, (x_r, P_r))$$

and outputs (hdr, K). The ciphertext thus contains r elements of $\mathbb{Z}_p$, $r+1$ elements of $\mathbb{G}_2$ and one element of $\mathbb{G}_1$.

$\mathsf{Decrypt}(\mathsf{dk}_i, \mathcal{R}, \mathsf{hdr})$. In order to retrieve the message encryption key K encapsulated in the header hdr, the i-th user uses $\{(x_j, P_j)\}_{j=1}^r \subseteq \mathsf{hdr}$ and $\mathsf{dk}_i = (\mathbb{S}, x_i, A_i, B_i)$ to compute

$$K = e\left(C_1, B_{i,\mathcal{R}}\right) \cdot e\left(A_i, C_2\right)$$

where

$$B_{i,\mathcal{R}} = \left[\frac{1}{\prod_{j=1}^r (\gamma + x_j)}\right] B_i = \left[\frac{1}{(\gamma + x_i)\prod_{j=1}^r (\gamma + x_j)}\right] H .$$

Here $B_{i,\mathcal{R}}$ is computed is computed in time $O(r)$ (instead of $O(r^2)$) given $(x_i, B_i) \subseteq \mathsf{dk}_i$ and $\{(x_j, P_j)\}_{j=1}^r \subseteq \mathsf{hdr}$ by using the $\mathsf{Aggregate}'$ algorithm given later on in the paper. This requires $x_i \neq x_j$ i.e. $i \notin \mathcal{R}$, otherwise $\mathsf{Aggregate}'$ faces a division by zero and returns an error.

Finally note that when $\mathcal{R} = \emptyset$, Construction 1 boils down to Construction 0 except for the encryption key $\mathsf{ek} = (\mathbb{S}, H, W, V) \cup \{(x_i, V_i, B_i)\}_{i=1}^n$ which, in Construction 0, does not include user labels.

4.3 Aggregation of 1-Degree Terms: Aggregate

The key encapsulation mechanism of Section 4.2 requires the computation of

$$P_r = \left[\frac{1}{(\gamma + x_1)\ldots(\gamma + x_r)}\right] H \in \mathbb{G}_2 \quad \text{and} \quad K' = e\left(G, H\right)^{\frac{1}{(\gamma+x_1)\cdots(\gamma+x_r)}} \in \mathbb{G}_T$$

given $\{\mathsf{lab}_j = (x_j, B_j, V_j)\}_{j=1}^r$ where the x_j's are pairwise distinct. We proceed to describe how $\mathsf{Aggregate}(\mathbb{G}_2, \cdots)$ allows to compute P_r from the x_j's and the B_j's. The same algorithm applies over $\mathbb{G}_T$ as well to compute K' from the x_j's and the V_j's.

Description. Given $x_1, \ldots, x_r$ and $B_j = \left[\frac{1}{\gamma + x_j}\right] H$ for $1 \leq j \leq r$, let us define for any (j, ℓ) such that $1 \leq j < \ell \leq r$,

$$P_{j,\ell} = \left[\frac{1}{\prod_{\kappa=1}^j (\gamma + x_\kappa)}\right] B_\ell = \left[\frac{1}{(\gamma + x_\ell)} \cdot \frac{1}{\prod_{\kappa=1}^j (\gamma + x_\kappa)}\right] H .$$

The Aggregate algorithm consists in computing sequentially $P_{j,\ell}$ for $j = 1, \ldots, r-1$ and $\ell = j+1, \ldots, r$ using the induction

$$P_{j,\ell} = \left[\frac{1}{x_\ell - x_j}\right](P_{j-1,j} - P_{j-1,\ell}) \tag{1}$$

and posing $P_{0,\ell} = B_\ell$ for $\ell = 1, \ldots, r$. The algorithm finally outputs $P_r = P_{r-1,r}$. Aggregate is displayed in more detail on Fig. 2. It is easily shown that Equ. 1 is sound. Note however that computing $P_{j,\ell}$ by Equ. 1 assumes $x_j \neq x_\ell$. If $x_j = x_\ell$ for some $j < \ell$ then $P_{j-1,j} = P_{j-1,\ell}$ and $P_{j,\ell}$, which then contains a $(\gamma + x_j)^2$ factor, cannot be computed (this is computationally infeasible without γ). In this case, we force Aggregate to abort and return an error symbol.

Complexity. One sees that the computation of a term of the form

$$\left[\frac{1}{\prod_{j\in R}(\gamma + x_j)}\right] H$$

for any subset $R \subseteq \{1, \ldots, n\}$ from the 1-degree terms $[1/(\gamma + x_j)]\, H$ where $j \in R$ is quadratic in the cardinality of R. More precisely, generalizing to $\mathbb{G} \in \{\mathbb{G}_2, \mathbb{G}_T\}$:

$$\mathsf{Time}\left[\mathsf{Aggregate}(\mathbb{G}, r \text{ terms})\right] \simeq \frac{r(r-1)}{2} \cdot (\tau_p + \tau_{\mathbb{G}}) \ ,$$

where τ_p is the execution time of a subtraction and an inversion modulo $p = |\mathbb{G}|$ and $\tau_{\mathbb{G}}$ the total time of a division and an exponentiation in $\mathbb{G}$.

4.4 Aggregation of Terms of Increasing Degree: Aggregate′

Description. The Aggregate algorithm can be accelerated if one is given $P_j = P_{j-1,j}$ for $j = 1, \ldots, r$ instead of B_j. It is easily seen that Equ. 1 provides a way to compute $P_1, \ldots, P_r$ from $B_1, \ldots, B_r$ (and vice versa) in quadratic time. So knowing either vector is quadratic-time equivalent. However, given $P_1, \ldots, P_r$ and $(x_i, B_i = \left[\frac{1}{\gamma + x_i}\right] H)$, Equ. 1 is reformulated as

$$\left[\frac{1}{(\gamma + x_i)\prod_{\kappa=1}^{j}(\gamma + x_\kappa)}\right] H = \left[\frac{1}{x_i - x_j}\right]\left(P_j - \left[\frac{1}{(\gamma + x_i)\prod_{\kappa=1}^{j-1}(\gamma + x_\kappa)}\right] H\right)$$

for any $j = 1, \ldots, r$. Note that the left-hand term gives $B_{i,\mathcal{R}}$ when $j = r$. This leads us to the iterative computation of $B_{i,\mathcal{R}}$ as depicted on Fig. 2.

Complexity. It is easily seen that $\mathsf{Time}\left[\mathsf{Aggregate}'(\mathbb{G}, r \text{ terms})\right] \simeq r \cdot (\tau_p + \tau_{\mathbb{G}})$.

5 Security Analysis

5.1 Security Reduction to (t, n)-GDHE

We prove the semantic security of our system by reformulating the security game in terms of sequences of polynomials and relying on the GDHE/GDDHE

Aggregate	Aggregate′
Input: two r-arrays $x = [x_1, \ldots, x_r]$ and $P = [B_1, \ldots, B_r]$ Output: P_r as defined above or $\perp$	Input: x_i, B_i, $x = [x_1, \ldots, x_r]$ and $P = [P_1, \ldots, P_r]$ Output: $B_{i,\mathcal{R}}$ as defined above or $\perp$
1. for $j = 1$ to $r - 1$ (a) for $\ell = j + 1$ to r i. if $x[j] = x[\ell]$ output $\perp$ ii. $P[\ell] = \left[\frac{1}{x[\ell]-x[j]}\right](P[j] - P[\ell])$ 2. output $P[r]$	1. set $temp = B_i$ 2. for $j = 1$ to r (a) if $x_i = x[j]$ output $\perp$ (b) set $temp = \left[\frac{1}{x_i - x[j]}\right](P[j] - temp)$ 3. output $temp$

Fig. 2. The Aggregate and Aggregate′ algorithms

framework of [1]. We start by defining the following intermediate computational problem.

Definition 5 (**(t, n)-GDHE**). *Let $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$ be a bilinear map group system and let f and g be the two random univariate polynomials*

$$f(X) = \prod_{i=1}^{t}(X + x_i) = \sum_{i=0}^{t} \mu_i X^i \,, \qquad g(X) = \prod_{i=t+1}^{n}(X + x_i) = \sum_{i=0}^{n-t} \nu_i X^i \,,$$

where all the x_i's are random and pairwise distinct elements of $\mathbb{Z}_p^\star$. Let G_0 be a generator of $\mathbb{G}_1$ and H_0 a generator of $\mathbb{G}_2$. Solving the (t, n)-GDHE problem consists, given

$$\begin{array}{lll} G_0\,, [\gamma]\,G_0\,, \ldots, \left[\gamma^{t-1}\right] G_0\,, & [\gamma \cdot f(\gamma)]\,G_0\,, & [k \cdot \gamma \cdot f(\gamma)]\,G_0\,, \\ H_0\,, [\gamma]\,H_0\,, \ldots, [\gamma^n]\,H_0\,, & & [k \cdot g(\gamma)]\,H_0\,, \\ e\,(G_0, H_0)^{f^2(\gamma)\cdot g(\gamma)}\,, & & \end{array}$$

in computing $e\,(G_0, H_0)^{k \cdot f(\gamma) \cdot g(\gamma)}$.

As usual, we denote by $\mathsf{Succ}^{\mathsf{gdhe}}(t, n, \mathcal{A})$ the success probability of a randomized algorithm $\mathcal{A}$ solving (t, n)-GDHE. Similarly to the above, one defines the decisional version of (t, n)-GDHE, which we call (t, n)-GDDHE. In the (t, n)-GDDHE game, an additional input $T \in \mathbb{G}_T$ is provided and solving (t, n)-GDDHE consists in deciding whether $T = e\,(G_0, H_0)^{k \cdot f(\gamma) \cdot g(\gamma)}$. We then denote by $\mathsf{Adv}^{\mathsf{gddhe}}(t, n, \mathcal{A})$ the advantage of an algorithm $\mathcal{A}$ in distinguishing the two distributions and set $\mathsf{Adv}^{\mathsf{gddhe}}(t, n) = \max_{\mathcal{A}} \mathsf{Adv}^{\mathsf{gddhe}}(t, n, \mathcal{A})$ over $\mathsf{poly}(|p|)$-time $\mathcal{A}$'s. The following statement is a corollary of Theorem 2 that can be found in Section 5.2.

Corollary 1 (Generic security of (t, n)-GDDHE). *For any probabilistic algorithm $\mathcal{A}$ that totalizes at most q queries to the oracles performing group operations in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and evaluations of the bilinear map $e(\cdot,\cdot)$,*

$$\mathsf{Adv}^{\mathsf{gddhe}}(t, n, \mathcal{A}) \leq \frac{(q + 2(n + t + 4) + 2)^2 \cdot (t + n)}{2p} .$$

Let $\mathcal{DBE}$ denote our construction (Construction 1) as per Section 4. We state:

Theorem 1. *For any n, t such that $0 \leq t \leq n$, one has $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n) \leq 2 \cdot \mathsf{Adv}^{\mathsf{gddhe}}(t, n)$.*

Proof. The rest of this section is dedicated to proving Theorem 1. To establish the semantic security of $\mathcal{DBE}$ against static adversaries, we assume to be given an adversary $\mathcal{A}$ breaking $\mathcal{DBE}$ under a (t, n)-collusion and we build a reduction algorithm $\mathcal{B}$ that distinguishes the two distributions of the GDDHE decision problem.

Generation of system parameters and initial ek. The reduction algorithm $\mathcal{B}$ is given as input a group system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ and a (t, n)-GDDHE instance in $\mathbb{S}$. Let then $f(X) = \prod_{i=1}^{i=t}(X + x_i)$ and $g(X) = \prod_{i=1}^{i=n-t}(X + x'_i)$ be two random polynomials of respective degree t and $n - t$ with non-zero pairwise distinct roots. $\mathcal{B}$ is also given a generator G_0 (resp. H_0) of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) and

$$\begin{array}{lll} G_0, [\gamma] G_0, \ldots, \left[\gamma^{t-1}\right] G_0 & [\gamma \cdot f(\gamma)] G_0 & [k \cdot \gamma \cdot f(\gamma)] G_0 \\ H_0, [\gamma] H_0, \ldots, [\gamma^n] H_0 & & [k \cdot g(\gamma)] H_0 \\ e(G_0, H_0)^{f^2(\gamma) \cdot g(\gamma)} , & & \end{array}$$

as well as $T \in \mathbb{G}_T$ which is either equal to $e(G_0, H_0)^{k \cdot f(\gamma) \cdot g(\gamma)}$ or to some random element of $\mathbb{G}_T$. $\mathcal{B}$ formally sets $G = [f(\gamma)] G_0$ (i.e. without computing it) and computes

$$\begin{array}{ll} H = [f(\gamma) \cdot g(\gamma)] H_0 & (\text{from } H_0, [\gamma] H_0, \ldots, [\gamma^n] H_0) \\ W = [\gamma] G = [\gamma \cdot f(\gamma)] G_0 & (\text{given as input}) \\ V = e(G_0, H_0)^{f^2(\gamma) \cdot g(\gamma)} = e(G, H) & (\text{given as input}) \end{array}$$

$\mathcal{B}$ then defines the group encryption key as $\mathsf{ek} = (H, W, V)$. Note that $\mathcal{B}$ can by no means compute the value of G. $\mathcal{B}$ then runs $\mathcal{A}$ on the system parameters $(\mathbb{S}, H)$ and ek.

Generation of corrupted keys. The adversary $\mathcal{A}$ chooses a t-subset $\mathcal{C} \subseteq \{1, \ldots, n\}$ that indicates which users are corrupted: the users whose index lies in $\mathcal{C}$ are corrupted, whereas the users with index in $\bar{\mathcal{C}} = \{1, \ldots, n\} \setminus \mathcal{C}$ are honest. To generate the keys of the corrupted users, we define $f_i(X) = f(X)/(X + x_i)$ for $i \in [1, t]$, which are polynomials of degree $t - 1$, and then set

$$A_i = [x_i \cdot f_i(\gamma)] G_0 = \left[\frac{x_i}{\gamma + x_i}\right] G , \qquad B_i = [f_i(\gamma) \cdot g(\gamma)] H_0 = \left[\frac{1}{\gamma + x_i}\right] H$$

and

$$V_i = e\left(G_0, H_0\right)^{f_i(\gamma)\cdot f(\gamma)\cdot g(\gamma)} = e\left(\left[f(\gamma)\right] G_0, \left[f\gamma)\cdot g(\gamma)\right] H_0\right)^{\frac{1}{\gamma+x_i}} = e\left(G, H\right)^{\frac{1}{\gamma+x_i}} \ .$$

To generate the labels of honest users, we define $g_i(X) = g(X)/(X + x'_i)$ for $i \in [1, n-t]$ and

$$B'_i = \left[f(\gamma)\cdot g_i(\gamma)\right] H_0 = \left[\frac{1}{\gamma + x'_i}\right] H \ , \qquad V'_i = e\left(G_0, H_0\right)^{f^2(\gamma)\cdot g_i(\gamma)} \ .$$

Note that the total number of users is n. Since the polynomials $f_i g$ and $f g_i$ are of degree $n-1$, B_i and B'_i can be computed easily. V_i can also be computed given $\left[\gamma^j\right] G_0$ for $j \in [0, t-1]$ and $\left[\gamma^j\right] H_0$ for $j \in [0, n]$ since these allow to compute $e\left(G_0, H_0\right)^{\gamma^k}$ for $k \in [0, n+t-1]$ and since polynomials $f_i f g$ and $f^2 g_i$ are of degree $n+t-1$. The user labels and keys $((x_i, A_i, B_i), (x_i, V_i, B_i))$ for $i \in [1, t]$ as well as the user labels (x'_i, B'_i, V'_i) for $i \in [1, n-t]$ are then given to the adversary $\mathcal{A}$.

Challenge broadcast ciphertext. $\mathcal{B}$ now builds a header $\mathsf{hdr} = (C_1, C_2, \{x_i, B_i\}_{i=1}^t)$ decryptable by the honest users i.e. where the users $\mathcal{C} \subseteq \mathcal{U}$ are revoked. Given the GDDHE instance, $\mathcal{B}$ computes

$$\begin{aligned} C_1 &= \left[k\cdot\gamma\cdot f(\gamma)\right] G_0 = \left[k\right] W \ , \\ C_2 &= \left[k\cdot g(\gamma)\right] H_0 = \left[\frac{k}{f(\gamma)}\right] H = \left[\frac{k}{(\gamma+x_1)\ldots(\gamma+x_t)}\right] H \ . \end{aligned}$$

Doing so, $\mathcal{B}$ implicitly defines the message encryption key K as

$$K = e\left(G, H\right)^{\frac{k}{(\gamma+x_1)\ldots(\gamma+x_t)}} = e\left(G_0, H_0\right)^{\frac{k\cdot f^2(\gamma)\cdot g(\gamma)}{f(\gamma)}} = e\left(G_0, H_0\right)^{k\cdot f(\gamma)\cdot g(\gamma)} \ .$$

$\mathcal{B}$ now selects a random bit $b \leftarrow \{0,1\}$, sets $K_b = T$ and sets K_{1-b} to a random element of $\mathbb{G}_T$. $\mathcal{B}$ sends the tuple (hdr, K_0, K_1) to $\mathcal{A}$.

Final outcome. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. $\mathcal{B}$ then outputs real if $b' = b$ or random otherwise. One has

$$\begin{aligned} \mathsf{Adv}^{\mathsf{gddhe}}(t, n, \mathcal{B}) &= \Pr[b' = b|\mathsf{real}] - \Pr[b' = b|\mathsf{random}] \\ &= \frac{1}{2} \times \left(\Pr[b' = 1|b = 1 \wedge \mathsf{real}] - \Pr[b' = 1|b = 0 \wedge \mathsf{real}]\right) \\ &\quad -\frac{1}{2} \times \left(\Pr[b' = 1|b = 1 \wedge \mathsf{random}] + \Pr[b' = 1|b = 0 \wedge \mathsf{random}]\right) . \end{aligned}$$

Now in the random case, the distribution of b is independent from the adversary's view wherefrom

$$\Pr[b' = 1|b = 1 \wedge \mathsf{random}] = \Pr[b' = 1|b = 0 \wedge \mathsf{random}] \ .$$

In the real case however, the distributions of all variables defined by $\mathcal{B}$ perfectly comply with the semantic security game since all simulations are perfect. Therefore

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(\mathcal{A}) = \Pr[b' = 1 | b = 1 \wedge \mathsf{real}] - \Pr[b' = 1 | b = 0 \wedge \mathsf{real}] \ .$$

Summing up, we get that $\mathsf{Adv}^{\mathsf{gddhe}}(t, n, \mathcal{B}) = \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n, \mathcal{A})/2$. It is obvious that $\mathcal{B}$ runs in time similar to the one of $\mathcal{A}$. □

5.2 Proving the Intractability of (t, n)-GDHE

In this section, we prove the intractability of distinguishing the two distributions involved in the (t, n)-GDDHE problem cf. Corollary 1. We first review known results on the General Diffie-Hellman Exponent problem from [1]. Sticking to the most general security result, we assume a bilinear map system of the most favorable type for the adversary i.e. $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ or there is an efficient isomorphism going both ways between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Theorem 2 ([1]). *Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_m]$ be two s-tuples of m-variate polynomials over $\mathbb{F}_p$ and let $F \in \mathbb{F}_p[X_1, \ldots, X_m]$. Let d_P (resp. d_Q, d_F) denote the maximal degree of elements of P (resp. of Q, F) and pose $d = \max(2d_P, d_Q, d_F)$. If $F \notin \langle P, Q \rangle$ then for any generic-model adversary $\mathcal{A}$ totalizing at most q queries to the oracles (group operations in $\mathbb{G}, \mathbb{G}_T$ and evaluations of e) which is given $H(x_1, \ldots, x_m)$ as input and tries to distinguish $g^{F(x_1,\ldots,x_m)}$ from a random value in $\mathbb{G}_T$, one has*

$$\mathsf{Adv}(\mathcal{A}) \leq \frac{(q + 2s + 2)^2 \cdot d}{2p} \ .$$

Proof (of Corollary 1). In order to conclude with Corollary 1, we need to prove that the (t, n)-GDHE problem lies in the scope of Theorem 2. As already said, we consider the weakest case $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ and thus pose $H_0 = [\alpha] G_0$. In the (t, n)-GDHE problem, if one replaces γ by x, k by y and α by z, we see that our problem is reformulated as (P, Q, F)-GDHE where

$$P = \begin{pmatrix} 1, x, x^2, \ldots, x^{t-1}, & x \cdot f(x), & y \cdot x \cdot f(x) \\ z, z \cdot x, \ldots, z \cdot x^n, & y \cdot z \cdot g(x) & \end{pmatrix}$$
$$Q = (1, z \cdot f(x)^2 g(x))$$
$$F = y \cdot z \cdot f(x) g(x),$$

and thus $m = 3$ and $s = t + n + 4$. We have to show that F is independent of (P, Q) i.e. that no coefficients $\{a_{i,j}\}_{i,j=1}^{s}$ and $\{b_1, b_2\}$ exist such that $F = \sum_{i,j=1}^{s} a_{i,j} p_i p_j + \sum_{k=1}^{2} b_k q_k$ where the polynomials p_i and q_k are the one listed in P and Q above. By making all possible products of two polynomials from P which are multiples of $y \cdot z$, we want to prove that no linear combination among the polynomials from the list R below leads to F:

$$R = \begin{pmatrix} z \cdot y \cdot x^{n+1} \cdot f(x), \ldots, & z \cdot y \cdot x^2 \cdot f(x), & z \cdot y \cdot x \cdot f(x), \\ z \cdot y \cdot x^{t-1} \cdot g(x), \ldots, & z \cdot y \cdot x \cdot g(x), & z \cdot y \cdot g(x), \\ z \cdot y \cdot x \cdot f(x) g(x) & & \end{pmatrix} \ .$$

Note that the last polynomial can be written as $z \cdot y \cdot x \cdot f(x)g(x) = \sum_{i=0}^{i=n-t} \nu_i \cdot z \cdot y \cdot x^{i+1} \cdot f(x)$, and thus as a linear combination of the polynomials from the first line. Moreover, $z \cdot y \cdot x^{n+1} \cdot f(x)$ is the unique polynomial of degree $t+n+1$ in x and such a monomial does not appear in F. Similarly, $z \cdot y \cdot x^n \cdot f(x)$, ..., $z \cdot y \cdot x^{n-t+1} \cdot f(x)$ cannot appear in the combination. We therefore simplify the task to refuting a linear combination of elements of the list R' below which leads to $f(x)g(x)$:

$$R' = \begin{pmatrix} x^{n-t} \cdot f(x), \ldots, \ x^2 \cdot f(x), \ x \cdot f(x), \\ x^{t-1} \cdot g(x), \ldots, \ x \cdot g(x), \ g(x) \end{pmatrix} .$$

Any such linear combination can be written as

$$f(x)g(x) = A(x) \cdot f(x) + B(x) \cdot g(x)$$

where A and B are polynomials such that $A(0) = 0$, $\deg A = n - t$ and $\deg B = t - 1$. Since f and g are coprime by assumption, we must have $f \mid B$. Since $\deg f = t$ and $\deg B = t - 1$ this implies $B = 0$. Hence $A = g$ resulting in that $g(0) = \prod_{i=1}^{n-t} x'_i = 0$ which contradicts $x'_i \neq 0$ for $i \in [1, n-t]$. □

6 Related Constructions

From a design perspective, our main construction is fairly simple and can therefore be refined in many ways. We now present a few broadcast encryption systems derived from Construction 1. Although we do not provide proof details here, it can be shown that all constructions inherit full collusion resistance under the GDDHE assumption from Construction 1.

6.1 Public-Key Broadcast Encryption with Constant-Size Ciphertexts (Construction 2)

We suggest a simple variant of the previous construction that reaches optimally short ciphertexts i.e. $\lambda_c = O(1)$. The basic observation is that the fraction of the ciphertext containing $\{(x_j, P_j)\}_{j=1}^r$ can be computed from $\mathcal{R}$ and $\{(x_j, B_j)\}_{j=1}^r$ which is a subset of the public encryption key. Therefore P_j for $j \in \mathcal{R}$ can be removed completely from hdr if the complete set $\{(x_1, B_1), \ldots, (x_n, B_n)\}$ is made available at decryption time. Construction 2 therefore redefines

$$\mathsf{hdr} = (C_1, C_2) \qquad \text{where} \qquad C_1 = [k]\, W \ , \quad C_2 = [k]\, H \ ,$$

and

$$\mathsf{dk}_i = (\mathbb{S}, (x_i, A_i, B_i), (x_1, B_1), \ldots, (x_n, B_n)) \ .$$

As a result, the new system enjoys constant-size ciphertexts at the expanse of linear-size user decryption keys. Decryption can be performed as

$$K = e\,(C_1, B_{i,\mathcal{R}}) \cdot e\,(A_i, C_2)$$

where $B_{i,\mathcal{R}}$ is computed in time $O(r^2)$ using the $\mathsf{Aggregate}$ algorithm with the list $\{(x_j, B_j)\}_{j\in\mathcal{R}} \subseteq \mathsf{dk}_i$ and $(x_i, B_i) \subseteq \mathsf{dk}_i$ as inputs. Note here that the presence of $\mathcal{R}$ in the full broadcast ciphertext is mandatory to allow proper decryption[5].

We also note that the dynamic aspect is lost when doing these changes since all users have to update their decryption key whenever a new member joins the group. It is worthwhile noting that the characteristics of this construction are exactly those of BGW_1. Our scheme is slightly less efficient since decryption keys are bigger than those of in BGW_1 by the inclusion of n extra integers modulo p.

6.2 Private-Key DBE with Constant-Size Decryption and Encryption Keys (Construction 3)

In this scheme, the broadcaster and the group manager are the same entity. All algorithms are unchanged except for the join procedure which now exclusively employs the encryption key ek, mk being obsolete. The random selection of $x_i, i = 1, \ldots, n$ is now replaced by a pseudo-random generation. More precisely, we modify Construction 1 as follows:

1. $\mathsf{Setup}(\lambda)$ generates $\mathbb{S}$, randomly selects a hash function $\mathcal{H} : \{0,1\}^* \mapsto \mathbb{Z}_p^\star$ and outputs $\mathsf{ek} = (\mathbb{S}, G, H, \gamma, W, V, \mathcal{H})$,
2. $\mathsf{Join}(\mathsf{ek}, i)$ sets $x_i = \mathcal{H}(i)$, computes A_i and B_i as in Construction 1 and outputs the user decryption key $\mathsf{dk}_i = (\mathbb{S}, x_i, A_i, B_i)$,
3. $\mathsf{Encrypt}(\mathsf{ek}, \mathcal{R})$ picks a random $k \leftarrow \mathbb{Z}_p^\star$ and computes on the spot

$$\frac{1}{\gamma + x_1}, \quad \frac{1}{(\gamma + x_1)(\gamma + x_2)}, \quad \ldots, \quad \frac{1}{(\gamma + x_1)\ldots(\gamma + x_r)},$$

$$\frac{k}{(\gamma + x_1)\ldots(\gamma + x_r)}, \quad k\gamma$$

   over $\mathbb{Z}_p^\star$, thereby allowing to compute $P_1, \ldots, P_r, C_2, C_1, K$ directly by exponentiating H, V or W in their respective groups,
4. decryption is identical to the one of Construction 1.

The main difference with Construction 1 is that the broadcaster needs no linear-size group encryption key nor user labels: these are recovered whenever necessary using G, H, γ and the user counter i. Therefore $\lambda_{\mathsf{ek}} = O(1)$ and both encryption and decryption stages process in time $O(r)$. Note that all efficiency measures do not depend on the number of users n.

6.3 Realizing Long-Term Revocation

To revoke user i, the group manager broadcasts (x_i, B_i, V_i) and resets $H := B_i$ and $V := V_i$. A non revoked user $j \neq i$ can update his label as

$$B_j := \left[\frac{1}{\gamma + x_i}\right] B_j = \left[\frac{1}{x_j - x_i}\right](B_i - B_j) \qquad V_j := V_j^{\frac{1}{\gamma + x_i}} = \left(\frac{V_i}{V_j}\right)^{\frac{1}{x_j - x_i}}.$$

[5] Including $\mathcal{R}$ in the full header was unnecessary in Construction 1, see Remark 1.

Therefore the broadcaster does not have to repeat (x_i, B_i, V_i) in future encryptions. In virtue of the full collusion resistance of Construction 1, long-term revocations can be shown to be forward-secure.

7 Conclusion

We introduced new alternatives to design public and private key fully collusion secure broadcast encryption. Our designs support the inclusion of new users at minimal cost and achieve the best known security level relative to a generically secure computational problem via tight reductions in the standard model. We leave as an open problem to realize dynamic public-key broadcast encryption with an encryption key substantially shorter that $O(n)$. Resisting fully adaptive adversaries would also be a significant improvement. Finally, we expect our trapdoor mechanism to find other cryptographic applications in the future.

References

1. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005), Extended version available at http://eprint.iacr.org/2005/015
2. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
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. Dodis, Y., Fazio, N.: Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 100–115. Springer, Heidelberg (2002)
5. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
6. Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)
7. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
8. Goodrich, M.T., Sun, J.Z., Tamassia, R.: Efficient tree-based revocation in groups of low-state devices. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 511–527. Springer, Heidelberg (2004)
9. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 47–60. Springer, Heidelberg (2002)
10. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: ANTS 2000, pp. 385–394 (2000)
11. Joux, A., Nguyen, K.: Separating decision Diffie-Hellman from computational diffie-hellman in cryptographic groups. Journal of Cryptology 16(4), 239–247 (2003)

12. Miyaji, A., Nakabayashi, M., Takano, S.: Characterization of elliptic curve traces under fr-reduction. In: ICISC 2000, pp. 90–108 (2000)
13. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for fr-reduction. IEICE Transactions on Fundamentals, E84-A(5), 1234–1243 (2001)
14. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
15. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)

A $\mathsf{BGW_1}$ and $\mathsf{BGW'_1}$

We briefly describe the system BGW_1 put forward by Boneh et al. [3]. As BGW_1 is not dynamic, the Setup algorithm takes as input the number n of users in addition to the security parameter λ, and is in charge of computing the user decryption keys since there is no Join algorithm.

$\mathsf{Setup}(\lambda, n)$. Given the security parameter λ, a symmetric bilinear map group system

$$\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$$

is constructed such that $|p| = \lambda$. The algorithm first picks a generator $G \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $G_i = [\alpha^i] G \in \mathbb{G}$ for $i = 1, \ldots, n, n+2, \ldots, 2n$. Next, it picks a random $\gamma \leftarrow \mathbb{Z}_p$ and sets $W = [\gamma] G \in \mathbb{G}$. The encryption key is

$$\mathsf{ek} = (\mathbb{S}, G, G_1, \ldots, G_n, G_{n+2}, \ldots, G_{2n}, W) \in \mathbb{G}^{2n+1} .$$

The decryption key of user $i \in \{1, \ldots, n\}$ is set as $\mathsf{dk}_i = [\gamma] G_i \in \mathbb{G}$. The algorithm outputs the encryption key ek and the n decryption keys $\mathsf{dk}_1, \ldots, \mathsf{dk}_n$.

$\mathsf{Encrypt}(\mathsf{ek}, S)$. Given the encryption key ek and a subset $S \subseteq \{1, \ldots, n\}$ of users, the broadcaster randomly picks $k \leftarrow \mathbb{Z}_p^\star$, sets $K = e(G_{n+1}, G)^k = e(G_n, G_1)^k \in \mathbb{G}_T$ and computes

$$\mathsf{hdr} = \left([k] G,\ [k] \left(W + \sum_{j \in S} G_{n+1-j} \right) \right) \in \mathbb{G}^2$$

and outputs (hdr, K).

$\mathsf{Decrypt}(\mathsf{dk}_i, \mathsf{ek}, S, \mathsf{hdr})$. In order to retrieve the message encryption key K encapsulated in the header $\mathsf{hdr} = (C_1, C_2)$, the i-th user computes

$$K = e(G_i, C_2) \ / \ e(\mathsf{dk}_i + P_i, C_1) \quad \text{where} \quad P_i = \sum_{j \in S, j \neq i} G_{n+1-j+i} .$$

We define $\mathsf{BGW'}_1$ as a modification of BGW_1 where the decryption algorithm does not take ek as input parameter anymore but directly includes $\{P_i\}_{i\in S}$ in the header hdr instead. This change excepted, the encryption and decryption algorithms are unchanged. The motivation for considering $\mathsf{BGW'}_1$ is that the decryption key material at the user side does not include the encryption key ek, resulting in that $\lambda_{\mathsf{dk}} = O(1)$ instead of $\lambda_{\mathsf{dk}} = O(n)$ in BGW_1. This comes at the cost of a ciphertext size in $O(n-r)$ instead of $O(1)$ in BGW_1.

Certificateless Public Key Encryption in the Selective-ID Security Model (Without Random Oracles)*

Jong Hwan Park, Kyu Young Choi, Jung Yeon Hwang, and Dong Hoon Lee

Center for Information Security Technologies(CIST), Korea University, Seoul, Korea
{decartian,young,videmot,donghlee}@korea.ac.kr

Abstract. The concept of Certificateless Public Key Encryption (CL-PKE) eliminates the use of certificates in certified Public Key Encryption (PKE) scheme and the key-escrow problem in Identity Based Encryption (IBE) scheme. Al-Riyami and Paterson first proposed a CL-PKE scheme and proved its security in their security model (AP-model) using idealized random oracles. Several generic constructions were also proposed to construct a CL-PKE scheme by composing the standard PKE and IBE schemes. Recently, it was proved that some generic constructions are not secure against chosen ciphertext attacks in light of the security goals in the AP-model. In this paper, we show that all the known generic constructions are not secure against chosen ciphertext attacks, in the AP-model or a weaker security model than the AP-model. We also propose a CL-PKE scheme which is provably secure against chosen ciphertext attacks without random oracles. Our construction is proven secure in the selective-ID security model, reflecting the feature of CL-PKE scheme.

Keywords: Certificateless Public Key Encryption, Chosen Ciphertext Security, Bilinear Maps.

1 Introduction

Al-Riyami and Paterson introduced the concept of Certificateless Public Key Encryption (CL-PKE) and proposed a CL-PKE scheme which is secure in their security model (i.e., AP-model) [2]. The basic idea of CL-PKE is to construct a public/private key pair by combining a master key of Key Generation Center (KGC) with a secret value generated by a user. The combination of these two values eliminates the need of certificates and management overheads that traditional certified Public Key Encryption (PKE) scheme has and the key-escrow problem of a

* This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement)(IITA-2006-(C1090-0603-0025)).

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 60–82, 2007.

user's private key that may be inherent in Identity Based Encryption (IBE) scheme [10,8,25,20]. To achieve these purposes, another concept of Certificate-Based Encryption (CBE) was independently suggested by Gentry [19].

The AP-model considered two types of adversaries [2]: Type I and Type II. The Type I adversary represents a normal third party attacker and is allowed to change public keys of users at will (because of the lack of authenticating information for public keys). This adversary however does not have access to the master key. The Type II adversary which models a malicious KGC is equipped with the master key, but is not able to replace public keys.

In the AP-model, the Type I adversary is able to obtain the correct plaintext of the ciphertext encrypted with a replaced public key by making a decryption query. However, the CL-PKE scheme [14] and the generic constructions [26,27] were proved to be secure in a relaxed security model where such an ability of the Type I adversary is disabled. In the relaxed model, the Type I adversary is forbidden to make decryption queries on an identity for which the public key is replaced (or can make such queries, but decryption oracle uses the original private key. See the details in [16]). We refer to this type of adversary as Type I^- adversary.

All the CL-PKE schemes [2,3,14,4,24] except the generic constructions [1,26,27] and the scheme [23] in the literature were proved secure in the random oracle model. The idealized random oracle methodology may enable the design and security proof of cryptographic schemes more easy and efficient. But a secure scheme in the random oracle model may not be secure in the real world if an idealized random function is instantiated with a real function [11,21,5]. Recently, Liu et al. [23] presented a CL-PKE scheme that is secure in the standard model, based on the Waters' IBE scheme [25]. Their scheme has long public parameters and does not have a tight security reduction. Also, they adopted the relaxed security model in which the Type I^- adversary is justified. Therefore, it still remains a natural open problem whether it is feasible to construct a secure CL-PKE scheme without random oracles in the AP-model.

Our Contributions. It has been widely believed that one can obtain a CL-PKE scheme secure against chosen ciphertext attacks without random oracles by composing the standard PKE and IBE schemes. In this paper, we show that this belief is not correct. In fact, our security analysis shows that all the known generic constructions do not guarantee the chosen ciphertext security against the Type I and II adversaries. This security analysis implies that it is important how the master key of KGC and the secret value of the user are embedded into the public and private keys.

Next, we present two new CL-PKE schemes that are secure without random oracles. The security of the proposed schemes is proved in a weaker security model. That is, the proposed schemes are selective-ID secure. In this model, the Type I and II adversaries must commit ahead of time to the identity they intend to attack. However, the Type I adversary can still replace a public key for any identity including the selected one. Also, it can issue decryption queries for the

replaced public key on the selected identity, and decryption oracle responds to such queries with a correct plaintext. The latter is one of the important (but strong) features in the AP-model.

Our first scheme is provably selective-ID chosen plaintext secure, based on the Gentry's IBE scheme [20]. It is extended to achieve chosen ciphertext security by using the idea of signature-based method [13] (so called "CHK transformation"). As stated in [6], we obtain chosen ciphertext security by having the sender honestly generate a ciphertext using the chosen plaintext secure CL-PKE scheme. For this, our second scheme needs to append a one-time signature to the ciphertext. Our approach provides a way to achieve chosen ciphertext security from chosen plaintext security in a generic manner of the CHK transformation without random oracles. The proposed schemes have short public parameters and a tight security reduction. The security against the Type I and II adversaries is based on q-Bilinear Diffie-Hellman Inversion (q-BDHI) and 1-BDHI assumptions respectively.

Related Work. Since the first construction of CL-PKE in [2], many research has been done to create an efficient CL-PKE scheme [3,14,24] using bilinear pairings. Baek et al. [4] proposed a CL-PKE scheme without using a pairing. All of their schemes except [23] were proven secure against chosen ciphertext attacks in the random oracle model. Yum and Lee [26] provided a generic construction of CL-PKE from general cryptographic primitives such as PKE and IBE, and subsequently considered the relations between IBE, CBE, and CL-PKE [27]. In [7], the authors extended the concept of key encapsulation mechanism to IBE and CL-PKE, and built generic constructions of identity-based key encapsulation mechanism and certificateless public key encapsulation mechanism provably secure in the random oracle model. Recently, Libert and Quisquater [24] identified that some previous CL-PKE schemes and generic constructions are insecure against chosen ciphertext attacks in light of the security goals in the AP-model, and explained how to fix these problems by giving a so-called generic random oracle-using conversion. According to their method, any chosen plaintext secure CL-PKE scheme is converted into a chosen ciphertext secure CL-PKE scheme to be provably secure in the random oracle model. More recently, Liu et al. [23] proposed a CL-PKE scheme that is secure without random oracles, but as stated above they proved its security against the Type I^- adversary.

2 Preliminaries

2.1 Certificateless Public Key Encryption

The formal definition of CL-PKE was first presented in [2]. We follow their definition. A CL-PKE scheme consists of seven algorithms as follows:

Setup(k): takes a security parameter k and returns a public parameters *params* and a secret master key *master-key*.

Extract-Partial-Private-Key(*master-key*, ID): takes the master key *master-key* and a given identity ID as inputs. It returns a partial private key d_{ID}.

Set-Secret-Value(ID, ***params*****):** uses the public parameters *params* and a user identity ID to generate a secret value x_{ID}.

Set-Private-Key(d_{ID}, ***params***, x_{ID}**):** takes a partial private key d_{ID}, the public parameters *params*, and a secret value x_{ID} as inputs. It returns a (full) private key SK_{ID}.

Set-Public-Key(ID, x_{ID}, ***params*****):** uses its own identity ID, a secret value x_{ID}, and the public parameters *params* to generate a public key PK_{ID} corresponding to its own identity.

Encrypt(M, ***params***, PK_{ID}**):** takes a message M, the public parameters *params*, and a public key PK_{ID} for ID as inputs. It checks the validity of the public key. If invalid, it returns $\perp$. Otherwise, it returns a ciphertext CT.

Decrypt(CT, SK_{ID}, ***params***, **):** takes a ciphertext CT, a private key SK_{ID}, and the public parameters *params* as inputs. It returns a message M.

When one wishes to encrypt a message using the intended recipient's public key *without certificate*, he verifies that the public key is valid by checking certain conditions such as the check of equality [2] or element [3].

2.2 Selective-ID Security Model for CL-PKE

To prove the chosen ciphertext security of our proposed schemes without random oracles, we propose a weaker security model, called selective-ID CL-PKE. This is a modified version of the AP-model [2]. This notion is analogous to the that of selective-ID secure IBE, defined in [12,13,8]. In the selective-ID model for CL-PKE, the Type I and II adversaries first select an identity that they intend to attack respectively before the setup phase. The adversaries cannot make extraction or private key queries for the selected identity. However, our security model still captures the property of CL-PKE additionally. More concretely, the Type I adversary is allowed to make replacement queries of the public key for the selected identity even before the challenge phase, and the Type I adversary can issue decryption queries on the selected identity even though the public key for the selected identity has been replaced. In this case, the adversary is given the correct plaintext as in the AP-model.

We describe two selective-ID security games against the Type I and Type II adversaries. The first game between the challenger and the Type I adversary (denoted by $\mathcal{A}_I$) runs as follows:

Init: $\mathcal{A}_I$ outputs an identity ID^* where it wishes to be challenged.

Setup: The challenger runs *Setup* algorithm. It gives $\mathcal{A}_I$ the resulting system parameters *params*. It keeps the *master-key* to itself.

Phase 1: $\mathcal{A}_I$ issues queries $q_1, ..., q_m$ adaptively where query q_i is one of:

- Extraction query on $\mathrm{ID}_i \neq \mathrm{ID}^*$. The challenger responds by running *Extract-Partial-Private-Key* algorithm to generate the partial private key d_{ID_i} for ID_i.
- Private key query on $\mathrm{ID}_i \neq \mathrm{ID}^*$ where the public key for ID_i has not been replaced. The challenger responds by running *Set-Private-Key* algorithm to generate the private key SK_{ID_i}.

- Public key query on ID_i, including ID^*. The challenger responds by running *Set-Public-Key* algorithm to generate the public key PK_{ID_i}.
- Replacement query on the public key for ID_i, including ID^*. $\mathcal{A}_I$ can replace the public key of ID_i with a new value PK'_{ID_i} of its choice.
- Decryption query CT_i on ID^* (where the public key for ID^* may be replaced). The challenger responds by running *Decrypt* algorithm to decrypt the ciphertext CT_i using the private key SK_{ID^*}. Even though the public key for ID^* may be replaced, the challenger is forced to respond with a correct answer as in the AP-model.

Challenge: Once $\mathcal{A}_I$ decides that Phase 1 is over, it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ on which it wishes to be challenged. The challenger picks a random bit $b \in \{0,1\}$ and computes the challenge ciphertext $\mathsf{CT} = Encrypt(M_b, params, PK_{\mathrm{ID}^*})$, where the PK_{ID^*} may be replaced. It sends CT as the challenge to $\mathcal{A}_I$.

Phase 2: $\mathcal{A}_I$ issues more queries $q_{m+1}, ..., q_n$ adaptively where q_i is one of:

- Extraction query on $\mathrm{ID}_i \neq \mathrm{ID}^*$. The challenger responds as in Phase 1.
- Private key query on $\mathrm{ID}_i \neq \mathrm{ID}^*$, where the public key for ID_i has not been replaced. The challenger responds as in Phase 1.
- Public key query on ID_i, including ID^*. The challenger responds as in Phase 1.
- Replacement query on the public key for ID_i, including ID^*. The challenger responds as in Phase 1.
- Decryption query $\mathsf{CT}_i \neq \mathsf{CT}$ on ID^*. The challenger responds as in Phase 1.

Guess: Finally, $\mathcal{A}_I$ outputs a guess $b' \in \{0,1\}$. $\mathcal{A}_I$ wins if $b' = b$.

The advantage of $\mathcal{A}_I$ in breaking the CL-PKE scheme is defined as

$$\mathrm{Adv}(\mathcal{A}_I) = \left|\Pr[b = b'] - \frac{1}{2}\right|.$$

Definition 1. *A CL-PKE scheme is $(t, q_{EX}, q_{SK}, q_{PK}, q_R, q_D, \epsilon)$-selective-ID, adaptive chosen ciphertext (IND-sID-CCA) secure against the Type I adversary if for any t-time adversary $\mathcal{A}_I$ that makes at most q_{EX} chosen extraction queries, at most q_{SK} chosen private key queries, at most q_{PK} chosen public key queries, at most q_R chosen replacement queries, and at most q_D chosen decryption queries we have that $Adv(\mathcal{A}_I) < \epsilon$.*

Next, we describe the second game with the Type II adversary (denoted by $\mathcal{A}_{II}$). The second game is similar to the first one except two cases: in Setup, $\mathcal{A}_{II}$ is given *master-key* as well as *params* so that it does not need to make extraction queries. It also cannot replace any public keys in Phase 1 and 2. The rest of the security game is same as the one with $\mathcal{A}_I$.

Init: $\mathcal{A}_{II}$ outputs an identity ID^* where it wishes to be challenged.

Setup: The challenger runs *Setup* algorithm. It gives $\mathcal{A}_{II}$ the resulting system parameters *params* and the *master-key*.

Phase 1: $\mathcal{A}_{II}$ issues queries $q_1, ..., q_m$ adaptively where query q_i is one of:

- Private key query on $\mathrm{ID}_i \neq \mathrm{ID}^*$. The challenger responds by running *Set-Private-Key* algorithm to generate the private key SK_{ID_i}.
- Public key query on ID_i, including ID^*. The challenger responds by running *Set-Public-Key* algorithm to generate the public key PK_{ID_i}.
- Decryption query CT_i on ID^*. The challenger responds by running *Decrypt* algorithm to decrypt the ciphertext CT_i using the private key SK_{ID^*}.

Challenge: Once $\mathcal{A}_{II}$ decides that Phase 1 is over, it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ on which it wishes to be challenged. The challenger picks a random bit $b \in \{0, 1\}$ and computes the challenge ciphertext $\mathsf{CT} = Encrypt(M_b, params, PK_{\mathrm{ID}^*})$. It sends CT as the challenge to $\mathcal{A}_{II}$.

Phase 2: $\mathcal{A}_{II}$ issues more queries $q_{m+1}, ..., q_n$ adaptively where q_i is one of:

- Private key query on $\mathrm{ID}_i \neq \mathrm{ID}^*$. The challenger responds as in Phase 1.
- Public key query on ID_i, including ID^*. The challenger responds as in Phase 1.
- Decryption query $\mathsf{CT}_i \neq \mathsf{CT}$ on ID^*. The challenger responds as in Phase 1.

Guess: Finally, $\mathcal{A}_{II}$ outputs a guess $b' \in \{0, 1\}$. $\mathcal{A}_{II}$ wins if $b' = b$.

The advantage of $\mathcal{A}_{II}$ in breaking the CL-PKE scheme is defined as

$$\mathrm{Adv}(\mathcal{A}_{II}) = \left|\Pr[b = b'] - \frac{1}{2}\right|.$$

Definition 2. *A CL-PKE scheme is $(t, q_{SK}, q_{PK}, q_D, \epsilon)$-selective-ID, adaptive chosen ciphertext (IND-sID-CCA) secure against the Type II adversary if for any t-time adversary $\mathcal{A}_{II}$ that makes at most q_{SK} chosen private key queries, at most q_{PK} chosen public key queries, and at most q_D chosen decryption queries we have that $Adv(\mathcal{A}_{II}) < \epsilon$.*

The above two games can be extended to define *selective-ID, chosen plaintext* (IND-sID-CPA) secure CL-PKE scheme against the Type I and Type II adversaries as the same procedure, except that the adversaries are not permitted to make any decryption queries.

2.3 Bilinear Pairing and Complexity Assumption

Following the notations in [10,8], we first review the admissible bilinear pairing, denoted by e. Let $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of prime order p. Suppose g be a generator of $\mathbb{G}$. Then, e is a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$:

1) Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$ and 2) Non-degenerate: $e(g, g) \neq 1$. Note that $e(,)$ is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

The q-Bilinear Diffie-Hellman Inversion (BDHI) problem in $\mathbb{G}$ is defined as follows: given a tuple $(g, g^\alpha, \ldots, g^{\alpha^q}) \in \mathbb{G}^{q+1}$ for a random $\alpha \in \mathbb{Z}_p^*$ as input,

compute $e(g,g)^{1/\alpha} \in \mathbb{G}_1$. Informally, we can also say that the *decision* q-BDHI problem in $\mathbb{G}$ refers to the problem where given a tuple $(g, g^{\alpha}, \ldots, g^{\alpha^q}, T) \in \mathbb{G}^{q+1} \times \mathbb{G}_1$ for a random $\alpha \in \mathbb{Z}_p^*$, a polynomial-time attacker $\mathcal{A}$ is to decide whether $T = e(g,g)^{1/\alpha}$ or $T = e(g,g)^{\gamma}$ for random $\gamma \in \mathbb{Z}_p^*$. This assumption was already used to prove the security of Boneh et al.' IBE scheme [8] in the selective-ID model. We say that the *decision* (t, q, ϵ)-BDHI assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the *decision* q-BDHI problem in $\mathbb{G}$.

3 Insecurity of Known Generic CL-PKE Constructions Against Chosen Ciphertext Attacks

Up till now, there have been four generic constructions [1,26,27] for CL-PKE scheme. We call these generic constructions Generic-CL-PKE-1,2,3, and 4. According to their approaches, a CL-PKE scheme can be obtained by composing the standard PKE scheme with an IBE scheme in a sequential or parallel manner. The authors [1,26,27] demonstrated that CL-PKE schemes derived from their generic constructions are chosen ciphertext secure against the Type I (or the Type I$^-$[1]) and Type II adversaries in their security models. Since their constructions are generic, it seems that we can obtain a chosen ciphertext secure CL-PKE scheme secure without random oracles if we use the Cramer-Shoup PKE [15] and Waters-Kiltz IBE [22] or Gentry IBE schemes, which are proven secure against chosen ciphertext attacks without random oracles. But we observe that their generic constructions do not guarantee the chosen ciphertext security even if PKE and IBE schemes are secure against chosen ciphertext attacks.

Let $\prod_{\mathsf{IBE}}$=($\mathsf{ID}^{\mathsf{Gen}}$,$\mathsf{ID}^{\mathsf{Ext}}$,$\mathsf{ID}^{\mathsf{Enc}}$,$\mathsf{ID}^{\mathsf{Dec}}$) be an identity-based encryption scheme and $\prod_{\mathsf{PKE}}$=($\mathsf{PK}^{\mathsf{Gen}}$,$\mathsf{PK}^{\mathsf{Enc}}$,$\mathsf{PK}^{\mathsf{Dec}}$) be a public key encryption scheme (see the concrete definitions of IBE and PKE in Appendix A). We *briefly* describe the Generic-CL-PKE-1,2,3, and 4 in Table 1.

In their security models [1,26,27], all the Type II adversaries who model an eavesdropping KGC are given the master key in Setup phase. Since the Type II adversary with the master key can compute the partial private keys for itself, it does not have to make partial private key queries. It means that all the partial private keys in the Generic-CL-PKEs must be given to the Type II adversary in their security models. As the similar arguments to those in [17,24], we can show that the Generic-CL-PKE-1, 2, and 4 are not chosen ciphertext secure against, in particular, the Type II adversary. In the Generic-CL-PKE-1, for example, the Type II adversary can obtain a decryption $C' = \mathsf{ID}^{\mathsf{Dec}}_{d_{\mathrm{ID}_{ch}}}(\mathsf{CT})$ of challenge ciphertext CT using the partial private key $d_{\mathrm{ID}_{ch}}$ and a re-encryption $\mathsf{CT}' = \mathsf{ID}^{\mathsf{Enc}}_{params,\mathrm{ID}_{ch}}(C') \neq \mathsf{CT}$ of the same plaintext M_b. The ciphertext CT' can be submitted to the decryption oracle properly. We easily see that this strategy is similarly applied to the Generic-CL-PKE-2 and 4. This analysis means that

[1] In [26], the Type I$^-$ adversary is not allowed to ever extract the partial private key of the challenge identity ID_{ch}.

CL-PKE schemes from the Generic-CL-PKE-1, 2, and 4 are not chosen ciphertext secure, regardless of the Type I or Type I$^-$ adversary. The Generic-CL-PKE-3 resists the above attack, but a suitable attack by the Type I or Type I$^-$ adversary was already suggested by Libert et al. [24]. Consequently, all the known generic constructions for CL-PKE scheme do not guarantee the chosen ciphertext security against both the Type I (or the Type I$^-$) and Type II adversaries, simultaneously.

Table 1. Generic-CL-PKE -1, 2, 3, and 4

Generic-CL-PKE-1 [26]	Generic-CL-PKE-2 [27]
Setup: $(params,\ master\text{-}key) \leftarrow \mathsf{ID}^{\mathsf{Gen}}$ Extract($master\text{-}key$, ID): $d_{\mathrm{ID}} \leftarrow \mathsf{ID}^{\mathsf{Ext}}(master\text{-}key, \mathrm{ID})$ KeyGen(ID, $params$, d_{ID}): $(pk, sk) \leftarrow \mathsf{PK}^{\mathsf{Gen}}$ $PK_{\mathrm{ID}} \leftarrow (\mathrm{ID}, pk)$ $SK_{\mathrm{ID}} \leftarrow (d_{\mathrm{ID}}, sk)$ Encrypt($M, params, PK_{\mathrm{ID}}$): $\mathsf{CT} \leftarrow \mathsf{ID}^{\mathsf{Enc}}_{params,\mathrm{ID}}(\mathsf{PK}^{\mathsf{Enc}}_{pk}(M))$ Decrypt($\mathsf{CT}, SK_{\mathrm{ID}}$): $M \leftarrow \mathsf{PK}^{\mathsf{Dec}}_{sk}(\mathsf{ID}^{\mathsf{Dec}}_{d_{\mathrm{ID}}}(\mathsf{CT}))$	Setup: $(params,\ master\text{-}key) \leftarrow \mathsf{ID}^{\mathsf{Gen}}$ Extract($master\text{-}key$, ID): $d_{\mathrm{ID}} \leftarrow \mathsf{ID}^{\mathsf{Ext}}(master\text{-}key, \mathrm{ID})$ KeyGen(ID, $params$, d_{ID}): $(params', master\text{-}key') \leftarrow \mathsf{ID}^{\mathsf{Gen}}$ $d'_{\mathrm{ID}} \leftarrow \mathsf{ID}^{\mathsf{Ext}}(master\text{-}key', \mathrm{ID})$ $PK_{\mathrm{ID}} \leftarrow (\mathrm{ID}, params')$ $SK_{\mathrm{ID}} \leftarrow (d_{\mathrm{ID}}, d'_{\mathrm{ID}})$ Encrypt($M, params, PK_{\mathrm{ID}}$): $\mathsf{CT} \leftarrow \mathsf{ID}^{\mathsf{Enc}}_{params,\mathrm{ID}}(\mathsf{ID}^{\mathsf{Enc}}_{params',\mathrm{ID}}(M))$ Decrypt($\mathsf{CT}, SK_{\mathrm{ID}}$): $M \leftarrow \mathsf{ID}^{\mathsf{Dec}}_{d'_{\mathrm{ID}}}(\mathsf{ID}^{\mathsf{Dec}}_{d_{\mathrm{ID}}}(\mathsf{CT}))$
Generic-CL-PKE-3 [1]	**Generic-CL-PKE-4 [1]**
Setup: $(params,\ master\text{-}key) \leftarrow \mathsf{ID}^{\mathsf{Gen}}$ Extract($master\text{-}key$, ID): $d_{\mathrm{ID}} \leftarrow \mathsf{ID}^{\mathsf{Ext}}(master\text{-}key, \mathrm{ID})$ KeyGen(ID, $params$, d_{ID}): $(pk, sk) \leftarrow \mathsf{PK}^{\mathsf{Gen}}$ $PK_{\mathrm{ID}} \leftarrow (\mathrm{ID}, pk)$ $SK_{\mathrm{ID}} \leftarrow (d_{\mathrm{ID}}, sk)$ Encrypt($M, params, PK_{\mathrm{ID}}$): $\mathsf{CT} \leftarrow \mathsf{PK}^{\mathsf{Enc}}_{pk}(\mathsf{ID}^{\mathsf{Enc}}_{params,\mathrm{ID}}(M))$ Decrypt($\mathsf{CT}, SK_{\mathrm{ID}}$): $M \leftarrow \mathsf{ID}^{\mathsf{Dec}}_{d_{\mathrm{ID}}}(\mathsf{PK}^{\mathsf{Dec}}_{sk}(\mathsf{CT}))$	Setup: $(params,\ master\text{-}key) \leftarrow \mathsf{ID}^{\mathsf{Gen}}$ Extract($master\text{-}key$, ID): $d_{\mathrm{ID}} \leftarrow \mathsf{ID}^{\mathsf{Ext}}(master\text{-}key, \mathrm{ID})$ KeyGen(ID, $params$, d_{ID}): $(pk, sk) \leftarrow \mathsf{PK}^{\mathsf{Gen}}$ $PK_{\mathrm{ID}} \leftarrow (\mathrm{ID}, pk)$ $SK_{\mathrm{ID}} \leftarrow (d_{\mathrm{ID}}, sk)$ Encrypt($M, params, PK_{\mathrm{ID}}$): set $M_B = M_A \oplus M$ for random M_A $\mathsf{CT} \leftarrow (\mathsf{ID}^{\mathsf{Enc}}_{params,\mathrm{ID}}(M_A), \mathsf{PK}^{\mathsf{Enc}}_{pk}(M_B))$ Decrypt($\mathsf{CT}, SK_{\mathrm{ID}}$): let $\mathsf{CT} = (C_0, C_1)$ $M \leftarrow \mathsf{ID}^{\mathsf{Dec}}_{d_{\mathrm{ID}}}(C_0) \oplus \mathsf{PK}^{\mathsf{Dec}}_{sk}(C_1)$

However, as long as the Type I (or the Type I$^-$) and Type II adversaries are not allowed to make any decryption queries, we notice that the above Generic-CL-PKEs are secure against chosen plaintext attacks. This was already proved in [24] except the Generic-CL-PKE-2.[2] In [24], the authors presented a generic

[2] In the case of the Generic-CL-PKE-2 [27], the chosen plaintext security may be obtained although the authors did not state it explicitly.

way, called random oracle-using conversion, to construct a chosen ciphertext secure CL-PKE scheme from any chosen plaintext secure CL-PKE scheme. The resulting CL-PKE scheme is proven secure against chosen ciphertext attacks in the random oracle model.

4 Chosen Plaintext Secure CL-PKE

In this section we present a CL-PKE scheme that is IND-sID-CPA secure without random oracles under the q-BDHI and 1-BDHI assumptions. Though our construction is similar to the Gentry's IBE scheme [20], encryption algorithm requires two more pairing computations than Gentry's scheme in order to check the validity of a public key.

4.1 Construction

Let $\mathbb{G}$ be a bilinear group of prime order p.

Setup(k): To generate CL-PKE parameters, the KGC selects a random generator $g \in \mathbb{G}$ and random elements $h, u \in \mathbb{G}$. It randomly selects $\alpha \in \mathbb{Z}_p^*$ and defines $g_1 = g^\alpha \in \mathbb{G}$. The public parameters *params* (with the description of e, $\mathbb{G}$, $\mathbb{G}_1$, and p) and the secret *master-key* are given by

$$params = (g,\ g_1,\ h,\ u), \qquad master\text{-}key = \alpha.$$

Extract-Partial-Private-Key(*master-key*, ID): To generate a partial private key for a user $\mathrm{ID} \in \mathbb{Z}_p$, the KGC selects a random $r_{\mathrm{ID}} \in \mathbb{Z}_p$, and outputs the partial private key d_{ID}

$$d_{\mathrm{ID}} = (r_{\mathrm{ID}},\ h_{\mathrm{ID}}), \quad \text{where } \ h_{\mathrm{ID}} = (hg^{-r_{\mathrm{ID}}})^{1/(\alpha - \mathrm{ID})}.$$

If $\mathrm{ID} = \alpha$, the KGC aborts. As in [20], the KGC always uses the same random value r_{ID} for user ID.

Set-Secret-Value(ID, *params*): This algorithm picks a random $x_{\mathrm{ID}} \in \mathbb{Z}_p^*$, and outputs x_{ID} as a secret value.

Set-Private-Key(d_{ID}, *params*, x_{ID}): This algorithm outputs the (full) private key $SK_{\mathrm{ID}} = (x_{\mathrm{ID}},\ r_{\mathrm{ID}},\ h_{\mathrm{ID}})$ for user ID.

Set-Public-Key(ID, x_{ID}, *params*): This algorithm outputs the public key $PK_{\mathrm{ID}} = (X,\ Y)$ for user ID, where

$$X = (g_1 g^{-\mathrm{ID}})^{x_{\mathrm{ID}}}, \quad Y = u^{x_{\mathrm{ID}}}.$$

Encrypt(M, *params*, PK_{ID}): To encrypt a message $M \in \mathbb{G}_1$ under a public key PK_{ID}:

1. First check that the equality $e(X, u) = e(g_1 g^{-\mathrm{ID}}, Y)$ holds. If not, output $\perp$ and abort encryption.
2. Otherwise, pick randomly $s \in \mathbb{Z}_p^*$ and compute the ciphertext

$$\mathsf{CT} = (X^s,\ e(g,g)^s,\ e(g,h)^{-s} \cdot M).$$

Notice that two pairing values $e(g,g)$ and $e(g,h)$ can be precomputed so that the encryption algorithm needs two pairing computations to check the validity of PK_{ID}.

Decrypt(CT, SK_{ID}, *params*): To decrypt a ciphertext $\mathsf{CT} = (C_1, C_2, C_3)$ using the private key $SK_{\mathrm{ID}} = (x_{\mathrm{ID}}, r_{\mathrm{ID}}, h_{\mathrm{ID}})$, output

$$M = e(C_1^{1/x_{\mathrm{ID}}}, h_{\mathrm{ID}}) \cdot C_2^{r_{\mathrm{ID}}} \cdot C_3.$$

We can easily check that the decryption algorithm is correct as follows:

$$e(C_1^{\frac{1}{x_{\mathrm{ID}}}}, h_{\mathrm{ID}}) \cdot C_2^{r_{\mathrm{ID}}} = e(g^{s \cdot x_{\mathrm{ID}} \cdot (\alpha - \mathrm{ID}) \cdot \frac{1}{x_{\mathrm{ID}}}}, (hg^{-r_{\mathrm{ID}}})^{\frac{1}{\alpha - \mathrm{ID}}}) \cdot e(g,g)^{s \cdot r_{\mathrm{ID}}} = e(g,h)^s.$$

Note that the above scheme needs only one pairing computation in the decryption algorithm. Moreover, if the recipient's public key is not replaced, no pairing computation is required in a subsequent encryption algorithm.

4.2 Security

We prove security of the above scheme under the decision BDHI assumption described in Section 2.3. Full proofs are found in Appendix B.

Theorem 1. *Let $q = q_{EX} + q_{SK} + 1$. Suppose that the decision (t, q, ϵ')-BDHI assumption holds in $\mathbb{G}$. Then the previous CL-PKE scheme is $(t, q_{EX}, q_{SK}, q_{PK}, q_R, 0, \epsilon)$-selective-ID, adaptive chosen plaintext secure against the Type I adversary, where $\epsilon' \geq \epsilon$.*

Theorem 2. *Suppose that the decision $(t, 1, \epsilon')$-BDHI assumption holds in $\mathbb{G}$. Then the previous CL-PKE scheme is $(t, q_{SK}, q_{PK}, 0, \epsilon)$-selective-ID, adaptive chosen plaintext secure against the Type II adversary, where $\epsilon' \geq \epsilon$.*

5 Chosen Ciphertext Secure CL-PKE

We present a IND-sID-CCA secure CL-PKE scheme without random oracles by applying the idea of the CHK transformation [13] to the IND-sID-CPA secure CL-PKE scheme described in Section 4.1. In this construction, we need a one-time signature scheme $Sig = (SigKeyGen, Sign, Verify)$ which is strongly existentially unforgeable. Briefly speaking, a signature scheme is (t, q_S, ϵ)-strongly existentially unforgeable if no t-time forger who makes at most q_S signature queries is able to generate a new signature on even a previously signed message with probability at least ϵ (see the complete definition in [9]). Instead of signature-based method, we note that Message Authentication Code (MAC)-based method [6] can be applied. We will also need a collision resistant hash function that maps verification keys to $\mathbb{Z}_p$. For simplicity's sake, we assume that the verification keys are elements of $\mathbb{Z}_p$.

5.1 Construction

Let $\mathbb{G}$ be a bilinear group of prime order p.

Setup(k): The KGC picks a random generator $g \in \mathbb{G}$ and random elements $h, u, g_2, g_3, g_4 \in \mathbb{G}$. It randomly selects $\alpha \in \mathbb{Z}_p^*$ and defines $g_1 = g^\alpha$. The public parameters *params* (with the description of e, $\mathbb{G}$, $\mathbb{G}_1$, and p) and the secret *master-key* are given by

$$params = (g,\ g_1,\ g_2,\ g_3,\ g_4,\ h,\ u), \quad master\text{-}key = \alpha.$$

Extract-Partial-Private-Key(*master-key*, ID): To generate a partial private key for a user $\mathrm{ID} \in \mathbb{Z}_p$, the KGC selects random $r_{\mathrm{ID}} \in \mathbb{Z}_p$ and outputs the partial private key d_{ID}

$$d_{\mathrm{ID}} = (r_{\mathrm{ID}},\ h_{\mathrm{ID}}), \quad \text{where } \ h_{\mathrm{ID}} = (hg^{-r_{\mathrm{ID}}})^{1/(\alpha - \mathrm{ID})}.$$

If $\mathrm{ID} = \alpha$, the KGC aborts. As before, the KGC always uses the same random value r_{ID} for user ID.

Set-Secret-Value(ID, *params*): This algorithm picks a random $x_{\mathrm{ID}} \in \mathbb{Z}_p^*$, and outputs x_{ID} as a secret value.

Set-Private-Key(d_{ID}, *params*, x_{ID}): This algorithm outputs the (full) private key $SK_{\mathrm{ID}} = (x_{\mathrm{ID}},\ r_{\mathrm{ID}},\ h_{\mathrm{ID}})$ for user ID.

Set-Public-Key(ID, x_{ID}, *params*): This algorithm outputs the public key $PK_{\mathrm{ID}} = (X,\ Y)$ for user ID, where

$$X = (g_1 g^{-\mathrm{ID}})^{x_{\mathrm{ID}}}, \quad Y = u^{x_{\mathrm{ID}}}.$$

Encrypt(M, *params*, PK_{ID}): To encrypt a message $M \in \mathbb{G}_1$ under a public key PK_{ID}:

1. First check that the equality $e(X, u) = e(g_1 g^{-\mathrm{ID}}, Y)$ holds. If not, output $\perp$ and abort encryption.
2. Otherwise, run the *SigKeyGen* to obtain a signing key SigK and a verification key VerK.
3. Pick $s \in \mathbb{Z}_p^*$ at random and compute $C = (C_1, C_2, C_3, C_4, C_5)$, where

$$C_1 = X^s, \quad C_2 = e(g, g)^s, \quad C_3 = e(g, h)^{-s} \cdot M,$$
$$C_4 = (g_2 g_3^{-\mathsf{VerK}})^s, \quad C_5 = g_4^s.$$

4. Output the ciphertext $\mathsf{CT} = (C, Sign_{\mathsf{SigK}}(C), \mathsf{VerK})$.

As before, $e(g, g)$ and $e(g, h)$ can be also precomputed so that encryption algorithm requires two pairing computations in checking the validity of the public key.

Decrypt(CT, SK_{ID}, *params*): To decrypt a ciphertext $\mathsf{CT} = (C, \sigma, \mathsf{VerK})$ with $SK_{\mathrm{ID}} = (x_{\mathrm{ID}},\ r_{\mathrm{ID}},\ h_{\mathrm{ID}})$,

1. Run $Verify$ to check the validity of the signature σ on C, using the verification key VerK. If invalid, output $\perp$.

2. Otherwise, let $C = (C_1, C_2, C_3, C_4, C_5)$. Check two equalities $e(C_1, g_4) = e(C_5, (g_1 g^{-\mathrm{ID}})^{x_{\mathrm{ID}}})$ and $e(C_4, g_4) = e(C_5, g_2 g_3^{-\mathsf{VerK}})$ hold.
3. If either check fails, output $\perp$. Otherwise, output

$$M = e(C_1^{1/x_{\mathrm{ID}}},\ h_{\mathrm{ID}}) \cdot C_2^{r_{\mathrm{ID}}} \cdot C_3.$$

The correctness of decryption algorithm can be shown as in the IND-sID-CPA secure scheme,

$$e(C_1^{\frac{1}{x_{\mathrm{ID}}}}, h_{\mathrm{ID}}) \cdot C_2^{r_{\mathrm{ID}}} = e(g^{s \cdot x_{\mathrm{ID}} \cdot (\alpha - \mathrm{ID}) \cdot \frac{1}{x_{\mathrm{ID}}}}, (hg^{-r_{\mathrm{ID}}})^{\frac{1}{\alpha - \mathrm{ID}}}) \cdot e(g,g)^{s \cdot r_{\mathrm{ID}}} = e(g,h)^s$$

as required. The two bilinear checks during decryption are needed to ensure that C is a valid form in the security proofs. Using the same technique as in [18], we can save two pairing computations. In that case, the decryption algorithm selects two random values $r_1, r_2 \in \mathbb{Z}_p$ and checks that equality $e(C_1^{r_1} \cdot C_4^{r_2},\ g_4) = e(C_5,\ (g_1 g^{-\mathrm{ID}})^{x_{\mathrm{ID}} \cdot r_1} \cdot (g_2 g_3^{-\mathsf{VerK}})^{r_2})$ holds.

5.2 Security

Theorem 3. *Let $q = q_{EX} + q_{SK} + 1$. Suppose that the decision (t, q, ϵ_1)-BDHI assumption holds in $\mathbb{G}$ and the signature scheme is $(t, 1, \epsilon_2)$-strongly existentially unforgeable. Then the previous CL-PKE scheme is $(t, q_{EX}, q_{SK}, q_{PK}, q_R, q_D, \epsilon)$-selective-ID, adaptive chosen ciphertext secure against the Type I adversary, where $\epsilon_1 \geq \epsilon - \epsilon_2$.*

Proof. Suppose there exists an adversary $\mathcal{A}_I$ which has advantage ϵ in breaking the IND-sID-CCA security of the CL-PKE scheme in Section 5. We want to build an algorithm $\mathcal{B}$ that uses $\mathcal{A}_I$ to solve the decision q-BDHI problem in $\mathbb{G}$. On input $(g, g^\alpha, \ldots, g^{\alpha^q}, T) \in \mathbb{G}^{q+1} \times \mathbb{G}_1$ for some unknown $\alpha \in \mathbb{Z}_p^*$, $\mathcal{B}$ outputs 1 if $T = e(g,g)^{1/\alpha}$ and 0 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}_I$ in a selective-ID game as follows:

Init: $\mathcal{A}_I$ first outputs an identity $\mathrm{ID}^* \in \mathbb{Z}_p$ that it intends to attack.
Setup: To generate system parameters *params*, $\mathcal{B}$ does the following:

1. Select random $r_1, r_2, r_3, r_4, r_5 \in \mathbb{Z}_p^*$.
2. Run the *SigKeyGen* algorithm to obtain a signing key SigK^* and a verification key VerK^* (we assumed that $\mathsf{VerK}^* \in \mathbb{Z}_p$).
3. Generate a random polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree q. Expand the terms of $f(r_1 x + \mathrm{ID}^*)$ to get $f(r_1 x + \mathrm{ID}^*) = \sum_{i=0}^{q} c_i x^i$, where the constant term c_0 is non-zero. If $c_0 = 0$, try again with a new random polynomial $f(x)$.
4. Let $\alpha' = r_1 \cdot \alpha + \mathrm{ID}^*$ for some (unknown) $\alpha \in \mathbb{Z}_p^*$. Then, $\mathcal{B}$ computes $h = g^{f(\alpha')}$ with the values $g, g^\alpha, \ldots, g^{\alpha^q}$.

5. Set $g_1 = (g^{\alpha})^{r_1} g^{\mathrm{ID}^*} = g^{r_1 \cdot \alpha + \mathrm{ID}^*} = g^{\alpha'}$, $g_2 = (g^{\alpha})^{r_2} g^{r_3 \cdot \mathsf{VerK}^*}$, $g_3 = g^{r_3}$, $g_4 = (g^{\alpha})^{r_4}$, and $u = g^{r_5}$.
6. Publish $params = (g, g_1, g_2, g_3, g_4, h, u)$. Note that $master\text{-}key = \alpha'$.

Since α, $\{r_i\}$, and $f(x)$ are chosen uniformly at random, the values $\{g_i\}$, h, and u are uniformly random and this *params* has a distribution identical to that in the actual construction.

Phase 1: $\mathcal{A}_I$ issues extraction, private key, public key, replacement, and decryption queries.

1. For a extraction query, $\mathcal{B}$ can respond to a query on $\mathrm{ID} \neq \mathrm{ID}^*$ as follows. If $\mathrm{ID} = \alpha'$, $\mathcal{B}$ can easily solve decision q-BDHI problem (from obtaining the value α). Otherwise, let $f_{\mathrm{ID}}(x)$ be the $(q-1)$-degree polynomial $(f(x) - f(\mathrm{ID}))/(x - \mathrm{ID})$. $\mathcal{B}$ computes $r_{\mathrm{ID}} = f(\mathrm{ID})$ and $h_{\mathrm{ID}} = g^{f_{\mathrm{ID}}(\alpha')}$ with the values $g, g^{\alpha}, \ldots, g^{\alpha^{q-1}}$. $\mathcal{B}$ replies with a partial private key $d_{\mathrm{ID}} = (r_{\mathrm{ID}},\ h_{\mathrm{ID}})$. The validity of this partial private key is checked as $g^{f_{\mathrm{ID}}(\alpha')} = g^{(f(\alpha') - f(\mathrm{ID}))/(\alpha' - \mathrm{ID})} = (h g^{-f(\mathrm{ID})})^{1/(\alpha' - \mathrm{ID})}$.
2. When $\mathcal{A}_I$ makes a private key query for $\mathrm{ID} \neq \mathrm{ID}^*$, $\mathcal{B}$ picks a random $x_{\mathrm{ID}} \in \mathbb{Z}_p^*$ and returns $(x_{\mathrm{ID}},\ r_{\mathrm{ID}},\ h_{\mathrm{ID}})$ as the private key SK_{ID}. If necessary, $\mathcal{B}$ firstly computes a partial private key d_{ID} for ID.
3. For a public key query of ID (including ID^*), $\mathcal{B}$ replies with $X = (g_1 g^{-\mathrm{ID}})^{x_{\mathrm{ID}}}$ and $Y = u^{x_{\mathrm{ID}}}$ as $PK_{\mathrm{ID}} = (X, Y)$, with $x_{\mathrm{ID}} \in \mathbb{Z}_p^*$ corresponding to the private key SK_{ID}. This public key satisfies the equality $e(X, u) = e(g_1 g^{-\mathrm{ID}}, Y)$. Since x_{ID} is uniformly distributed among all elements in $\mathbb{Z}_p$, these public keys appear to $\mathcal{A}_I$ to be correctly distributed.
4. $\mathcal{A}_I$ can require $\mathcal{B}$ to replace the public key for ID (including ID^*) with its own choice. The validity of the replaced public key $PK_{\mathrm{ID}} = (X', Y')$ could be checked as $e(X', u) = e(g_1 g^{-\mathrm{ID}}, Y')$.
5. Finally, it remains to show that $\mathcal{B}$ can correctly respond to a decryption query for ID^*, where the public key PK_{ID^*} may be replaced by $\mathcal{A}_I$. In our security model, $\mathcal{B}$ must correctly respond to decryption queries even though the PK_{ID^*} has been replaced. Let $\mathsf{CT} = (C, \sigma, \mathsf{VerK})$ be a decryption query where $C = (C_1, C_2, C_3, C_4,\ C_5)$. Without loss of generality, we assume that the $PK_{\mathrm{ID}^*} = (X, Y)$ is valid. That is, it satisfies that $e(X, u) = e(g_1 g^{-\mathrm{ID}^*}, Y)$. Thus, for some $x_{\mathrm{ID}^*} \in \mathbb{Z}_p^*$ such that $Y = u^{x_{\mathrm{ID}^*}}$, $\mathcal{B}$ has that $X = (g_1 g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*}}$.

   $\mathcal{B}$ first runs $Verify$ to check the validity of the signature σ on C, using the verification key VerK. If the signature is invalid, $\mathcal{B}$ responds with $\perp$. Otherwise, $\mathcal{B}$ checks that two equalities $e(C_1, g_4) = e(C_5, X)$ and $e(C_4, g_4) = e(C_5, g_2 g_3^{-\mathsf{VerK}})$ hold (The first equality is the same as that of the decryption algorithm, because $X = (g_1 g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*}}$ for some $x_{\mathrm{ID}^*} \in \mathbb{Z}_p^*$. If the PK_{ID^*} has been replaced, the value x_{ID^*} is not known to $\mathcal{B}$). If either check fails, output $\perp$.

   Otherwise, if $\mathsf{VerK}^* = \mathsf{VerK}$, $\mathcal{B}$ outputs a random bit $b \in \{0, 1\}$ and aborts the simulation (actually, the forgery of one-time signature occurs).

Otherwise, since two equalities $e(C_1, g_4) = e(C_5, X)$ and $e(C_4, g_4) = e(C_5, g_2 g_3^{-\mathsf{VerK}})$ hold, it implies that $C_1 = X^s$ and $C_4 = (g_2 g_3^{-\mathsf{VerK}})^s$ for some (unknown) $s \in \mathbb{Z}_p$ such that $C_5 = g_4^s$. More precisely, $\mathcal{B}$ has

$$C_1 = (g_1 g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*} \cdot s} = \left(g^{r_1 \cdot \alpha + \mathrm{ID}^*} g^{-\mathrm{ID}^*}\right)^{x_{\mathrm{ID}^*} \cdot s} = g^{r_1 \cdot \alpha \cdot x_{\mathrm{ID}^*} \cdot s},$$
$$C_4 = (g_2 g_3^{-\mathsf{VerK}})^s = \left(g^{r_2 \cdot \alpha + r_3 \cdot \mathsf{VerK}^*} g^{-r_3 \cdot \mathsf{VerK}}\right)^s = g^{r_2 \cdot \alpha \cdot s} g^{r_3 \cdot (\mathsf{VerK}^* - \mathsf{VerK}) s},$$
$$C_5 = g_4^s = g^{r_4 \cdot \alpha \cdot s}.$$

$\mathcal{B}$ computes $C_5' = C_5^{r_2/r_4}$ and then $C_4' = C_4 / C_5' = g^{r_3 \cdot (\mathsf{VerK}^* - \mathsf{VerK}) s}$. Next, $\mathcal{B}$ checks that the following equality

$$C_2 \stackrel{?}{=} e(C_4', g)^{1/r_3 \cdot (\mathsf{VerK}^* - \mathsf{VerK})}$$

holds. If this equality does not hold, $\mathcal{B}$ knows that the C_2 is not of the right form. Then, $\mathcal{B}$ selects a random message $M \in \mathbb{G}_1$ and responds to the decryption query with M.
If the equality holds, $\mathcal{B}$ knows that $C_2 = e(g, g)^s$ for some (unknown) $s \in \mathbb{Z}_p^*$. $\mathcal{B}$ computes $C_5'' = C_5^{1/r_4} = g^{\alpha \cdot s}$ and

$$Z = e(\prod_{i=1}^{q} g^{c_i \cdot \alpha^{i-1}}, C_5'') \cdot C_2^{c_0}$$

where $\{c_i\}$ are derived from $f(r_1 x + \mathrm{ID}^*) = \sum_{i=0}^{q} c_i x^i$. Since $C_2 = e(g, g)^s$, $\mathcal{B}$ has that

$$Z = e(\prod_{i=1}^{q} g^{c_i \cdot \alpha^{i-1}}, g^{\alpha \cdot s}) \cdot e(g, g)^{s \cdot c_0} = e(g, g^{f(\alpha')})^s = e(g, h)^s.$$

Then, $\mathcal{B}$ replies to the decryption query with $Z \cdot C_3$.

Challenge: As before, $\mathcal{A}_I$ outputs two messages $M_0, M_1 \in \mathbb{G}_1$ for ID^*. For the public key $PK_{\mathrm{ID}^*} = (X, Y)$ (which may be replaced by $\mathcal{A}_I$), $\mathcal{B}$ has that $e(X, u) = e(g_1 g^{-\mathrm{ID}^*}, Y)$. Then, $\mathcal{B}$ has

$$X = (g_1 g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*}} = (g^{r_1 \cdot \alpha + \mathrm{ID}^*} g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*}} = g^{r_1 \cdot \alpha \cdot x_{\mathrm{ID}^*}},$$
$$Y = u^{x_{\mathrm{ID}^*}} = g^{r_5 \cdot x_{\mathrm{ID}^*}}$$

for some $x_{\mathrm{ID}^*} \in \mathbb{Z}_p^*$. Before the challenge ciphertext is constructed, $\mathcal{B}$ computes

$$T' = e(g, \prod_{i=0}^{q-1} g^{c_{i+1} \cdot \alpha^i}) \cdot T^{c_0}$$

(recall that the constant term c_0 is non-zero). Observe that if $T = e(g, g)^{1/\alpha}$, then we have

$$T' = e(g, g)^{f(\alpha')/\alpha} = e(g, h)^{1/\alpha}.$$

Next, $\mathcal{B}$ picks a random bit $b \in \{0,1\}$ and a random $l \in \mathbb{Z}_p^*$. After computing $C = (Y^{l \cdot r_1/r_5},\ T^l,\ T'^{-l} \cdot M_b,\ g^{r_2 \cdot l},\ g^{r_4 \cdot l})$, $\mathcal{B}$ constructs the challenge ciphertext $\mathsf{CT} = (C, Sign_{\mathsf{SigK}^*}(C), \mathsf{VerK}^*)$. Define $s = l/\alpha \in \mathbb{Z}_p^*$. If $T = e(g,g)^{1/\alpha}$, then $\mathcal{B}$ has

$$
\begin{aligned}
&Y^{l \cdot r_1/r_5} = (g^{r_5 \cdot x_{\mathrm{ID}^*}})^{l \cdot r_1/r_5} = g^{(r_1 \cdot \alpha \cdot x_{\mathrm{ID}^*})(l/\alpha)} = X^s,\\
&T^l = e(g,g)^{l/\alpha} = e(g,g)^s,\\
&T'^{-l} = e(g,h)^{-l/\alpha} = e(g,h)^{-s},\\
&g^{r_2 \cdot l} = g^{(r_2 \cdot \alpha + r_3 \cdot \mathsf{VerK}^* - r_3 \cdot \mathsf{VerK}^*)(l/\alpha)} = (g_2 g_3^{-\mathsf{VerK}^*})^s,\\
&g^{r_4 \cdot l} = g^{(r_4 \cdot \alpha)(l/\alpha)} = g_4^s.
\end{aligned}
$$

Therefore, CT is a valid ciphertext of M_b under the public key $PK_{\mathrm{ID}^*} = (X, Y)$, with the uniformly distributed random value $s = l/\alpha$. On the other hand, if T is random in $\mathbb{G}_1$, then T^l and $(T')^{-l}$ are just random elements of $\mathbb{G}_1$, independent of the bit b in the adversary's view. $\mathcal{B}$ returns to $\mathcal{A}_I$ the challenge ciphertext CT.

Phase 2: $\mathcal{A}_I$ issues more extraction, private key, public key, replacement, and decryption queries. $\mathcal{B}$ responds as in Phase 1.

Guess: $\mathcal{A}_I$ outputs a guess $b' \in \{0,1\}$. If $b = b'$ then $\mathcal{B}$ outputs 1, indicating $T = e(g,g)^{1/\alpha}$. Otherwise, it outputs 0, indicating $T \neq e(g,g)^{1/\alpha}$.

In the actual construction, the values $\{r_{\mathrm{ID}}\}$ are chosen independently from identities. Let $\mathcal{I}$ be a set consisting of α, ID^*, and the identities queried by $\mathcal{A}_I$ in extraction and private key queries; note that $|\mathcal{I}| \leq q+1$. Since $f(x)$ is a uniformly random polynomial of degree q, the values $\{f(\alpha) : a \in \mathcal{I}\}$ are uniformly random and independent. Thus, the values $\{r_{\mathrm{ID}}\}$ issued by $\mathcal{B}$ are appropriately distributed in the $\mathcal{A}_I$'s view. We notice that this analysis follows the proof in [20].

If T is random in $\mathbb{G}_1$, then $\Pr[\mathcal{B}(g, g^\alpha, \ldots, g^{\alpha^q}, T) = 0] = 1/2$. Let Forge denote the event that $\mathcal{A}_I$ submits a valid ciphertext $\mathsf{CT} = (C, \sigma, \mathsf{VerK}^*)$ as a decryption query. In the case of Forge, $\mathcal{B}$ cannot reply to the query and hence aborts the simulation. If Forge did not occur and $T = e(g,g)^{1/\alpha}$, $\mathcal{B}$ replied with a valid plaintext to $\mathcal{A}_I$. Then, $\mathcal{B}$ has

$$
\left|\Pr[\mathcal{B}(g, g^\alpha, \ldots, g^{\alpha^q}, T) = 0] - \frac{1}{2}\right| \geq \left|\Pr[b = b' \wedge \overline{\mathsf{Forge}}] - \frac{1}{2}\right| - \Pr[\mathsf{Forge}].
$$

Since $\mathcal{B}$ provided $\mathcal{A}_I$ with perfect simulation when event Forge did not occur, $\mathcal{B}$ has $|\Pr[b = b' \wedge \overline{\mathsf{Forge}}] - 1/2| \geq \epsilon$. Also, note that $\Pr[\mathsf{Forge}]$ is negligible. This means that $\Pr[\mathsf{Forge}] < \epsilon_2$ since otherwise, $\mathcal{B}$ can construct a forger, which is a contradiction to one-time signature. Therefore,

$$
\left|\Pr\left[\mathcal{B}(g, g^\alpha, \ldots, g^{\alpha^q}, e(g,g)^{1/\alpha}) = 0\right] - \Pr\left[\mathcal{B}(g, g^\alpha, \ldots, g^{\alpha^q}, T) = 0\right]\right| \geq \epsilon - \epsilon_2.
$$

This completes the proof of Theorem 3. □

Theorem 4. *Suppose that the decision* $(t, 1, \epsilon_1)$*-BDHI assumption holds in* $\mathbb{G}$ *and the signature scheme is* $(t, 1, \epsilon_2)$*-strongly existentially unforgeable. Then the previous CL-PKE scheme is* $(t, q_{SK}, q_{PK}, q_D, \epsilon)$*-selective-ID, adaptive chosen ciphertext secure against the Type II adversary, where* $\epsilon_1 \geq \epsilon - \epsilon_2$.

Security proof is provided in Appendix C.

6 Conclusion

We observed that all the known generic constructions of CL-PKE scheme do not guarantee the chosen ciphertext security against the Type I and II adversaries. Next, we presented a CL-PKE scheme that is selective-ID chosen plaintext secure without random oracles. This scheme was extended to obtain chosen ciphertext security using the idea of the CHK transformation. To prove the security of our proposed CL-PKE schemes, we defined a selective-ID security model which is a relaxed version of the AP-model.

Acknowledgement. The authors are very grateful to anonymous reviewers for helpful comments on earlier versions.

References

1. Al-Riyami, S.S.: Cryptographic schemes based on elliptic curve pairings. PhD thesis, University of London (2004)
2. Al-Riyami, S.S., Paterson, K.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
3. Al-Riyami, S.S., Paterson, K.: CBE from CL-PKE: A generic construction and efficient schems. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 398–415. Springer, Heidelberg (2005)
4. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005)
5. Bellarre, M., Boldyreva, A., Palacio, A.: An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004)
6. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen ciphertext secuirty from identity-based encryption. SIAM J. COMPUT 36(5), 1301–1328 (2006)
7. Bentahar, K., Farshim, P., Malone-Lee, J., Smart, N.P.: Generic construction of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058 (2005) http://eprint.iacr.org/2005/058
8. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
9. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

10. 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)
11. Canetti, C., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: Proceedings of the thirtieth anual ACM symposium on Theory of Computing, pp. 209–218 (1998)
12. Canetti, C., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) Advances in Cryptology – EUROCRPYT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
13. Canetti, C., 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)
14. Cheng, Z., Comley, R.: Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012 (2005) http://eprint.iacr.org/2005/012
15. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attacks. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
16. Dent, A.W.: A survey of certificateless encryption schemes and security models. Cryptology ePrint Archive, Report 2006/211 (2006) http://eprint.iacr.org/2006/211
17. Dodis, Y., Katz, J.: Chosen-ciphertext security of mutiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005)
18. Galindo, D., Kiltz, E.: Direct chosen ciphertext secure identity-based key encapsulation without random oracles. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 336–347. Springer, Heidelberg (2006)
19. Gentry, C.: Certificate-based encryption and the certificate revocation problem. In: Biham, E. (ed.) Advances in Cryptology – EUROCRPYT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
20. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
21. Goldwasser, S., Tauman, Y.: On the (In)security of the Fiat-Shamir Paradigm. In: FOC'03, pp. 102–115. IEEE Computer Society Press, Los Alamitos (2003)
22. Kiltz, E.: Chosen ciphertext secure identity-based encryption in the standard model with short ciphertexts. Cryptology ePrint Archive, Report 2006/122 (2006) http://eprint.iacr.org/2006/122
23. Liu, J.K., Au, M.H., Susilo, W.: Self-generated-certificate public key cryptography and certificateless signature / encryption scheme in the standard model. In: ACM AsiaCCS'07 (To appear)
24. Libert, B., Quisquater, J.J.: On construction certificateless cryptosystems from identity based encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 474–490. Springer, Heidelberg (2006)
25. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
26. Yum, D.H., Lee, P.J.: Generic construction of certificateless encryption. In: Laganà, A., Gavrilova, M., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) ICCSA 2004. LNCS, vol. 3043, pp. 802–811. Springer, Heidelberg (2004)
27. Yum, D.H., Lee, P.J.: Identity-based cryptography in public key management. In: Katsikas, S.K., Gritzalis, S., Lopez, J. (eds.) EuroPKI 2004. LNCS, vol. 3093, pp. 71–84. Springer, Heidelberg (2004)

A Definitions of PKE and IBE

Public Key Encryption: A PKE scheme consists of the following three algorithms.

$\mathsf{PK}^{\mathsf{Gen}}(k)$**:** takes a security parameter k and returns a public key PK and a private key SK.
$\mathsf{PK}^{\mathsf{Enc}}(PK, M)$**:** takes a public key PK and a message M as inputs. It returns a ciphertext CT.
$\mathsf{PK}^{\mathsf{Dec}}(\mathsf{CT}, SK)$**:** takes a ciphertext CT and the private key SK as inputs. It returns a message M or the symbol $\perp$.

Identity-Based Encryption: An IBE scheme consists of the following four algorithms.

$\mathsf{ID}^{\mathsf{Gen}}(k)$**:** takes a security parameter k and returns a public parameters *params* and a master key *master-key*.
$\mathsf{ID}^{\mathsf{Ext}}(\boldsymbol{master\text{-}key}, \mathrm{ID})$**:** takes a master key *master-key* and an identity ID. It returns a corresponding decryption key d_{ID}.
$\mathsf{ID}^{\mathsf{Enc}}(\mathrm{ID}, \boldsymbol{params}, M)$**:** takes an identity ID, a public parameters *params*, and a message M as inputs. It returns a ciphertext CT.
$\mathsf{ID}^{\mathsf{Dec}}(\mathsf{CT}, d_{\mathrm{ID}})$**:** takes a ciphertext CT and the private key d_{ID} as inputs. It returns a message M or the symbol $\perp$.

B Proofs of Chosen Plaintext Security

B.1 Proof of Theorem 1

Proof. Suppose there exists an adversary $\mathcal{A}_I$ which has advantage ϵ in attacking the IND-sID-CPA security of the CL-PKE scheme in Section 4. We want to build an algorithm $\mathcal{B}$ that uses $\mathcal{A}_I$ to solve the decision q-BDHI problem in $\mathbb{G}$. On input $(g, g^{\alpha}, \ldots, g^{\alpha^q}, T) \in \mathbb{G}^{q+1} \times \mathbb{G}_1$ for some unknown $\alpha \in \mathbb{Z}_p^*$, $\mathcal{B}$ outputs 1 if $T = e(g, g)^{1/\alpha}$ and 0 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}_I$ in a selective-ID game as follows:

Init: $\mathcal{A}_I$ first outputs an identity $\mathrm{ID}^* \in \mathbb{Z}_p$ that it intends to attack.
Setup: To generate system parameters *params*, $\mathcal{B}$ does the following:

1. Pick random $r_1, r_2 \in \mathbb{Z}_p^*$.
2. Generate a random polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree q. Expand the terms of $f(r_1 x + \mathrm{ID}^*)$ to get $f(r_1 x + \mathrm{ID}^*) = \sum_{i=0}^{q} c_i x^i$, where the constant term c_0 is non-zero. If $c_0 = 0$, try again with a new random polynomial $f(x)$.
3. Let $\alpha' = r_1 \cdot \alpha + \mathrm{ID}^*$ for some (unknown) $\alpha \in \mathbb{Z}_p^*$. $\mathcal{B}$ can compute $h = g^{f(\alpha')}$ with the values $g, g^{\alpha}, \ldots, g^{\alpha^q}$.
4. Compute $g_1 = (g^{\alpha})^{r_1} g^{\mathrm{ID}^*} = g^{r_1 \cdot \alpha + \mathrm{ID}^*} = g^{\alpha'}$ and $u = g^{r_2}$.
5. Publish $\mathit{params} = (g, g_1, h, u)$. Note that $\mathit{master\text{-}key} = \alpha'$.

Since α, r_1, r_2, and $f(x)$ are chosen uniformly at random, the values g_1, h, u, and α' are uniformly random so that this public key has a distribution identical to that in the actual construction.

Phase 1: $\mathcal{A}_I$ issues extraction, private key, public key, and replacement queries.

1. For a extraction query, $\mathcal{B}$ can respond to a query on $\mathrm{ID} \neq \mathrm{ID}^*$ as follows. If $\mathrm{ID} = \alpha'$, $\mathcal{B}$ can easily solve decision q-BDHI problem (from obtaining α). Otherwise, let $f_{\mathrm{ID}}(x)$ be the $(q-1)$-degree polynomial $(f(x) - f(\mathrm{ID}))/(x - \mathrm{ID})$. $\mathcal{B}$ computes $r_{\mathrm{ID}} = f(\mathrm{ID})$, and $h_{\mathrm{ID}} = g^{f_{\mathrm{ID}}(\alpha')}$ with the values $g, g^{\alpha}, \ldots, g^{\alpha^{q-1}}$. $\mathcal{B}$ replies with a partial private key $d_{\mathrm{ID}} = (r_{\mathrm{ID}}, h_{\mathrm{ID}})$. As in the IND-sID-CCA security proofs, we see that the partial private key d_{ID} for ID is valid.
2. When $\mathcal{A}_I$ makes a private key query for $\mathrm{ID} \neq \mathrm{ID}^*$, $\mathcal{B}$ picks a random $x_{\mathrm{ID}} \in \mathbb{Z}_p^*$ and returns $(x_{\mathrm{ID}}, r_{\mathrm{ID}}, h_{\mathrm{ID}})$ as the private key SK_{ID}. If necessary, $\mathcal{B}$ can firstly compute a partial private key d_{ID} for ID.
3. For a public key query of ID (including ID^*), $\mathcal{B}$ replies with $X = (g_1 g^{-\mathrm{ID}})^{x_{\mathrm{ID}}}$ and $Y = u^{x_{\mathrm{ID}}}$ as $PK_{\mathrm{ID}} = (X, Y)$, with $x_{\mathrm{ID}} \in \mathbb{Z}_p^*$ corresponding to the private key SK_{ID}. This public key satisfies the equality $e(X, u) = e(g_1 g^{-\mathrm{ID}}, Y)$. Since x_{ID} is uniformly distributed among all elements in $\mathbb{Z}_p$, these public keys appear to $\mathcal{A}_I$ to be correctly distributed.
4. Finally, $\mathcal{A}_I$ can require $\mathcal{B}$ to replace the public key for ID (including ID^*) with its own choice. The validity of the replaced public key $PK_{\mathrm{ID}} = (X', Y')$ could be checked as $e(X', u) = e(g_1 g^{-\mathrm{ID}}, Y')$.

Challenge: $\mathcal{A}_I$ outputs two messages $M_0, M_1 \in \mathbb{G}_1$ for ID^*. For the public key $PK_{\mathrm{ID}^*} = (X, Y)$ (which may be replaced by $\mathcal{A}_I$), without loss of generality, we assume that the PK_{ID^*} is valid, that is, $e(X, u) = e(g_1 g^{-\mathrm{ID}^*}, Y)$. If $Y = u^{x_{\mathrm{ID}^*}}$ for some $x_{\mathrm{ID}^*} \in \mathbb{Z}_p^*$ (If the PK_{ID^*} has been replaced, x_{ID^*} is not known to $\mathcal{B}$), $\mathcal{B}$ has

$$X = (g_1 g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*}} = (g^{r_1 \cdot \alpha + \mathrm{ID}^*} g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*}} = g^{r_1 \cdot \alpha \cdot x_{\mathrm{ID}^*}},$$
$$Y = u^{x_{\mathrm{ID}^*}} = g^{r_2 \cdot x_{\mathrm{ID}^*}}.$$

Before the challenge ciphertext is constructed, $\mathcal{B}$ computes $T' \in \mathbb{G}_1$ as

$$T' = e(g, \prod_{i=0}^{q-1} g^{c_{i+1} \cdot \alpha^i}) \cdot T^{c_0}$$

where $\{c_i\}$ are derived from $f(r_1 x + \mathrm{ID}^*) = \sum_{i=0}^{q} c_i x^i$ (Recall that the constant term c_0 is non-zero). Observe that if $T = e(g, g)^{1/\alpha}$, then $\mathcal{B}$ has

$$T' = e(g, g)^{f(\alpha')/\alpha} = e(g, h)^{1/\alpha}.$$

Now, $\mathcal{B}$ picks a random bit $b \in \{0, 1\}$ and a random $l \in \mathbb{Z}_p^*$. It responds with the ciphertext $\mathsf{CT} = (Y^{r_1 \cdot l / r_2},\ T^l,\ T'^{-l} \cdot M_b)$. Define $s = l/\alpha \in \mathbb{Z}_p^*$. If $T = e(g, g)^{1/\alpha}$ then $\mathcal{B}$ has

$$Y^{r_1 \cdot l / r_2} = (g^{r_2 \cdot x_{\mathrm{ID}^*}})^{(r_1 \cdot l \cdot \alpha / r_2 \cdot \alpha)} = g^{(r_1 \cdot \alpha \cdot x_{\mathrm{ID}^*})(l/\alpha)} = X^s,$$
$$T^l = e(g, g)^{l/\alpha} = e(g, g)^s,$$
$$T'^{-l} = e(g, h)^{-l/\alpha} = e(g, h)^{-s}.$$

Hence, CT is a valid ciphertext of M_b under the public key $PK_{\mathrm{ID}^*} = (X, Y)$, with the uniformly distributed random value $s = l/\alpha$. On the other hand, if T is random in $\mathbb{G}_1$, then T^l and $(T')^{-l}$ are just random elements of $\mathbb{G}_1$, independent of the bit b in the adversary's view.

Phase 2: $\mathcal{A}_I$ issues more extraction, private key, public key, and replacement queries. $\mathcal{B}$ responds as in Phase 1.

Guess : $\mathcal{A}_I$ outputs a guess $b' \in \{0, 1\}$. If $b = b'$ then $\mathcal{B}$ outputs 1, indicating $T = e(g, g)^{1/\alpha}$. Otherwise, it outputs 0, indicating $T \neq e(g, g)^{1/\alpha}$.

As in the IND-sID-CCA security proof, the values $\{r_{\mathrm{ID}}\}$ issued by $\mathcal{B}$ are appropriately distributed in the $\mathcal{A}_I$'s view. Next, we consider two cases. When $T = e(g, g)^{1/\alpha}$, $\mathcal{B}$ replies with a valid ciphertext using $T^l = e(g, g)^s$ and $T'^{-l} = e(g, h)^s$. So it must satisfy $|\Pr[b = b'] - 1/2| \geq \epsilon$. On the other hand, when T is random in $\mathbb{G}_1$, T^l and T'^{-l} are also random in $\mathbb{G}_1$ in which case $\Pr[b = b'] = 1/2$. Then, $\mathcal{B}$ has

$$\left|\Pr\left[\mathcal{B}(g, g^\alpha, \ldots, g^{\alpha^q}, e(g, g)^{1/\alpha}) = 0\right] - \Pr\left[\mathcal{B}(g, g^\alpha, \ldots, g^{\alpha^q}, T) = 0\right]\right| \geq \epsilon.$$

This concludes the proof of Theorem 1. □

B.2 Proof of Theorem 2

Proof. Suppose there exists an adversary $\mathcal{A}_{II}$ which has advantage ϵ in attacking the IND-sID-CPA security of the CL-PKE scheme in Section 4. We want to build an algorithm $\mathcal{B}$ that uses $\mathcal{A}_{II}$ to solve the decision 1-BDHI problem in $\mathbb{G}$. On input $(g, g^\alpha, T) \in \mathbb{G}^2 \times \mathbb{G}_1$ for some unknown $\alpha \in \mathbb{Z}_p^*$, $\mathcal{B}$ outputs 1 if $T = e(g, g)^{1/\alpha}$ and 0 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}_{II}$ in a selective-ID game as follows:

Init: $\mathcal{A}_{II}$ first outputs an identity $\mathrm{ID}^* \in \mathbb{Z}_p$ that it intends to attack.

Setup: To generate system parameters $params = (g, g_1, h, u)$ and *master-key*, $\mathcal{B}$ does the following:

1. Pick a random $\beta \in \mathbb{Z}_p^*$ that plays a role of the master key.
2. Select random $r_0, r_1, r_2 \in \mathbb{Z}_p^*$, and set $g_1 = g^\beta, h = (g^\alpha)^{r_0} g^{r_1}$, and $u = g^{r_2}$.
3. $\mathcal{B}$ gives $\mathcal{A}_{II}$ the $params = (g, g_1, h, u)$ and $master\text{-}key = \beta$.

Since $\{r_i\}$, and β are uniformly distributed among all elements in $\mathbb{Z}_p$, the values g_1, h, u, and β are uniformly random and identical to the actual construction.

Phase 1: $\mathcal{A}_{II}$ issues private key, public key, and decryption queries.

1. When $\mathcal{A}_{II}$ makes a private key query for $\mathrm{ID} \neq \mathrm{ID}^*$, $\mathcal{B}$ picks random x_{ID} and $r_{\mathrm{ID}} (\neq r_0 \cdot \alpha + r_1) \in \mathbb{Z}_p$. If $r_{\mathrm{ID}} = r_0 \cdot \alpha + r_1$, $\mathcal{B}$ obtains α to solve the decision 1-BDHI immediately. $\mathcal{B}$ replies with $SK_{\mathrm{ID}} = (x_{\mathrm{ID}}, r_{\mathrm{ID}}, h_{\mathrm{ID}})$, where $h_{\mathrm{ID}} = (hg^{-r_{\mathrm{ID}}})^{1/(\beta - \mathrm{ID})}$.

2. For the public key query of ID $\neq$ ID*, $\mathcal{B}$ computes $X = g^{(\beta-\mathrm{ID})x_{\mathrm{ID}}}$ and $Y = u^{x_{\mathrm{ID}}}$, and replies with $PK_{\mathrm{ID}} = (X, Y)$, using the value x_{ID} corresponding to the private key of ID. If ID = ID*, $\mathcal{B}$ selects a random $t \in \mathbb{Z}_p^*$ and computes $X^* = (g^\alpha)^{t(\beta-\mathrm{ID}^*)}$, $Y^* = (g^\alpha)^{t \cdot r_2}$. If we let $x_{\mathrm{ID}^*} = t \cdot \alpha$, we have that $X^* = (g_1 g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*}}$ and $Y^* = u^{x_{\mathrm{ID}^*}}$. $\mathcal{B}$ returns to the query on ID* as $PK_{\mathrm{ID}^*} = (X^*, Y^*)$.

It follows that $\mathcal{B}$ can generate all public keys for ID including ID*, and private keys for ID except ID*. In this procedure, x_{ID} and x_{ID^*} are uniformly distributed among all elements in $\mathbb{Z}_p$ and r_{ID} is uniform in $\mathbb{Z}_p \setminus \{r_0 \cdot \alpha + r_1\}$. These values are currently independent of $\mathcal{A}_{II}$'s view.

Challenge: $\mathcal{A}_{II}$ outputs two messages $M_0, M_1 \in \mathbb{G}_1$. $\mathcal{B}$ selects a random bit $b \in \{0,1\}$ and a random $l \in \mathbb{Z}_p^*$. $\mathcal{B}$ responds with the ciphertext $\mathsf{CT} = (g^{t \cdot l \cdot (\beta - \mathrm{ID}^*)},\ T^l,\ e(g,g)^{-r_0 \cdot l} \cdot T^{-r_1 \cdot l} \cdot M_b)$. Define $s = l/\alpha \in \mathbb{Z}_p^*$. If $T = e(g,g)^{1/\alpha}$ then $\mathcal{B}$ has

$$
\begin{aligned}
&g^{t \cdot l \cdot (\beta - \mathrm{ID}^*)} = g^{t \cdot \alpha \cdot (\beta - \mathrm{ID}^*) \cdot (l/\alpha)} = (X^*)^s, \\
&T^l = e(g,g)^{l/\alpha} = e(g,g)^s, \\
&e(g,g)^{-r_0 \cdot l} \cdot T^{-r_1 \cdot l} = e(g,g)^{-r_0 \cdot \alpha \cdot (l/\alpha)} \cdot e(g,g)^{-r_1 \cdot l/\alpha} = e(g,h)^{-s}.
\end{aligned}
$$

Therefore, CT is a valid encryption of M_b under the public key $PK_{\mathrm{ID}^*} = (X^*, Y^*)$. On the other hand, if T is random in $\mathbb{G}_1$, then T^l and $e(g,g)^{-r_0 \cdot l} \cdot T^{-r_1 \cdot l}$ are just random elements of $\mathbb{G}_1$, independent of the bit b in the $\mathcal{A}_{II}$'s view.

Phase 2: $\mathcal{A}_{II}$ issues more private/public key queries. $\mathcal{B}$ responds as in Phase 1.

Guess : $\mathcal{A}_{II}$ outputs a guess $b' \in \{0,1\}$. If $b = b'$ then $\mathcal{B}$ outputs 1, indicating $T = e(g,g)^{1/\alpha}$. Otherwise, it outputs 0, indicating $T \neq e(g,g)^{1/\alpha}$.

When $T = e(g,g)^{1/\alpha}$, $\mathcal{B}$ replies with a valid ciphertext using T. So it must satisfy $|\Pr[b = b'] - 1/2| \geq \epsilon$. On the other hand, when T is uniform and independent in $\mathbb{G}_1$, $\Pr[b = b'] = 1/2$. Therefore,

$$
\left| \Pr\left[\mathcal{B}(g, g^\alpha, e(g,g)^{1/\alpha}) = 0\right] - \Pr\left[\mathcal{B}(g, g^\alpha, T) = 0\right] \right| \geq \epsilon.
$$

This concludes the proof of Theorem 2. □

C Proofs of Chosen Ciphertext Security

C.1 Proof of Theorem 4

Proof. Suppose there exists an adversary $\mathcal{A}_{II}$ which has advantage ϵ in breaking the IND-sID-CCA security of the CL-PKE scheme in Section 5. We want to build an algorithm $\mathcal{B}$ that uses $\mathcal{A}_{II}$ to solve the decision 1-BDHI problem in $\mathbb{G}$. On input $(g, g^\alpha, T) \in \mathbb{G}^2 \times \mathbb{G}_1$ for some unknown $\alpha \in \mathbb{Z}_p^*$, $\mathcal{B}$ outputs 1

if $T = e(g,g)^{1/\alpha}$ and 0 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}_{II}$ in a selective-ID game as follows:

Init: $\mathcal{A}_{II}$ first outputs an identity $\mathrm{ID}^* \in \mathbb{Z}_p$ that it intends to attack.

Setup: To generate system parameters $params = (g, g_1, g_2, g_3, g_4, h, u)$ and *master-key*, $\mathcal{B}$ does the following:

1. Pick a random $\beta \in \mathbb{Z}_p^*$ that plays a role of the master key.
2. Run the *SigKeyGen* algorithm to obtain a signing key SigK^* and a verification key VerK^* (we assumed that $\mathsf{VerK}^* \in \mathbb{Z}_p$).
3. Select random $\{r_i\} \in \mathbb{Z}_p^*$ for $i = 0, 1, \ldots, 5.$ and set $h = (g^\alpha)^{r_0} g^{r_1}$, $g_1 = g^\beta$, $g_2 = (g^\alpha)^{r_2} g^{-r_3 \cdot \mathsf{VerK}^*}$, $g_3 = g^{r_3}$, $g_4 = (g^\alpha)^{r_4}$, and $u = g^{r_5}$.
4. $\mathcal{B}$ gives $params = (g, g_1, g_2, g_3, g_4, h, u)$ and $master\text{-}key = \beta$ to $\mathcal{A}_{II}$.

Since $\{r_i\}$ and β are uniformly distributed among all elements in $\mathbb{Z}_p$, the values $\{g_i\}$, h, u, and β are uniformly random and identical to the actual construction.

Phase 1: $\mathcal{A}_{II}$ issues private key, public key, and decryption queries.

1. When $\mathcal{A}_{II}$ makes a private key query for $\mathrm{ID} \neq \mathrm{ID}^*$, $\mathcal{B}$ picks random $x_{\mathrm{ID}} \in \mathbb{Z}_p^*$ and $r_{\mathrm{ID}} (\neq r_0 \cdot \alpha + r_1) \in \mathbb{Z}_p$. If $r_{\mathrm{ID}} = r_0 \cdot \alpha + r_1$, then $\mathcal{B}$ obtains α to solve the decision 1-BDHI immediately. Otherwise, $\mathcal{B}$ replies with $SK_{\mathrm{ID}} = (x_{\mathrm{ID}}, r_{\mathrm{ID}}, h_{\mathrm{ID}})$, where $h_{\mathrm{ID}} = (hg^{-r_{\mathrm{ID}}})^{1/(\beta - \mathrm{ID})}$.
2. For the public key query of $\mathrm{ID} \neq \mathrm{ID}^*$, $\mathcal{B}$ computes $X = g^{(\beta - \mathrm{ID})x_{\mathrm{ID}}}$ and $Y = u^{x_{\mathrm{ID}}}$, and replies with $PK_{\mathrm{ID}} = (X, Y)$, using the value x_{ID} corresponding to the private key of ID. If $\mathrm{ID} = \mathrm{ID}^*$, $\mathcal{B}$ selects a random $t \in \mathbb{Z}_p^*$ and computes $X^* = (g^\alpha)^{t(\beta - \mathrm{ID}^*)}$, $Y^* = (g^\alpha)^{t \cdot r_5}$. If we let $x_{\mathrm{ID}^*} = t \cdot \alpha$ for some unknown $\alpha \in \mathbb{Z}_p^*$, we have that $X^* = (g_1 g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*}}$ and $Y^* = u^{x_{\mathrm{ID}^*}}$. $\mathcal{B}$ returns to the query on ID^* as $PK_{\mathrm{ID}^*} = (X^*, Y^*)$.
3. $\mathcal{A}_{II}$ issues decryption queries. Let $\mathsf{CT} = (C, \sigma, VerK)$ be a decryption query and $C = (C_1, C_2, C_3, C_4, C_5)$. $\mathcal{B}$ first runs $Verify$ to check the validity of the signature σ on C, using the verification key VerK. If the signature is invalid, $\mathcal{B}$ responds with $\perp$. Next, $\mathcal{B}$ checks that two equalities $e(C_1, g_4) = e(C_5, X^*)$ and $e(C_4, g_4) = e(C_5, g_2 g_3^{-\mathsf{VerK}})$ hold. If either check fails, output $\perp$.

   Otherwise, if $\mathsf{VerK} = \mathsf{VerK}^*$, $\mathcal{B}$ outputs a random bit $b \in \{0, 1\}$ and aborts the simulation (actually, the forgery of one-time signature occurs). Otherwise, since the two equalities $e(C_1, g_4) = e(C_5, X^*)$ and $e(C_4, g_4) = e(C_5, g_2 g_3^{-\mathsf{VerK}})$ hold, it implies that $C_1 = (X^*)^s$, $C_4 = (g_2 g_3^{-\mathsf{VerK}})^s$, and $C_5 = g_4^s$ for some (unknown) $s \in \mathbb{Z}_p^*$. More precisely, $\mathcal{B}$ has

$$
\begin{aligned}
C_1 &= (g_1 g^{-\mathrm{ID}^*})^{x_{\mathrm{ID}^*} \cdot s} = \left(g^\beta g^{-\mathrm{ID}^*}\right)^{x_{\mathrm{ID}^*} \cdot s} = g^{(\beta - \mathrm{ID}^*) t \cdot \alpha \cdot s}, \\
C_4 &= (g_2 g_3^{-\mathsf{VerK}})^s = \left(g^{r_2 \alpha + r_3 \cdot \mathsf{VerK}^*} g^{-r_3 \cdot \mathsf{VerK}}\right)^s = g^{r_2 \alpha \cdot s} g^{r_3 \cdot (\mathsf{VerK}^* - \mathsf{VerK}) s}, \\
C_5 &= g_4^s = g^{r_4 \cdot \alpha \cdot s}.
\end{aligned}
$$

   As before, $\mathcal{B}$ computes $C_5' = C_5^{r_2 / r_4}$ and $C_4' = C_4 / C_5' = g^{r_3 \cdot (\mathsf{VerK}^* - \mathsf{VerK}) s}$. Next, $\mathcal{B}$ checks that the following equality

$$
C_2 \stackrel{?}{=} e(C_4', g)^{1/r_3 \cdot (\mathsf{VerK}^* - \mathsf{VerK})}
$$

holds. If this equality does not hold, $\mathcal{B}$ knows that the C_2 is not of the right form. Then, $\mathcal{B}$ selects a random message $M \in \mathbb{G}_1$ and responds to the decryption query with M.

If the equality holds, $\mathcal{B}$ knows that $C_2 = e(g,g)^s$ for some (unknown) $s \in \mathbb{Z}_p^*$. Then, $\mathcal{B}$ computes $C_5'' = C_5^{1/r_4} = g^{\alpha \cdot s}$ and $Z = e(g, C_5'')^{r_0} \cdot C_2^{r_1}$. We can check that Z becomes $e(g,h)^s$ as follows:

$$Z = e(g, g^{\alpha \cdot s})^{r_0} \cdot e(g,g)^{s \cdot r_1} = e(g, g^{r_0 \cdot \alpha + r_1})^s = e(g,h)^s.$$

Then, $\mathcal{B}$ replies to the decryption query with $Z \cdot C_3$.

Challenge: $\mathcal{A}_{II}$ outputs two messages $M_0, M_1 \in \mathbb{G}_1$ for ID^*. $\mathcal{B}$ picks a random bit $b \in \{0,1\}$ and a random $l \in \mathbb{Z}_p^*$. $\mathcal{B}$ computes $C = (g^{(\beta - \mathrm{ID}^*)t \cdot l},\ T^l,\ e(g,g)^{-r_0 \cdot l} \cdot T^{-r_1 \cdot l} \cdot M_b,\ g^{r_2 \cdot l},\ g^{r_4 \cdot l})$, and responds with the ciphertext $\mathsf{CT} = (C, Sign_{\mathsf{SigK}^*}(C), \mathsf{VerK}^*)$. Define $s = l/\alpha \in \mathbb{Z}_p^*$. If $T = e(g,g)^{1/\alpha}$, then $\mathcal{B}$ has

$$g^{(\beta - \mathrm{ID}^*)t \cdot l} = g^{(\beta - \mathrm{ID}^*)t \cdot \alpha \cdot (l/\alpha)} = (X^*)^s,$$
$$T^l = e(g,g)^{l/\alpha} = e(g,g)^s,$$
$$e(g,g)^{-r_0 \cdot l} \cdot T^{-r_1 \cdot l} = e(g,g)^{-r_0 \cdot \alpha \cdot (l/\alpha)} \cdot e(g,g)^{-r_1 \cdot l/\alpha} = e(g,h)^{-s},$$
$$g^{r_2 \cdot l} = g^{(r_2 \cdot \alpha + r_3 \cdot \mathsf{VerK}^* - r_3 \cdot \mathsf{VerK}^*)(l/\alpha)} = (g_2 g_3^{-\mathsf{VerK}^*})^s,$$
$$g^{r_4 \cdot l} = g^{(r_4 \cdot \alpha)(l/\alpha)} = g_4^s.$$

As before, CT is a valid ciphertext of M_b under the public key $PK_{\mathrm{ID}^*} = (X^*, Y^*)$, with the uniformly distributed random value $s = l/\alpha$. On the other hand, if T is random in $\mathbb{G}_1$, then T^l and $e(g,g)^{-r_0 \cdot l} \cdot T^{-r_1 \cdot l}$ are just random elements of $\mathbb{G}_1$, independent of the bit b in the adversary's view.

Phase 2: $\mathcal{A}_{II}$ issues private key, public key, and decryption queries. $\mathcal{B}$ responds as in Phase 1.

Guess : $\mathcal{A}_{II}$ outputs a guess $b' \in \{0,1\}$. If $b = b'$, then $\mathcal{B}$ outputs 1, indicating $T = e(g,g)^{1/\alpha}$. Otherwise, it outputs 0, indicating $T \neq e(g,g)^{1/\alpha}$.

If T is uniform and independent in $\mathbb{G}_1$, then $\Pr[\mathcal{B}(g, g^\alpha, T) = 0] = 1/2$. Let Forge denote the event that $\mathcal{A}_{II}$ submits a valid ciphertext $\mathsf{CT} = (C, \sigma, \mathsf{VerK}^*)$ as a decryption query. In case of Forge, $\mathcal{B}$ cannot reply to the query and aborts the simulation. If Forge did not occur and $T = e(g,g)^{1/\alpha}$, $\mathcal{B}$ replied with a valid plaintext to $\mathcal{A}_{II}$. Then, $\mathcal{B}$ has

$$\left| \Pr[\mathcal{B}(g, g^\alpha, T) = 0] - \frac{1}{2} \right| \geq \left| \Pr[b = b' \wedge \overline{\mathsf{Forge}}] - \frac{1}{2} \right| - \Pr[\mathsf{Forge}].$$

Since $\mathcal{B}$ provided $\mathcal{A}_{II}$ with perfect simulation when event Forge did not occur, $\mathcal{B}$ has $|\Pr[b = b' \wedge \overline{\mathsf{Forge}}] - 1/2| \geq \epsilon$. Also, as before, $\Pr[\mathsf{Forge}] < \epsilon_2$. Therefore,

$$\left| \Pr\left[\mathcal{B}(g, g^\alpha, e(g,g)^{1/\alpha}) = 0\right] - \Pr\left[\mathcal{B}(g, g^\alpha, T) = 0\right] \right| \geq \epsilon - \epsilon_2.$$

This completes the proof of Theorem 4. □

General and Efficient Certificateless Public Key Encryption Constructions

Zhaohui Cheng[1], Liqun Chen[2], Li Ling[3], and Richard Comley[1]

[1] School of Computing Science, Middlesex University, London, UK
{m.z.cheng,r.comley}@mdx.ac.uk
[2] Hewlett-Packard Laboratories, Bristol, UK
liqun.chen@hp.com
[3] Department of Communication Science and Engineering, Fudan University, Shanghai, China
lingli@fudan.edu.cn

Abstract. In 2003, Al-Riyami and Paterson introduced a new public key encryption paradigm called Certificateless Public Key Encryption (CL-PKE), which like Identity-Based Encryption (IBE) is certificate-free, and meanwhile which unlike IBE but similar to certificate-based encryption is key-escrow-free. In this paper, based on a heuristic observation on some existing IBE schemes and PKE schemes, we propose a general approach to build a CL-PKE solution, which makes use of a simple combination of an IBE scheme, a Diffie-Hellman type key establishment algorithm and a secure hash-function. Following this approach we construct two efficient concrete CL-PKE schemes and formally analyse their security in the random oracle model.

1 Introduction

To address the threat of the impersonation attack on the public key cryptography (PKC), a common strategy is to introduce into the system an authority trusted by all users. With the interventions of the authority, the impersonation attack launched by a malicious user can be thwarted by different methods.

One method is that the authority explicitly provide a guarantee that one user's ownership of a claimed public key is authentic. The certificate-based public key cryptography takes this approach. Each user obtains from the authority a certificate which securely binds the user identity with the user's public key by a signature generated by the authority. By this approach, an infrastructure to issue certificates has to be constructed and also one has to verify certificates to obtain others' authentic public keys. Such infrastructure can be very complicated and faces many challenges in practice, such as the efficiency and scalability of the infrastructure.

The second method is that users use their identity directly as their public keys and so the public key authenticity problem is trivial and certificates are no longer necessary. However, each user's private key has to be generated by the authority. This is the approach taken by the identity-based cryptography (IBC).

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 83–107, 2007.

However, in this type of system, the authority knows every user's private key, i.e., the system has the inherent key-escrow function, and no method can prevent a curious authority from decrypting users' ciphertexts or impersonating a user.

Though the IBC paradigm offers great advantage of simplicity over the certificate-based PKC, the key escrow property is not desirable in some settings. The natural question arises that whether a public key system as IBC certificate-free and at the same time as PKC key-escrow-free is constructible. In 2003, Al-Riyami and Paterson brought forth the notion of "Certificateless Public Key Cryptography" (CL-PKC) [3] to respond to this challenge. In the CL-PKC, a user has a public key generated by himself and his private key is determined by two pieces of secret information: one secret associated with the user's identity is passed by the authority and the other associated with the public key is generated by the user himself. Moreover, one secret is not computable from the other, so the authority cannot compute the private key corresponding to a user's public key. Hence the CL-PKC is key-escrow-free.

The approach against the impersonation attack in the CL-PKC is not to provide authenticity of a public key by a certificate. Instead, a CL-PKC guarantees that even if a malicious user successfully replaces a victim's public key with its own choice and so could know the secret associated with the public key but not the other secret obtained by the victim user from the authority, it still cannot generate a valid signature or decrypt the message encrypted under the false public key and the victim's identifier. This will certainly reduce the interest of launching the impersonation attack.

Since the certificateless public key encryption (CL-PKE) notion was introduced, there have been a number of generic constructions. In [1,29], three general constructions of CL-PKE are constructed. They are the sequential or parallel composition of a secure identity-based encryption (IBE) with a secure public key encryption (PKE) to encrypt a message. Unfortunately, none of those generic constructions is secure [22,25] regarding the model defined in [3]. Libert and Quisquater [25] showed simple variants can rescue them. Yum and Lee proposed yet another generic construction but by double-encryption with two secure IBE schemes [30], which is also found insecure [22]. In [7], Bentahar *et al.* extended the key encapsulation mechanism (KEM) to the CL-PKE setting and proposed a generic construction from an IBE and a PKE. There have been a couple of general constructions in the standard model as well [15,24]. All of these general constructions are not very efficient both on computation and communication. They all need double-encryption with either two IBE schemes or one IBE with a PKE, and the ciphertext of these schemes are longer than the used IBE or PKE's. In this work, based on a heuristic observation on some constructions of IBE and PKE, we propose a general approach of constructing efficient CL-PKE, specifically, we can make use of a hash function to tightly integrate an IBE with a PKE of similar ciphertext structure to form a CL-PKE. Following this approach, we construct two efficient CL-PKE schemes and formally analyse their security in an enhanced security model.

The paper is constructed as follows. In Section 2 we recap the basic facts of pairing and various CL-PKE security formulations. In Section 3, we present a heuristic approach of constructing CL-PKE from IBE and PKE schemes. Then we construct two efficient concrete CL-PKE schemes and provide a formal security proof of them in Section 4. After that, we discuss relevant efficiency of the CL-PKE proposals. Finally, we draw a conclusion.

2 Preliminaries

2.1 Pairing

Here we briefly recall some basic facts of pairings.

Definition 1. *A pairing is a bilinear map* $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ *between three groups* $\mathbb{G}_1$, $\mathbb{G}_2$ *and* $\mathbb{G}_t$ *of exponent* p, *which has the following properties:*

1. *Bilinear: For all* $(P_1, P_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ *and for all* $(a, b) \in \mathbb{Z}_p \times \mathbb{Z}_p$, *we have* $\hat{e}(aP_1, bP_2) = \hat{e}(P_1, P_2)^{ab}$.
2. *Non-degenerate: There exist non-trivial points* $P_1 \in \mathbb{G}_1$ *and* $P_2 \in \mathbb{G}_2$ *both of order* q *such that* $\hat{e}(P_1, P_2) \neq 1$.
3. *Computable: For all* $(P_1, P_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, $\hat{e}(P_1, P_2)$ *is efficiently computable.*

We shall use following assumptions to analyse the proposed schemes. Each problem is assumed to be defined for a given set of pairing parameters including the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$, the generators $P_1 \in \mathbb{G}_1$ and $P_2 \in \mathbb{G}_2$, the pairing $\hat{e}$, and possibly the morphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.

Assumption 1 (Diffie–Hellman ($\mathbf{DH}_{i,j,k}$)). *For* $a, b \in_R \mathbb{Z}_p$ *and some values of* $i, j, k \in \{1, 2\}$, *given* (aP_i, bP_j), *computing* abP_k *is hard.*

Assumption 2 (Bilinear Diffie–Hellman ($\mathbf{BDH}_{i,j,k}$)). *For* $a, b, c \in_R \mathbb{Z}_p$, *given* (aP_i, bP_j, cP_k), *for some values of* $i, j, k \in \{1, 2\}$, *computing* $\hat{e}(P_1, P_2)^{abc}$ *is hard.*

Assumption 3 (General BDH). *For* $a, b, c \in_R \mathbb{Z}_p$, *given* (aP_1, cP_1, aP_2, bP_2), *computing* $\hat{e}(P_1, P_2)^{abc}$ *is hard.*

We note that if an efficient isomorphism ψ exists, then the $\mathrm{BDH}_{2,2,2}$ assumption implies the General BDH assumption.

2.2 CL-PKE Security Model

Here we first specify the CL-PKE algorithms and then revisit various CL-PKE security models and define a strong security notion for this type of encryption.

For a CL-PKE scheme we define the public key, message, ciphertext and randomness spaces by $\mathbb{P}_{\mathtt{CL}}(\cdot)$, $\mathbb{M}_{\mathtt{CL}}(\cdot)$, $\mathbb{C}_{\mathtt{CL}}(\cdot)$ and $\mathbb{R}_{\mathtt{CL}}(\cdot)$. These spaces are

parametrised by the master public key $M_{\mathtt{pk}}$, and hence by the security parameter k. A CL-PKE scheme consists of following algorithms:

- **CL.Gen**(1^k). The algorithm given the system security parameter k generates the master secret key $M_{\mathtt{sk}}$ and the master public key $M_{\mathtt{pk}}$.
- **CL.PartialKey**$(M_{\mathtt{sk}}, M_{\mathtt{pk}}, \mathtt{ID}_A)$. The algorithm takes $M_{\mathtt{sk}}$, $M_{\mathtt{pk}}$ and an arbitrary identity string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A as input and returns a partial private key D_A corresponding to $\mathtt{ID}_A$.
- **CL.SecretVal**$(M_{\mathtt{pk}}, \mathtt{ID}_A)$. The algorithm takes $M_{\mathtt{pk}}$ and the identity string $\mathtt{ID}_A$ as input and returns the secret value X_A associated with the entity A.
- **CL.PrivateKey**$(M_{\mathtt{pk}}, D_A, X_A)$. The algorithm takes $M_{\mathtt{pk}}$, D_A and X_A as input and outputs the private key S_A of entity A.
- **CL.PublicKey**$(M_{\mathtt{pk}}, X_A, \mathtt{ID}_A)$. The algorithm takes $M_{\mathtt{pk}}$ and X_A as input and outputs the public key P_A of the entity A.
- **CL.Encrypt**$(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, m;\ r)$. The algorithm takes $M_{\mathtt{pk}}$, $\mathtt{ID}_A$, P_A, a message $m \in \mathbb{M}_{\mathtt{CL}}(M_{\mathtt{pk}})$ and the randomness $r \in \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}})$ as input and returns the ciphertext $C \in \mathbb{C}_{\mathtt{CL}}(M_{\mathtt{pk}})$ of message m. We also use the interface **CL.Encrypt** $(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, m)$ by assuming that r is sampled in the algorithm when the context is clear.
- **CL.Decrypt**$(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. The algorithm takes $M_{\mathtt{pk}}$, $\mathtt{ID}_A$, S_A and a ciphertext C as input, and outputs the value of the corresponding plaintext m or a failure symbol $\perp$.

Similar to IBE and PKE, to cope with probabilistic ciphers, we will require that not too many choices for r encrypt a given message to a given ciphertext. To formalize this concept we let $\gamma(M_{\mathtt{pk}})$ be the least upper bound such that $|\ \{r \in \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}}) : \mathbb{E}_{\mathtt{CL}}(M_{\mathtt{pk}}, \mathtt{ID}, P_{\mathtt{ID}}, m;\ r) = C\}\ | \leq \gamma(M_{\mathtt{pk}})$ for every $\mathtt{ID}$, $P_{\mathtt{ID}} \in \mathbb{P}_{\mathtt{CL}}(M_{\mathtt{pk}})$, $m \in \mathbb{M}_{\mathtt{CL}}(M_{\mathtt{pk}})$ and $C \in \mathbb{C}_{\mathtt{CL}}(M_{\mathtt{pk}})$. We say a CL-PKE is γ**-uniform** if $\gamma(M_{\mathtt{pk}}) / |\ \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}})\ | < \gamma$. In this work, we require that γ is negligible of security parameter k.

Following Al-Riyami-Paterson's CL-PKE security formulation, we can define various security notions for this type of encryption. These security notions are defined by two games as in Table 1 and 2. Game 1 is conducted between a challenger and a Type-I adversary $\mathcal{A}_I$ of two PPT algorithms $(\mathcal{A}_{I_1}, \mathcal{A}_{I_2})$. A Type-I adversary does not know the master secret key and can replace an entity's public key with its choice. Game 2 is conducted between a challenger and a Type-II adversary $\mathcal{A}_{II}$ of two PPT algorithms $(\mathcal{A}_{II_1}, \mathcal{A}_{II_2})$. A Type-II adversary as a malicious KGC knows the master secret key (so every entity's partial private key) and intends to decrypt a user's ciphertext.

In the games, s is some state information and $\mathcal{O}_{\mathtt{CL}}$ are the oracles that the adversary can access during the game. Depending on the security model, these oracles may include the follows:

- a public key broadcast oracle *Public-Key-Broadcast* which takes as input an identifier and returns the associated public key. If necessary, the oracle will execute the **CL.PublicKey** algorithm first.

Table 1. IND CL-PKE Games

Game 1: Type-I Adversarial	Game 2: Type-II Adversarial
1. $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow$ **CL.Gen**(1^k).	1. $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow$ **CL.Gen**(1^k).
2. $(s, \mathtt{ID}^*, m_0, m_1) \leftarrow \mathcal{A}_{I-1}^{\mathcal{O}_{\mathtt{CL}}}(M_{\mathtt{pk}})$.	2. $(s, \mathtt{ID}^*, m_0, m_1) \leftarrow \mathcal{A}_{II-1}^{\mathcal{O}_{\mathtt{CL}}}(M_{\mathtt{pk}}, M_{\mathtt{sk}})$.
3. $b \leftarrow \{0,1\}, r \leftarrow \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}})$.	3. $b \leftarrow \{0,1\}, r \leftarrow \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}})$.
4. $C^* \leftarrow$ **CL.Encrypt**$(M_{\mathtt{pk}}, \mathtt{ID}^*, P_{\mathtt{ID}^*}, m_b;\ r)$.	4. $C^* \leftarrow$ **CL.Encrypt**$(M_{\mathtt{pk}}, \mathtt{ID}^*, P_{\mathtt{ID}^*}, m_b;\ r)$.
5. $b' \leftarrow \mathcal{A}_{I-2}^{\mathcal{O}_{\mathtt{CL}}}(s, M_{\mathtt{pk}}, C^*, \mathtt{ID}^*, P_{\mathtt{ID}^*}, m_0, m_1)$.	5. $b' \leftarrow \mathcal{A}_{II-2}^{\mathcal{O}_{\mathtt{CL}}}(s, M_{\mathtt{pk}}, M_{\mathtt{sk}}, C^*, \mathtt{ID}^*, P_{\mathtt{ID}^*}, m_0, m_1)$.

Table 2. OW CL-PKE Games

Game 1: Type-I Adversarial	Game 2: Type-II Adversarial
1. $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow$ **CL.Gen**(1^k).	1. $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow$ **CL.Gen**(1^k).
2. $(s, \mathtt{ID}^*) \leftarrow \mathcal{A}_{I-1}^{\mathcal{O}_{\mathtt{CL}}}(M_{\mathtt{pk}})$.	2. $(s, \mathtt{ID}^*) \leftarrow \mathcal{A}_{II-1}^{\mathcal{O}_{\mathtt{CL}}}(M_{\mathtt{pk}}, M_{\mathtt{sk}})$.
3. $m \leftarrow \mathbb{M}_{\mathtt{CL}}(M_{\mathtt{pk}}), r \leftarrow \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}})$.	3. $m \leftarrow \mathbb{M}_{\mathtt{CL}}(M_{\mathtt{pk}}), r \leftarrow \mathbb{R}_{\mathtt{CL}}(M_{\mathtt{pk}})$.
4. $C^* \leftarrow$ **CL.Encrypt**$(M_{\mathtt{pk}}, \mathtt{ID}^*, P_{\mathtt{ID}^*}, m;\ r)$.	4. $C^* \leftarrow$ **CL.Encrypt**$(M_{\mathtt{pk}}, \mathtt{ID}^*, P_{\mathtt{ID}^*}, m;\ r)$.
5. $m' \leftarrow \mathcal{A}_{I-2}^{\mathcal{O}_{\mathtt{CL}}}(s, M_{\mathtt{pk}}, C^*, \mathtt{ID}^*, P_{\mathtt{ID}^*})$.	5. $m' \leftarrow \mathcal{A}_{II-2}^{\mathcal{O}_{\mathtt{CL}}}(s, M_{\mathtt{pk}}, M_{\mathtt{sk}}, C^*, \mathtt{ID}^*, P_{\mathtt{ID}^*})$.

- a partial key exposure oracle *Partial-Private-Key-Extract* which returns the partial private key associated with an identity. If necessary, the oracle will execute the **CL.PartialKey** algorithm first. This oracle is only useful to Type-I adversaries, as a Type-II adversary can compute every partial private key using the master secret key.
- a secret value exposure oracle *Secret-Value-Extract* which reveals the secret value of entity whose public key was not replaced. If necessary, the algorithm will execute algorithm **CL.SecretVal** first.
- a public key replace oracle *Public-Key-Replace* which takes as input an identifier and a public key from the public key space and replaces the current public key associated with the identifier with the provided key.
- a strong decryption oracle $Decrypt^S$ which takes as input a ciphertext and an identifier, and outputs the decryption of the ciphertext using the the *current* private key associated with the identifier. Note that in the games, the adversary may have replaced the public key associated with an identity, and this decryption oracle is required to output the correct decryption (which can be failure symbol as well) using the private key corresponding to the current public key, even if it may not know that the corresponding secret value. As this oracle does not reflect general practice, a normal decryption oracle is defined as follow:
- a decryption oracle $Decrypt^P$ which takes as input a ciphertext and an identifier, and outputs the decryption of the ciphertext using the the original (before any *Public-Key-Replace* query) private key associated with the identifier. Though this query reflects the common practice that given a ciphertext a party uses its own private key to decrypt it, some attacks of conceptional

interest are not simulated (see the attack on the second Al-Riyami-Paterson CL-PKE [4], which we call AP-CL-PKE2, in Appendix A.2). A conceptional decryption oracle is used in some formulation as follows:

– a decryption oracle $Decrypt^C$ which takes as input a ciphertext, an identifier, a public key and the secret value corresponding to the given public key, and outputs the decryption of the ciphertext using the private key determined by the partial key corresponding to the identifier and the given secret value in the query. If the secret value is not given in the query, then the oracle works as $Decrypt^P$.

For each type of adversary, we define two attack models in which the adversary is allowed to access different oracles.

For Type-I adversaries, we define the following two attack models.

– CCA2 model. In this model, the adversary is allowed to access the *Public-Key-Broadcast*, *Partial-Private-Key-Extract*, *Secret-Value-Extract*, *Public-Key-Replace* and $Decrypt^S$ oracles. However, there are a few restrictions on the adversary.
  - If *Public-Key-Replace* has been issued on an identity, then *Secret-Value-Extract* on the identity is disallowed.
  - *Partial-Private-Key-Extract* on $\mathtt{ID}^*$ is disallowed.
  - $Decrypt^S$ on $(\mathtt{ID}^*, C^*)$ is not allowed in $\mathcal{A}_{I-2}$ when $\mathtt{ID}^*$'s current public key $P_{\mathtt{ID}^*}$ is the same as when the challenge query is issued.
– CPA model. In this model, the adversary has the access to the similar oracles as in the CCA2 model, but the $Decrypt^S$ oracle is disallowed.

As the strong decryption oracle $Decrypt^S$ may not reflect the practice, we can define two weaker CCA2 models Type-I^P and Type-I^C in which the adversary is allowed to access the $Decrypt^P$ and $Decrypt^C$ oracle instead of $Decrypt^S$ respectively.

For Type-II adversaries we define the following two attack models.

– CCA2 model. In this model, the adversary is allowed to access the *Public-Key-Broadcast*, *Secret-Value-Extract*, *Public-Key-Replace* and $Decrypt^S$ oracles. Again, there are a few restrictions on the adversary.
  - If *Public-Key-Replace* has been issued on an identity, then *Secret-Value-Extract* on the identity is disallowed.
  - *Public-Key-Replace* on $\mathtt{ID}^*$ is disallowed.
  - *Secret-Value-Extract* on $\mathtt{ID}^*$ is disallowed.
  - $Decrypt^S$ on $(\mathtt{ID}^*, C^*)$ is not allowed in $\mathcal{A}_{II-2}$.
– CPA model. In this model, the adversary has the access to the similar oracles as in the CCA2 model, but the $Decrypt^S$ oracle is disallowed.

Similarly, we can define a weaker CCA2 model Type-II^P in which the adversary is allowed to access the $Decrypt^P$ oracle instead. There is no interest of defining Type-II^C security as it requires the adversary to know both the partial key (because of the knowledge of the master secret key) and the secret value which implies the adversary can decrypt the ciphertext on its own.

If we let MOD denote the mode of attack, either CPA or CCA2, the adversary's advantage in the indistinguishability-based game is defined to be

$$\mathtt{Adv}_{\mathtt{CL}}^{\mathtt{CL-IND-MOD}}(\mathcal{A}) = \mid 2\Pr[b' = b] - 1 \mid,$$

while, the advantage in the one-way game is given by

$$\mathtt{Adv}_{\mathtt{CL}}^{\mathtt{CL-OW-MOD}}(\mathcal{A}) = \Pr[m' = m].$$

A CL-PKE algorithm is considered to be secure, in the sense of a given goal and attack model (CL-IND-CCA2 for example) if, for any PPT Type-I (and Type-II) adversary, the advantage in the relevant game is a negligible function of the security parameter k.

There are two main differences between the model defined above and the Al-Riyami-Paterson's CL-PKE security model [3,4]. First, in the Al-Riyami-Paterson's model a private key exposure oracle which returns the private key of an entity is used, while, here we provide the secret value exposure oracle instead. As in CL-PKE each entity has two pieces of secret information, it is natural to provide the adversary with an exposure oracle for each secret. Because the entity private key is determined by two secrets, the *Secret-Value-Extract* oracle with the *Partial-Key-Extract* oracle certainly can simulate the private key exposure oracle. On the other hand, given a private key and the corresponding partial key, the adversary may not be able to recover the related secret value if the **CL.PrivateKey** algorithm is a one way function such as the algorithm used in [3]. Hence, the *Secret-Value-Extract* oracle provides extra capability to the adversary against certain schemes. The second difference is that in Game 2 above the adversary can access the *Public-Key-Replace* oracle and the strong decryption oracle $Decrypt^S$ (this formulation has been adopted in a number of other works [25,14,2]), while, in the Al-Riyami-Paterson's model the *Public-Key-Replace* oracle is disallowed and the decryption oracle $Decrypt^P$ is used instead. A trivial Type-II attack applicable in the enhanced model on AP-CL-PKE2 (see Appendix A.2.) shows that the new formulation defines a stronger model in theory.

On the relation of different security formulations, the Type-I (resp. Type-II) security is certainly stronger than the Type-I^P and Type-I^C (resp. Type-II^P) security. Because of the behavior of $Decrypt^C$, the Type-I^C security is at least as strong as the Type-I^P one. The Type-I^C attack on AP-CL-PKE2 shows that the Type-I^C security is indeed stronger than Type-I^P.

Recently, Au *et al.* [2] made an interesting observation that a *passive-but-malicious* KGC may generate the master public/private key pair in a special way to help it decrypt some user's ciphertext. And they presented such an attack against the first Al-Riyami-Paterson CL-PKE [3] (we call it AP-CL-PKE1). The attack shows that it is meaningful to let a Type-II$^+$ adversary generate the master keys in Game 2. However, it remains an open problem whether a secure CL-PKE against both Type-I and Type-II$^+$ adversaries is constructible in the standard model. Even in the random oracle model, this strong security notion

might be difficult to achieve. We note that when the Type-II$^+$ adversary generates the master public key, a question arises that who controls any random oracle included in the master public key. In [2], a generic CL-PKE scheme is claimed to be secure in this strong model. However, no proof is given and it is unclear who (the challenger or the adversary) controls the random oracles.

On the other hand, Au *et al.*'s attack can be defeated by requiring the KGC to demonstrate the randomness of the choice of parameters. In particular, the attack against AP-CL-PKE1 requires the adversary to choose from the used group a specific generator P, which supposes to be random. For its innocence the KGC can show a witness of the randomness of the generator such as a public string S with $P = H(S)$ for a cryptographic hash function H. One can refer to IEEE P1363 and ANSI X9.62 standards for examples of methods used to generate verifiably random parameters.

In this work, we adopt the enhanced Al-Riyami-Paterson formulation (the Type-I+Type-II security model) and conduct the security analysis of the proposed schemes in the random oracle model.

3 Heuristic Approach of CL-PKE

Now we explain our heuristic approach of constructing CL-PKE. This approach is based on a simple observation of some existing IBE and PKE schemes.

Most of the discrete-logarithm based PKE schemes, e.g. the ElGamal encryption [16], essentially take the same general approach, which can be presented in a simple equation as follows:

$$\text{PKE ciphtext}=\langle \text{DH token(s)}, \text{Hiding(message; DH value)}\rangle,$$

where a DH token is defined as an input to a DH key establishment protocol, and a DH value is defined as a result of the protocol; a well-known example is that two entities exchange their DH tokens g^x and g^y respectively and then compute a DH value g^{xy}. In the above general PKE approach, the encrypter generates one or more DH tokens and uses the DH token(s) and the decrypter's public key to compute the DH value. Then the DH value is used as the secret to hide messages in a message hiding algorithm. The decrypter uses its private key and the DH token(s) in the ciphertext to compute the DH value and so to recover the conveyed message.

Most of the existing IBE schemes, e.g. [6,11,5], make use of pairings and base their security on the BDH assumption or its variants which are the descendants of the DH assumption. These IBE encryption schemes essentially adopt the same approach, like those PKE schemes based on the DH assumption, which can be presented as:

$$\text{IBE ciphtext}=\langle \text{pairing-DH token(s)}, \text{Hiding(message; pairing-DH value)}\rangle,$$

where a pairing-DH token and pairing-DH value is a variety of the DH token and DH value. Again, the encrypter first computes one or more pairing-DH tokens, then computes a pairing-DH value, which can be computed through pairings by

the decrypter with its priavate key, the pairing-DH token(s) and possibly the system parameters as well. The pairing-DH value is used as the secret to hide the message by the encrypter in a message hiding algorithm and to recover the conveyed message by the decrypter.

We can see that pairing-based IBE and DH-based PKE schemes have a common structure and some generic constructions such as the hybrid encryption (KEM-DEM) [13,7], the Fujisaki-Okamoto (FO) conversions [17,18,26,31] and REACT [27,26] can be used in to construct both PKE and IBE. In [5], Boyen classified the existing IBE schemes from pairing into three categories: full-domain hash IBE, exponent-inverse IBE and commutative blinding IBE. In each category, some efficient schemes exactly make use of constructions that are applicable in both IBE and PKE. For example, both BF-IBE [6,21] which is a full-domain hash IBE and SK-IBE [11] which is an exponent-inverse IBE adopt the FO-conversions. BB_1-IBE, which is a commutative blinding IBE, takes an approach that can also be used to construct the ElGamal-like secure PKE. BF-KEM [5], SK-KEM [12] and BB_1-KEM [5] all follow the hybrid encryption construction.

Based on the above observation, it seems natural to use a hash function to integrate a secure IBE scheme with a secure PKE scheme to achieve a secure CL-PKE scheme, which can be presented as follows:

$$\text{CL-PKE ciphtext} = \langle \text{PKE.DH-tokens(s)}, \text{IBE.pairing-DH token(s)}, \text{Hiding}'(\text{message}; H(\text{DH value}, \text{pairing-DH value}))\rangle$$

where H is a hash function, a PKE.DH-token is a DH token used in a PKE scheme and an IBE.pairing-DH token is a pairing-DH token used in an IBE scheme. The interesting bit is that the PKE.DH-token(s) and IBE.pairing-DH token(s) could be generated in a simple way, where they can share the same randomness. The message hiding algorithm might need to be slightly modified as the input secret value is no longer a DH or pairing-DH value but a hash-function output. However, if these values are only used in the hash functions in the Hiding algorithm then H is unnecessary.

The idea of using hash functions to integrate an IBE and a PKE scheme to construct a CL-PKE scheme was first demonstrated in the early version of this work [10]. An intuitive view on the security of this construction is that to recover message, one has to obtain the hash on both the DH value and the pairing-DH value, which in turn requires one to know both values. While, a Type-I adversary cannot compute the pairing-DH value if the underlying IBE scheme is secure and a Type-II adversary cannot compute the DH value if the underlying PKE schem is secure. Similarly, one can construct CL-KEM with the same approach.

Note that this approach is only based on the heuristic observation. A CL-PKE scheme constructed with this approach might not be secure in the model defined in Section 2.2. On the other hand, following this approach of using hash function on the (pairing) DH values to tightly integrate an IBE and a PKE scheme, we indeed are able to construct highly efficient and secure CL-PKE. In the following part, we shall present two concrete CL-PKE schemes constructed in this way.

4 Two Concrete CL-PKE Schemes

Two CL-PKE schemes, which we call SLOS-CL-PKE [28] and LQ-CL-PKE [25] based on SK-IBE [11] were presented. These schemes rely on a stronger ℓ-BDHI [11] assumption than BDH. In this section, following the general approach in Section 3, we present other two CL-PKE constructions from a full-domain hash IBE and a commutative blinding IBE respectively whose security is based on the BDH assumption.

4.1 CL-PKE1

From the well-known BF-IBE [6], Al-Riyami and Paterson constructed two CL-PKE schemes: AP-CL-PKE1 [3] and AP-CL-PKE2 [4]. AP-CL-PKE1 is based on a stronger assumption than BDH and the CL.Encrypt algorithm of the scheme requires three pairing operations which are very costly. AP-CL-PKE2 improves upon the previous scheme with better performance. However, as noted before both schemes are insecure against certain attacks.

Here, we present a CL-PKE which using a hash function integrates BF-IBE with the ElGamal-like PKE enhanced with the Fujisaki-Okamoto conversion [17]. We note that Fujisaki and Okamoto proposed two generic conversions [17,18], both can transform an OW-CPA secure PKE to an IND-CCA2 secure PKE. In [25], Libert and Quisquater showed that with slight modification, the second Fujisaki-Okamoto conversion (FO-2) [18] can convert a CL-IND-CPA secure CL-PKE to a CL-IND-CCA2 secure scheme. Here we demonstrate a similar result on the first Fujisaki-Okamoto conversion (FO-1) [17]. A simpler version of the scheme which strictly follows the general approach in Section 3 was first shown in an early draft [10] of this work. For the ease of the security analyse, here we adopt the enhanced FO-1 conversion, which just introduces minor extra computation overhead and may be of independent interest.

Generic Construction. Let Π be a CL-PKE scheme with the encryption algorithm $\mathcal{E}$ and the decryption algorithm $\mathcal{D}$. Define a CL-PKE scheme $\overline{\Pi}$ with the encryption algorithm $\overline{\mathcal{E}}$ as

$$\langle C_1, C_2\rangle \leftarrow \overline{\mathcal{E}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, m;\ \sigma)$$

where

$$\langle C_1, C_2\rangle = \langle \mathcal{E}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, \sigma;\ G_1(m, \sigma, \mathtt{ID}_A, P_A)), m \oplus G_2(\sigma)\rangle$$

and the decryption algorithm $\overline{\mathcal{D}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, \langle C_1, C_2\rangle)$ as

- $\sigma \leftarrow \mathcal{D}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C_1)$
- $m = C_2 \oplus G_2(\sigma)$
- If $C_1 = \mathcal{E}(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, \sigma;\ G_1(m, \sigma, \mathtt{ID}_A, P_A))$, output m; otherwise output $\perp$.

and other algorithms essentially the same as Π.

$\overline{\Pi}$ shares the same parameters with Π except $\overline{M_{\mathtt{pk}}}$, the master public system parameters of $\overline{\Pi}$ which includes Π's master public system parameters $M_{\mathtt{pk}}$ and two extra hash functions G_1 and G_2 defined as follows

$$G_1 : \mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}}) \times \mathbb{M}_{\Pi}(M_{\mathtt{pk}}) \times \{0,1\}^* \times \mathbb{P}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}}) \rightarrow \mathbb{R}_{\Pi}(M_{\mathtt{pk}})$$
$$G_2 : \mathbb{M}_{\Pi}(M_{\mathtt{pk}}) \rightarrow \mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})$$

Theorem 1. *Suppose Π is a γ-uniform CL-PKE scheme against CL-OW-CPA attacks, then $\overline{\Pi}$ is a CL-IND-CCA2 scheme. More specifically, suppose that a Type-I (resp. Type-II) CL-IND-CCA2 adversary $\mathcal{A}$ has advantage $\epsilon(k)$ against $\overline{\Pi}$ with running time $t(k)$, making q_D decryption queries and $q_{G_1} < 2^{|\mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})|}$ and q_{G_2} random oracle queries on G_1 and G_2 respectively. Then there exists a Type-I (resp. Type-II) CL-OW-CPA adversary $\mathcal{B}$ with advantage*

$$\mathtt{Adv}_{\mathcal{B}}(k) \geq \frac{\epsilon(k)}{q_{G_1} + q_{G_2}}(1-\gamma)^{q_{G_1} q_D}$$

over Π in running time

$$t_{\mathcal{B}}(k) \leq t(k) + O(q_{G_1} t_{\mathcal{E}}),$$

where $t_{\mathcal{E}}$ is the cost of $\mathcal{E}$.

Proof: We show how to make use of Type-I (resp. Type-II) CL-IND-CCA2 adversary $\mathcal{A}$ against $\overline{\Pi}$ to construct a Type-I (resp. Type-II) CL-OW-CPA adversary $\mathcal{B}$ again Π. The challenger $\mathcal{T}$ starts a Type-I (resp. Type-II) CL-OW-CPA game by passing $\mathcal{B}$ the master public key $M_{\mathtt{pk}}$ and providing with the oracle access including the possible random oracles, *Public-Key-Broadcast*, *Public-Key-Replace*, *Secret-Value-Extract* and for Type-I $\mathcal{B}$ with *Partial-Private-Key-Extract* as well. In the Type-II game, $\mathcal{T}$ also gives $M_{\mathtt{sk}}$ to $\mathcal{B}$.

$\mathcal{B}$ forwards $M_{\mathtt{pk}}$ with G_1 and G_2 as the master public key $\overline{M_{\mathtt{pk}}}$ of $\overline{\Pi}$ to $\mathcal{A}$ where G_1 and G_2 are two random oracles controlled by $\mathcal{B}$. In the Type-II game, $\mathcal{B}$ also passes $M_{\mathtt{sk}}$ to $\mathcal{A}$. $\mathcal{B}$ provides $\mathcal{A}$ with the oracle access as follows, and for simplicity of presentation we assume that $\mathcal{A}$ will abide by the rules defined in the models in Section 2.2.

- $\mathcal{B}$ forwards to its CL-OW-CPA challenger $\mathcal{T}$ the queries on oracles including *Public-Key-Broadcast*, *Secret-Value-Extract*, *Public-Key-Replace*, and for Type-I $\mathcal{T}$ *Partial-Private-Key-Extract* as well, and relays the answers from the challenger to $\mathcal{A}$.
- $\mathcal{B}$ forwards to its challenger $\mathcal{T}$ the queries on the possible random oracles provided by the challenger and relays the answers to $\mathcal{A}$.
- $G_1(m_i, \sigma_i, \mathtt{ID}_i, P_i)$**:** To respond to these queries $\mathcal{B}$ maintains a list G_1^{list}. Each entry in the list is a tuple of the form $(m_i, \sigma_i, \mathtt{ID}_i, P_i, r_i, C_1^i, C_2^i)$ indexed by $(m_i, \sigma_i, \mathtt{ID}_i, P_i)$. To respond to a query, $\mathcal{B}$ does the following operations:
  - If on G_1^{list} a tuple indexed by $(m_i, \sigma_i, \mathtt{ID}_i, P_i)$ exits, then $\mathcal{B}$ responds with the corresponding r_i.
  - Otherwise,
    - $\mathcal{B}$ randomly chooses a string $r_i \in \mathbb{R}_{\Pi}(M_{\mathtt{pk}})$.
    - $\mathcal{B}$ computes $C_1^i = \mathcal{E}(M_{\mathtt{pk}}, \mathtt{ID}_i, P_i, \sigma_i;\ r_i)$ and $C_2^i = \mathbb{G}_2(\sigma_i) \oplus m_i$.

  - $\mathcal{B}$ inserts a new tuple $(m_i, \sigma_i, \mathtt{ID}_i, P_i, r_i, C_1^i, C_2^i)$ into the list and responds to $\mathcal{A}$ with r_i.

– $G_2(\sigma_i)$**:** To respond to these queries $\mathcal{B}$ maintains a list G_2^{list}. Each entry in the list is a tuple of the form (σ_i, h_i) indexed by σ_i. To respond to a query, $\mathcal{B}$ does the following operations:
  - If on the list there is a tuple indexed by σ_i, then $\mathcal{B}$ responds with the corresponding h_i.
  - Otherwise, $\mathcal{B}$ randomly chooses a string $h_i \in \mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})$ and inserts a new tuple (σ_i, h_i) into the list. It responds to $\mathcal{A}$ with h_i.

– **Decrypt**$^S(\mathtt{ID}_i, C_i)$**:** $\mathcal{B}$ takes the following steps to respond to the query:
  - $\mathcal{B}$ queries its challenger the current public key P_i associated with $\mathtt{ID}_i$ by issuing *Public-Key-Broadcast*($\mathtt{ID}_i$).
  - $\mathcal{B}$ parses C_i as $\langle C_1^i, C_2^i \rangle$ and searches G_1^{list} to find tuples $(*, *, \mathtt{ID}_i, P_i, *, C_1^i, C_2^i)$.
  - If no such tuple is found, then $\mathcal{B}$ outputs $\perp$.
  - If more than one tuple is found, then $\mathcal{B}$ outputs $\perp$.
  - $\mathcal{B}$ outputs m_i in the only found tuple.

– **Challenge:** Once $\mathcal{A}$ decides that Phase 1 is over, it outputs identity $\mathtt{ID}^*$ and two messages m_0, m_1 on which it wishes to be challenged.
  - $\mathcal{B}$ queries $\mathcal{T}$ with the *Public-Key-Broadcast*($\mathtt{ID}^*$) to get the current public key P^* associated with $\mathtt{ID}^*$.
  - $\mathcal{B}$ forwards $\mathtt{ID}^*$ as the challenge ID to $\mathcal{T}$ and gets the challenge ciphertext as C_1^*.
  - $\mathcal{B}$ randomly samples $C_2^* \in \mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})$ and replies $\mathcal{A}$ with $C^* = \langle C_1^*, C_2^* \rangle$ as the challenge ciphertext in the CL-IND-CCA2 game.

– **Guess:** Once $\mathcal{A}$ outputs its guess b'. $\mathcal{B}$ randomly chooses a σ from G_1^{list} or G_2^{list} and outputs σ as the answer of the CL-OW-CPA game.

Now we analyse $\mathcal{B}$'s probability of outputting the correct response σ to $\mathcal{T}$. We define two events.

– Event 1, denoted by $\mathcal{H}_1$, is that in the game $\mathcal{A}$ queries $G_1(*, \mathcal{D}(M_{\mathtt{pk}}, \mathtt{ID}^*, P^*, S^*, C_1^*), \mathtt{ID}^*, P^*)$ or $G_2(\mathcal{D}(M_{\mathtt{pk}}, \mathtt{ID}^*, P^*, S^*, C_1^*))$ where S^* is the private key associated with $\mathtt{ID}^*$ when the challenge is issued.
– Event 2, denoted by $\mathcal{H}_2$, is that in the game $\mathcal{A}$ differentiates $\mathcal{B}$ from a real world before Event 1 happens.

Now we look at the possibility that $C^* = \langle C_1^*, C_2^* \rangle$ is a valid ciphertext of m_b for $b \in \{0, 1\}$. For C^* to be a valid challenge ciphertext, it is required that

$$\mathcal{D}(M_{\mathtt{pk}}, \mathtt{ID}^*, P^*, S^*, C_1^*) = \sigma'$$

$$G_2(\sigma') \oplus C_2^* = m_b$$

and

$$\mathcal{E}(M_{\mathtt{pk}}, \mathtt{ID}^*, P^*, \sigma';\ G_1(m_b, \sigma', \mathtt{ID}^*, P^*)) = C_1^*.$$

As C_2^* is randomly sampled from $\mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})$, C_2^* is valid for m_0 or m_1 with equal probability $\frac{1}{|\mathbb{M}_{\overline{\Pi}}(\overline{M_{\mathtt{pk}}})|}$. And as σ' is randomly sampled by $\mathcal{T}$ and G_1 is a

random oracle, C_1^* is valid for m_0 or m_1 with equal probability as well. Hence, given C^*, $\mathcal{A}$ either finds that C^* is not a valid ciphertext for either m_0 or m_1, or (C^* is a valid ciphertext for either m_0 or m_1 with equal probability) to win the game, outputs b if C^* is a valid ciphertext for m_b.

Here we conceptually force the adversary $\mathcal{A}$ to immediately output a random $b' \in \{0,1\}$ if it finds that C^* is an invalid challenge ciphertext. This change does not affect $\mathcal{A}$'s chance of winning the game. As G_1 and G_2 are random oracles,

$$\Pr[\mathcal{A} \text{ win} \mid \overline{\mathcal{H}_1}] = 1/2.$$

Then we have that

$$\begin{aligned}
\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins} | \mathcal{H}_1] \Pr[\mathcal{H}_1] + \Pr[\mathcal{A} \text{ wins} | \overline{\mathcal{H}_1}] \Pr[\overline{\mathcal{H}_1}] \\
&\leq \Pr[\mathcal{H}_1] + \frac{1}{2}(1 - \Pr[\mathcal{H}_1]) = \frac{1}{2} + \frac{1}{2}\Pr[\mathcal{H}_1]. \\
\Pr[\mathcal{A} \text{ wins}] &\geq \Pr[\mathcal{A} \text{ wins} | \overline{\mathcal{H}_1}] \Pr[\overline{\mathcal{H}_1}] \\
&= \frac{1}{2}(1 - \Pr[\mathcal{H}_1]) = \frac{1}{2} - \frac{1}{2}\Pr[\mathcal{H}_1].
\end{aligned}$$

So we have $\Pr[\mathcal{H}_1] \geq \epsilon(k)$. Now we estimate the probability of Event 2. In the game, $\mathcal{A}$ will notice the difference between the simulation and the real world only if $\mathcal{B}$ rejects a valid $Decrypt^S$ query or $\mathcal{A}$ finds that C^* is an invalid challenge ciphertext. As argued above, the latter event happens only if Event 1 occurs. So we only investigate the rejection of valid decryption query which occurs when

- Case 1. $\mathcal{A}$ queries $Decrypt^S(\mathtt{ID}_i, \langle C_1^i, C_2^i\rangle)$ such that $C_1^i = \mathcal{E}(M_{\mathtt{pk}}, \mathtt{ID}_i, P_i, \sigma_i; G_1(m_i, \sigma_i, \mathtt{ID}_i, P_i))$ and $C_2^i = G_2(\sigma_i) \oplus m_i$ without querying $G_1(m_i, \sigma_i, \mathtt{ID}_i, P_i)$ where P_i is the public key currently associated with $\mathtt{ID}_i$, or
- Case 2. $\mathcal{A}$ queries $Decrypt^S(\mathtt{ID}_i, \langle C_1^i, C_2^i\rangle)$ such that there are at least two tuples $(m_a, \sigma_a, \mathtt{ID}_i, P_i, r_a, C_1^i, C_2^i)$ and $(m_b, \sigma_b, \mathtt{ID}_i, P_i, r_b, C_1^i, C_2^i)$ in G_1^{list}. First this case cannot happen if $\sigma_a = \sigma_b$; otherwise $m_a = G_2(\sigma_a) \oplus C_2^i = G_2(\sigma_b) \oplus C_2^i = m_b$, and $(m_a, \sigma_a, \mathtt{ID}_i, P_i)$ uniquely defines a tuple in G_1^{list}. Hence Case 2 happens only if $\sigma_a \neq \sigma_b$ which implies $C_1^i = \mathcal{E}(M_{\mathtt{pk}}, \mathtt{ID}_i, P_i, \sigma_a;\ r_a) = \mathcal{E}(M_{\mathtt{pk}}, \mathtt{ID}_i, P_i, \sigma_b;\ r_b)$.

Case 1 happens with probability at most γ because $\mathcal{E}$ is γ-uniform and G_1 is truly random so $G_1(m_i, \sigma_i, \mathtt{ID}_i, P_i)$ is valid for $(M_{\mathtt{pk}}, \mathtt{ID}_i, P_i, \sigma_i, C_1^i)$ with probability at most γ.

Now we consider the probability of Case 2. Because $\mathcal{E}$ is γ-uniform and G_1 is truly random so one query $G_1(m_b, \sigma_b, \mathtt{ID}_i, P_i)$ is valid for $(M_{\mathtt{pk}}, \mathtt{ID}_i, P_i, \sigma_b, C_1^i)$ with probability at most γ where C_1^i is determined by $C_1^i = \mathcal{E}(M_{\mathtt{pk}}, \mathtt{ID}_i, P_i, \sigma_a;\ r_a)$. Note that for a fixed C_2^i there are $2^{|\mathbb{M}\overline{\pi}(\overline{M_{\mathtt{pk}}})|}$ pairs of (m_b, σ_b). For q_{G_1} queries ($q_{G_1} < 2^{|\mathbb{M}\overline{\pi}(\overline{M_{\mathtt{pk}}})|}$), Case 2 happens with probability at most $1 - (1-\gamma)^{q_{G_1}}$.

And if Case 1 happens, Case 2 won't happen. It comes that

$$\Pr[\overline{\mathcal{H}_2}] \geq (1 - \max\{\gamma, 1 - (1-\gamma)^{q_{G_1}}\})^{q_D} = ((1-\gamma)^{q_{G_1}})^{q_D}$$

Overall, we have that

$$\begin{aligned}\mathtt{Adv}_{\mathcal{B}}(k) &\geq \tfrac{1}{q_{G_1}+q_{G_2}} \Pr[\overline{\mathcal{H}_2}] \Pr[\mathcal{H}_1] \\ &\geq \tfrac{1}{q_{G_1}+q_{G_2}} (1-\gamma)^{q_{G_1} q_D} \cdot \epsilon(k)\end{aligned}$$

The Scheme. CL-PKE1 consists of following algorithms:
CL.Gen(1^k). Given a security parameter k, the algorithm works as follows.

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ of prime order p, an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_t$. Pick a random generator $P_2 \in \mathbb{G}_2$ and set $P_1 = \psi(P_2)$.
2. Pick a random $s \in \mathbb{Z}_p$ and compute $P_{pub} = sP_1$.
3. Pick four cryptographic hash functions:

$$\begin{aligned}&H_1 : \{0,1\}^* \rightarrow \mathbb{G}_2, \\ &H_2 : \mathbb{G}_t \times \mathbb{G}_1 \rightarrow \{0,1\}^n, \\ &H_3 : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p, \\ &H_4 : \{0,1\}^n \rightarrow \{0,1\}^n,\end{aligned}$$

   for some integer $n > 0$.
4. Output the master public key $M_{\mathtt{p\ell}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, P_{pub}, H_1, H_2, H_3, H_4)$ and the master secret key $M_{\mathtt{s\ell}} = s$.

The message space is $\mathbb{M} = \{0,1\}^n$, the ciphertext space is $\mathbb{C} = \mathbb{G}_1 \times \{0,1\}^n \times \{0,1\}^n$ and the randomness space is $\mathbb{R} = \{0,1\}^n$.
CL.PartialKey$(M_{\mathtt{s\ell}}, M_{\mathtt{p\ell}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathtt{p\ell}}$ and $M_{\mathtt{s\ell}}$, the algorithm computes $Q_A = H_1(\mathtt{ID}_A) \in \mathbb{G}_2$, $D_A = sQ_A$ and returns D_A.
CL.SecretVal$(M_{\mathtt{p\ell}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A$ and $M_{\mathtt{p\ell}}$, the algorithm outputs a random $X_A \in \mathbb{Z}_p$.
CL.PrivateKey$(M_{\mathtt{p\ell}}, D_A, X_A)$. Given $M_{\mathtt{p\ell}}$, D_A and X_A, the algorithm outputs $S_A = (D_A, X_A)$.
CL.PublicKey$(M_{\mathtt{p\ell}}, X_A, \mathtt{ID}_A)$. Given $M_{\mathtt{p\ell}}$ and X_A, the algorithm outputs $P_A = X_A P_1$.
CL.Encrypt$(M_{\mathtt{p\ell}}, \mathtt{ID}_A, P_A, m)$. Given a plaintext $m \in \{0,1\}^n$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathtt{p\ell}}$ and the public key P_A of the entity, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, m, \mathtt{ID}_A, P_A)$.
2. Compute $Q_A = H_1(\mathtt{ID}_A)$, $\xi = \hat{e}(P_{pub}, Q_A)^r$ and $f = rP_A$.
3. Set the ciphertext to $C = \langle rP_1, \sigma \oplus H_2(\xi, f), m \oplus H_4(\sigma) \rangle$.

CL.Decrypt$(M_{\mathtt{p\ell}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle U, V, W \rangle \in \mathbb{C}$, the private key $S_A = (D_A, X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathtt{p\ell}}$, the algorithm takes the following steps:

1. Compute $\xi' = \hat{e}(U, D_A)$, $f' = X_A U$ and $\sigma' = V \oplus H_2(\xi', f')$.
2. Compute $m' = W \oplus H_4(\sigma')$ and $r' = H_3(\sigma', m', \mathtt{ID}_A, P_A)$.
3. If $U \neq r'P_1$, output $\perp$, else return m' as the plaintext.

To prove the security of CL-PKE1, we shall first prove that the following Basic-CL-PKE1 is CL-OW-CPA secure, then apply Theorem 1 to obtain the security result.

Basic-CL-PKE1 shares most of the algorithms with CL-PKE1 except the following two (the hash functions H_3 and H_4 in **CL.Gen** of CL-PKE1 are unnecessary in Basci-CL-PKE1).

CL.Encrypt$(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, m;\ r)$. Given a plaintext $m \in \{0,1\}^n$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathtt{pk}}$ and the public key P_A of the entity, the following steps are performed.

1. Compute $Q_A = H_1(\mathtt{ID}_A)$, $\xi = \hat{e}(P_{pub}, Q_A)^r$ and $f = rP_A$.
2. Set the ciphertext to $C = \langle rP_1, m \oplus H_2(\xi, f)\rangle$.

CL.Decrypt$(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle U, V\rangle$, the private key $S_A = (D_A, X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathtt{pk}}$, the algorithm takes the following steps:

1. Compute $\xi' = \hat{e}(U, D_A)$ and $f' = X_A U$.
2. Return $m' = V \oplus H_2(\xi', f')$ as the plaintext.

Lemma 1. *Basic-CL-PKE1 is secure in the sense of CL-OW-CPA against Type-I adversaries provided that H_1 and H_2 are modeled as random oracles and the $BDH_{2,2,2}$ assumption is sound. Specifically, assume there exists a Type-I adversary $\mathcal{A}$ breaks Basic-CL-PKE1 with CL-OW-CPA attacks with advantage $\epsilon(k)$, and in the attack $\mathcal{A}$ runs in time $t(k)$ and makes q_{H_1} and q_{H_2} queries on H_1 and H_2 respectively. Then there exists an algorithm $\mathcal{B}$ to solve the $BDH_{2,2,2}$ problem with advantage and time as follows:*

$$\begin{aligned} \mathtt{Adv}_{\mathcal{B}}^{\mathtt{BDH}_{2,2,2}} &\geq (1 - \tfrac{q_{H_2}}{2^n})\epsilon(k)/q_{H_1}, \\ t_{\mathcal{B}}(k) &\leq t(k) + O(q_{H_2} \cdot \tau), \end{aligned}$$

where τ is the time of a pairing.

The proof is given in Appendix A.1.

Lemma 2. *Basic-CL-PKE1 is secure in the sense of CL-OW-CPA against Type-II adversaries provided that H_2 is modeled as random oracle and the $DH_{1,2,1}$ assumption is sound. Specifically, assume there exists a Type-II adversary $\mathcal{A}$ breaks Basic-CL-PKE1 with CL-OW-CPA attacks with advantage $\epsilon(k)$, and in the attack $\mathcal{A}$ runs in time $t(k)$ and gets q_P entity public keys. Then there exists an algorithm $\mathcal{B}$ to solve the $DH_{1,2,1}$ problem with advantage and time as follows:*

$$\begin{aligned} \mathtt{Adv}_{\mathcal{B}}^{\mathtt{DH}_{1,2,1}} &\geq \epsilon(k)/q_P, \\ t_{\mathcal{B}}(k) &\leq t(k) + O(q_{H_2} \cdot \tau), \end{aligned}$$

where τ is the time of a pairing.

The proof strategy is similar to Lemma 1. Due to lack of space, the details are omitted.

Following from Theorem 1 and Lemma 1 and 2, we have the following security result of CL-PKE1. Note that Basic-CL-PKE1 is $\frac{1}{2^n}$-uniform.

Theorem 2. *CL-PKE1 is secure against Type-I adversary with CL-IND-CCA2 attacks provided H_i $(1 \le i \le 4)$ are modeled as random oracles and the $BDH_{2,2,2}$ assumption is sound. CL-PKE1 is secure against Type-II adversary with CL-IND-CCA2 attacks provided H_i $(2 \le i \le 4)$ are modeled as random oracles and the $DH_{1,2,1}$ assumption is sound.*

Specifically, assume a Type-I adversary $\mathcal{A}_I$ breaks CL-PKE1 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$ and in the attack $\mathcal{A}_I$ makes q_D decryption queries and q_i queries on H_i for $1 \le i \le 4$ and $q_3 < 2^n$, then there exists an algorithm $\mathcal{B}_I$ to solve the $BDH_{2,2,2}$ problem with following advantage and time

$$\mathtt{Adv}^{\mathtt{BDH}_{2,2,2}}_{\mathcal{B}_I}(k) \ge \frac{(1-q_2/2^n)\epsilon(k)}{q_1(q_3+q_4)}(1-\tfrac{1}{2^n})^{q_3 q_D},$$
$$t_{\mathcal{B}_I}(k) \le t(k) + O(q_3 t_{\mathcal{E}} + q_2\tau),$$

where $t_{\mathcal{E}}$ is the cost of Basic-CL-PKE1 and τ is the time of a pairing.

Assume a Type-II adversary $\mathcal{A}_{II}$ breaks CL-PKE1 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$ and in the attack $\mathcal{A}_{II}$ makes q_P public key queries and q_i queries on H_i for $2 \le i \le 4$, then there exists an algorithm $\mathcal{B}_{II}$ to solve the $DH_{1,2,1}$ problem with following advantage and time

$$\mathtt{Adv}^{\mathtt{DH}_{1,2,1}}_{\mathcal{B}_{II}}(k) \ge \frac{\epsilon(k)}{q_P(q_3+q_4)}(1-\tfrac{1}{2^n})^{q_3 q_D},$$
$$t_{\mathcal{B}_{II}}(k) \le t(k) + O(q_3 t_{\mathcal{E}} + q_2\tau).$$

Following the general approach in Section 3, we can construct a CL-KEM: CL-KEM1 (see Appendix B in [10] for details).

4.2 CL-PKE2

In this subsection, we construct a CL-PKE (referred to as CL-PKE2) directly following the heuristic approach in Section 3. The scheme is based on BB_1-IBE [5], a commutative blinding IBE. CL-PKE2 consists of following algorithms:

CL.Gen(1^k): Given a security parameter k, the algorithm works as follows:

1. Generate three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ of prime order p and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$. Pick random generator $P_2 \in \mathbb{G}_2$ and $P_1 \in \mathbb{G}_1$.
2. Randomly sample a, b and $c \in \mathbb{Z}_p$. Set $Q_1 = aP_1, Q_2 = bP_1, Q_3 = cP_1 \in \mathbb{G}_1$, and $\hat{Q}_1 = aP_2, \hat{Q}_2 = bP_2, \hat{Q}_3 = cP_2 \in \mathbb{G}_2$. Compute $v_0 = \hat{e}(Q_1, \hat{Q}_2) = \hat{e}(P_1, P_2)^{ab}$.
3. Pick three cryptographic hash functions:

$$H_1 : \{0,1\}^* \to \mathbb{Z}_p,$$
$$H_2 : \mathbb{G}_t \times \mathbb{G}_1 \to \{0,1\}^n,$$
$$H_3 : \mathbb{G}_t \times \mathbb{G}_1 \times \{0,1\}^n \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_p,$$

   for some integer $n > 0$.
4. Output the master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, n, P_1, Q_1, Q_3, v_0, H_1, H_2, H_3)$ and the master secret key $M_{\mathfrak{sk}} = (P_2, a, b, c)$.

The message space is $\mathbb{M} = \{0,1\}^n$, the ciphertext space is $\mathbb{C} = \{0,1\}^n \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{Z}_p$ and the randomness space is $\mathbb{R} = \mathbb{Z}_p$.

CL.PartialKey$(M_{\mathfrak{sk}}, M_{\mathfrak{pk}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathfrak{pk}}$ and $M_{\mathfrak{sk}}$, the algorithm randomly picks $t \in \mathbb{Z}_p$ and outputs

$$D_A = (D_1, D_2) = ((ab + (aH_1(\mathtt{ID}_A) + c)t)P_2, tP_2).$$

We note that P_2 in $M_{\mathfrak{sk}}$ can be disclosed, in particular, on Type-1 pairings $P_1 = P_2$. Given $M_{\mathfrak{pk}}$ with P_2, one can verify if D_A is a signature on $\mathtt{ID}_A$.

CL.SecretVal$(M_{\mathfrak{pk}}, \mathtt{ID}_A)$. Given a string $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm returns a random $X_A \in \mathbb{Z}_p$.

CL.PrivateKey$(M_{\mathfrak{pk}}, D_A, X_A)$. Given $M_{\mathfrak{pk}}$, D_A and X_A, the algorithm outputs $S_A = (D_A, X_A)$.

CL.PublicKey$(M_{\mathfrak{pk}}, X_A, \mathtt{ID}_A)$. Given $M_{\mathfrak{pk}}$ and X_A, the algorithm outputs $P_A = X_A P_1$.

CL.Encrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, m)$. Given a plaintext $m \in \mathbb{M}$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathfrak{pk}}$ and the public key P_A of the entity, the following steps are performed.

1. Pick a random $r \in \mathbb{Z}_p$ and compute $C_1 = rP_1$ and $C_2 = rQ_3 + rH_1(\mathtt{ID}_A)Q_1$.
2. Compute $\xi = v_0^r$, $f = rP_A$ and $C_0 = m \oplus H_2(\xi, f)$.
3. Compute $\sigma = r + H_3(\xi, f, C_0, C_1, C_2) \mod p$.
4. Set the ciphertext to $C = \langle C_0, C_1, C_2, \sigma \rangle$.

CL.Decrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle C_0, C_1, C_2, \sigma \rangle \in \mathbb{C}$, the private key $S_A = (D_A = (D_1, D_2), X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm takes the following steps:

1. Compute $\xi' = \frac{\hat{e}(C_1, D_1)}{\hat{e}(C_2, D_2)}$ and $f' = X_A C_1$.
2. Compute $r' = \sigma - H_3(\xi', f', C_0, C_1, C_2) \mod p$, output $\perp$ if $(\xi', C_1) \neq (v_0^{r'}, r'P_1)$.
3. Compute $m' = C_0 \oplus H_2(\xi', f')$ and return m' as the plaintext.

CL-PKE2's security is summarised by the following theorem.

Theorem 3. *CL-PKE2 is secure against Type-I adversary with CL-IND-CCA2 attacks provided H_i $(1 \leq i \leq 3)$ are modeled as random oracles and the general BDH assumption is sound. CL-PKE2 is secure against Type-II adversary with CL-IND-CCA2 attacks provided H_i $(i = 2, 3)$ are modeled as random oracles and the $DH_{1,1,1}$ assumption is sound.*

Specifically, assume a Type-I adversary $\mathcal{A}_I$ breaks CL-PKE2 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$ and in the attack $\mathcal{A}_I$ makes q_D decryption queries and q_i queries on H_i for $1 \leq i \leq 3$, then there exists an algorithm $\mathcal{B}_I$ to solve the general BDH problem with following advantage and time

$$\begin{aligned} \mathtt{Adv}_{\mathcal{B}_I}^{\mathtt{Genenral\ BDH}}(k) &\geq \tfrac{(1-q_1/p)^{q_1}\epsilon(k)}{q_1(q_2+q_3)}(1-\tfrac{1}{p})^{q_D} \\ t_{\mathcal{B}_I}(k) &\leq t(k) + O(q_1 t_2 + q_D(t_1 + t_3)), \end{aligned}$$

where t_i for $i = 1, 2, 3$ is the cost of operation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ respectively.

Assume a Type-II adversary $\mathcal{A}_{II}$ breaks CL-PKE2 with CL-IND-CCA2 attack with advantage $\epsilon(k)$ in time $t(k)$, and in the attack $\mathcal{A}_{II}$ makes q_P public key queries and q_i queries on H_i for $i = 2, 3$, then there exists an algorithm $\mathcal{B}_{II}$ to solve the $DH_{1,1,1}$ problem with following advantage and time

$$\mathtt{Adv}^{\mathtt{DH}_{1,1,1}}_{\mathcal{B}_{II}}(k) \geq \frac{\epsilon(k)}{q_P(q_2+q_3)}(1-\frac{1}{p})^{q_D},$$
$$t_{\mathcal{B}_{II}}(k) \leq t(k) + O(q_D t_1).$$

The proof is given in Appendix A.3.

Similarly, by following the general approach in Section 3, we can construct another CL-KEM by simply outputting (C_1, C_2) as the encapsulation of secret key $H(\xi, f)$.

5 Efficiency Discussion and Comparison

We now assess the comparative efficiency of several concrete CL-PKE schemes. There have been some related security schemes in the literature. For example, in [8], the authors proposed a CL-PKE-like scheme without using pairing, which is more efficient than those schemes based on pairings. However, the scheme requires a user to execute CL.PartialKey first before generating the public key. This makes it more like a self-certified public key scheme [19] instead of CL-PKE. Moreover, the scheme unlike other CL-PKE proposals based on IBE schemes cannot work compatibly with exiting IBE, i.e., the scheme cannot degenerate smoothly to an IBE. Gentry proposed the security notion "Certificate-Based Encryption" (CBE) which is closely related with CL-PKE, and constructed a concrete CBE scheme [20]. Here we only consider schemes with proofs in a security formulation compatible with the one defined in Section 2.2.

Table 3. CL-PKE Efficiency Comparison

Schemes	Based IBE	IBE Type	Computation	Ciphertext Size
AP-CL-PKE1 [3](1)	BF-IBE	Full Domain Hash	=BF-IBE+$2P^{(2)}$	=BF-IBE
AP-CL-PKE2 [4](3)	BF-IBE	Full Domain Hash	=BF-IBE+$1M$	=BF-IBE
CL-PKE1	BF-IBE	Full Domain Hash	=BF-IBE+$1M$	=BF-IBE
CL-PKE2	BB_1-IBE	Commutative Blinding	=BB_1-IBE+$1M$	=BB_1-IBE
LQ-CL-PKE [25]	SK-IBE2	Exponent Inversion	=SK-IBE2+$1E$	=SK-IBE2
SLOS-CL-PKE [28]	SK-IBE	Exponent Inversion	=SK-IBE+$1M^{(4)}$	=SK-IBE

1. The scheme is enhanced by a verifiable random parameter generation.
2. The scheme requires two more pairings in the encryption algorithm than BF-IBE. The decryption operation is of the same cost as BF-IBE.
3. The scheme is enhanced as suggested in Section 2.2.
4. The scheme requires the Weil pairing.

As efficiency is the primary concern, we ignore any schemes designed in the standard model because their performance is far worse than those with random oracles. Table 3 summarises the used IBE scheme, the IBE type, computation and ciphertext size of several CL-PKE proposals. The computation cost

only counts the extra operations including Pairing, Multiplication in $\mathbb{G}_1$ and Exponentiation in $\mathbb{G}_t$ and the used IBE. Please refer to [5,9] for the detailed efficiency discussion of relevant IBE schemes.

Among the schemes using the full domain hash IBE schemes, CL-PKE1 and AP-CL-PKE2 have same performance and are faster than AP-CL-PKE1 in encryption. Among the schemes using the exponent inversion IBE, SLOS-CL-PKE makes use of the Weil pairing. As the Weil pairing is slower than the Tate pairing [23], SLOS-CL-PKE is slower than LQ-CL-PKE. Like BB_1-IBE, CL-PKE2 enjoys security reductions based on weak complexity assumptions and is efficient in encryption and relatively slow in decryption.

6 Conclusion

In this work, we revisited various of CL-PKE formulations and defined a strong security model for this type of encryption. Based on a simple observation on some existing IBE and PKE schemes, we proposed a heuristic approach of constructing efficient CL-PKE and following the approach, we constructed two efficient concrete CL-PKE schemes which are strongly secure in the random oracle model. Beside, we also demonstrated that a slightly modified Fujisaki-Okamoto conversion can transform a weak CL-PKE to a CL-IND-CCA2 secure scheme.

References

1. Al-Riyami, S.: Cryptographic schemes based on elliptic curve pairings. PhD thesis, Royal Holloway, University of London (2004)
2. Au, M.H., Chen, J., Liu, J.K., et al.: Malicious KGC attack in certificateless cryptography. Cryptology ePrint Archive, Report 2006/255
3. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
4. Al-Riyami, S.S., Paterson, K.G.: CBE from CL-PKE: a generic construction and efficient schemes. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 398–415. Springer, Heidelberg (2005)
5. Boyen, X.: The BB_1 identity-based cryptosystem: a standard for encryption and key encapsulation (August 2006) http://grouper.ieee.org/groups/1363/IBC/submissions/Boyen-bb1_ieee.pdf
6. 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)
7. Bentahar, K., Farshim, P., Malone-Lee, J., Smart, N.P.: Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058 (2005)
8. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005)
9. Cheng, Z.: Pairing-based cryptosystems and key agreement protocols. Thesis, Middlesex University (2007)

10. Cheng, Z., Comley, R.: Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012 (2005)
11. Chen, L., Cheng, Z.: Security proof of the Sakai-Kasahara's identity-based encryption scheme. In: Fukś, H., Lukosch, S., Salgado, A.C. (eds.) CRIWG 2005. LNCS, vol. 3706, pp. 442–459. Springer, Heidelberg (2005)
12. Chen, L., Cheng, Z., Malone-Lee, J., Smart, N.: An efficient ID-KEM based on the Sakai–Kasahara key construction. IEE Proc. Information Security 153(1), 19–26 (2006)
13. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33, 167–226 (2003)
14. Dent, A.: A Survey of certificateless encryption schemes and security models. Cryptology ePrint Archive, Report 2006/211 (2006)
15. Dent, A., Libert, B., Paterson, K.: Certificateless encryption schemes strongly secure in the standard model. Cryptology ePrint Archive, Report 2007/121 (2007)
16. ElGamal, T.: A public key cryptosystem and signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31, 469–472 (1985)
17. Fujisaki, E., Okamotom, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 535–554. Springer, Heidelberg (1999)
18. Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. IEICE Trans. Fund E83-9(1), 24–32 (2000)
19. Girault, M.: Self-certified public keys. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 490–497. Springer, Heidelberg (1992)
20. Gentry, C.: Certificate-based encryption and the certificate revocation problem. In: Biham, E. (ed.) Advances in Cryptology – EUROCRPYT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
21. Galindo, D.: Boneh-Franklin identity based encryption revisited. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 791–802. Springer, Heidelberg (2005)
22. Galindo, D., Morillo, P., Ráfols, C.: Breaking Yum and Lee generic constructions of certificate-less and certificate-based encryption schemes. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 81–91. Springer, Heidelberg (2006)
23. Granger, R., Page, D., Smart, N.P.: High security pairing-based cryptography revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. LNCS, vol. 4076, pp. 480–494. Springer, Heidelberg (2006)
24. Huang, Q., Wong, D.S.: Generic certificateless encryption in the standard model. Cryptology ePrint Archive, Report 2007/095
25. Libert, B., Quisquater, J.-J.: On constructing certificateless cryptosystems from identity based encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 474–490. Springer, Heidelberg (2006)
26. Kitagawa, T., Yang, P., Hanaoka, G., et al.: Generic transforms to acquire CCA-Security for identity based encryption: the cases of FOpkc and REACT. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 348–359. Springer, Heidelberg (2006)
27. Okamoto, T., Pointcheval, D.: REACT: rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159C-174. Springer, Heidelberg (2001)
28. Shi, Y., Li, J., Pan, J., Shi, J.: Efficient certificateless public key encryption with pairing. In: Proc. of Networks and Communication Systems 2006 (2006)

29. Yum, D.H., Lee, P.J.: Generic construction of certificateless encryption. In: Laganà, A., Gavrilova, M., Kumar, V., Mun, Y., Tan, C.J.K., Gervasi, O. (eds.) ICCSA 2004. Part I, LNCS, vol. 3043, pp. 802–811. Springer, Heidelberg (2004)
30. Yum, D.H., Lee, P.J.: Identity-based cryptography in public key management. In: Katsikas, S.K., Gritzalis, S., Lopez, J. (eds.) EuroPKI 2004. LNCS, vol. 3093, pp. 71–84. Springer, Heidelberg (2004)
31. Yang, P., Kitagawa, T., Hanaoka, G., et al.: Applying Fujisaki-Okamoto to identity-based encryption. In: Fossorier, M.P.C., Imai, H., Lin, S., Poli, A. (eds.) Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. LNCS, vol. 3857, pp. 183–192. Springer, Heidelberg (2006)
32. Zhang, Z., Feng, D.: On the security of a certificateless public-key encryption. Cryptology ePrint Archive, Report 2005/426

A Appendices

A.1 Proof of Lemma 1

Proof: Given a $\text{BDH}_{2,2,2}$ challenge (aP_2, bP_2, cP_2) with pairing parameters. Algorithm $\mathcal{B}$ simulates algorithm **CL.Gen** of Basic-CL-PKE1 to create the master public key $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, \psi, n, P_1, P_2, \psi(aP_2), H_1, H_2)$ and sets the master secret key $M_{\mathtt{sk}} = a$ which $\mathcal{B}$ does not know. H_1, H_2 are random oracles controlled by $\mathcal{B}$. Algorithm $\mathcal{B}$ passes $M_{\mathtt{pk}}$ to $\mathcal{A}$ and randomly selects $1 \leq I \leq q_{H_1}$ and responds to queries as follows (for simplicity, we assume that the adversary abides by the rules of the CL-OW-CPA game).

- $H_1(\mathtt{ID}_i)$**:** $\mathcal{B}$ maintains a list H_1^{list} of tuples $\langle \mathtt{ID}_j, Q_j, h_j \rangle$ as explained below. The list is initially empty. When $\mathcal{A}$ makes a query at a point $\mathtt{ID}_i$, $\mathcal{B}$ responds as follows:
  - If $\mathtt{ID}_i$ already appears on H_1^{list} in a tuple $\langle \mathtt{ID}_i, Q_i, h_i \rangle$, then $\mathcal{B}$ responds with Q_i.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $\langle ID_I, bP_2, \perp \rangle$ into the tuple list and responds with $H_1(\mathtt{ID}_I) = bP_2$.
  - Otherwise, $\mathcal{B}$ selects a random integer $h_i \in \mathbb{Z}_p$, computes $Q_i = h_i P_2$, and stores $\langle \mathtt{ID}_i, Q_i, h_i \rangle$ into the tuple list. $\mathcal{B}$ responds with $H_1(\mathtt{ID}_i) = Q_i$.
- **Public-Key-Broadcast(**$\mathtt{ID}_i$**):** $\mathcal{B}$ maintains a list P^{list} with tuples of $\langle \mathtt{ID}_i, U_i, X_i \rangle$ and responds to the queries as follows:
  - If a tuple exits for $\mathtt{ID}_i$, then $\mathcal{B}$ returns the corresponding U_i.
  - Otherwise, $\mathcal{B}$ randomly samples $X_i \in \mathbb{Z}_p$ and inserts a tuple $\langle \mathtt{ID}_i, X_i P_1, X_i \rangle$ into the list and returns $X_i P_1$.
- **Public-Key-Replace(**$\mathtt{ID}_i, P$**):** $\mathcal{B}$ replaces the tuple on P^{list} for $\mathtt{ID}_i$ with a new tuple $\langle \mathtt{ID}_i, P, \perp \rangle$.
- **Partial-Private-Key-Extract(**$\mathtt{ID}_i$**):** $\mathcal{B}$ first searches H_1^{list} for the tuple with $\mathtt{ID}_i$. If no such tuple is found, then $H_1(\mathtt{ID}_i)$ is queried. If in the found tuple $h_i = \perp$, then $\mathcal{B}$ aborts the game. Otherwise, $\mathcal{B}$ returns $h_i a P_2$.
- **Secret-Value-Extract(**$\mathtt{ID}_i$**):** $\mathcal{B}$ searches P^{list} for $\mathtt{ID}_i$, if no tuple is found, then *Public-Key-Broadcast*($\mathtt{ID}_i$) is queried first. $\mathcal{B}$ returns X_i found on P^{list} with $\mathtt{ID}_i$.

- $H_2(K_i, F_i)$**:** To respond to the query, $\mathcal{B}$ maintains a list H_2^{list} with tuples of the form $\langle K_i, F_i, \zeta_i \rangle$. On the query, $\mathcal{B}$ does the following operations:
  - If a tuple (K_i, F_i, ζ_i) is on the list, then $\mathcal{B}$ responds with ζ_i.
  - Otherwise, $\mathcal{B}$ randomly chooses $\zeta_i \in \{0,1\}^n$ and adds the tuple $\langle K_i, F_i, \zeta_i \rangle$ to the list. It responds to $\mathcal{A}$ with ζ_i.
- **Challenge:** Once $\mathcal{A}$ decides that Phase 1 of the game is over, it outputs $\mathtt{ID}^*$ on which it wishes to be challenged. $\mathcal{B}$ queries $H_1(\mathtt{ID}^*)$ and if h^* on H_1^{list} for $\mathtt{ID}^*$ is not $\perp$, then $\mathcal{B}$ aborts the game (for simplicity). Otherwise $\mathcal{B}$ randomly samples $C_2^* \in \{0,1\}^n$ and returns $C^* = (\psi(cP_2), C_2^*)$ as the challenge ciphertext. Note that the plaintext m^* of the C^* is

$$m^* = C_2^* \oplus H_2(\hat{e}(\psi(cP_2), abP_2), cP^*)$$

  where P^* is the currently public key on P^{list} for $\mathtt{ID}^*$.
- **Guess:** Once $\mathcal{A}$ decides Phase 2 of the game is over, it outputs m'. $\mathcal{B}$ does the following to respond to the BDH challenge.
  - $\mathcal{B}$ computes $\zeta^* = m' \oplus C_2^*$.
  - $\mathcal{B}$ goes through H_2^{list} to find tuples with $\langle K_i, F_i, \zeta^* \rangle$ with $\hat{e}(F_i, P_2) = \hat{e}(P^*, cP_2)$.
  - If no such tuple or more than one tuple is found, $\mathcal{B}$ fails the game.
  - $\mathcal{B}$ returns K_i from the found tuple to the BDH challange.

We define following events: Event 1, denoted by $\mathcal{H}_1$, is that $\mathcal{B}$ aborts the game prematurely. Event 2, denoted by $\mathcal{H}_2$, is that $H_2(\hat{e}(P_1, P_2)^{abc}, cP^*)$ is queried at some point during the simulation above. Event 3, denoted by $\mathcal{H}_3$, is that more than one tuple $\langle K_i, F_i, \zeta^* \rangle$ with $\hat{e}(F_i, P_2) = \hat{e}(P^*, cP_2)$ is found in H_2^{list}.

Through a standard argument, we have

$$\begin{aligned} &\Pr[\overline{\mathcal{H}_1}] \geq 1/q_{H_1} \\ &\Pr[\mathcal{H}_2] \geq \epsilon(k) \\ &\Pr[\mathcal{H}_3] \leq q_{H_2}/2^n \end{aligned}$$

Overall we have

$$\mathtt{Adv}_{\mathcal{B}}^{\mathtt{BDH}_{2,2,2}}(k) \geq \Pr[\overline{\mathcal{H}_1} \wedge \mathcal{H}_2 \wedge \overline{\mathcal{H}_3}] \approx (1 - q_{H_2}/2^n)\epsilon(k)/q_{H_1}.$$

A.2 AP-CL-PKE2 and Its Attacks

AP-CL-PKE2 shares most the algorithms with CL-PKE1 except CL.Encrypt and CL.Decrypt. And AP-CL-PKE2 uses the hash functions $H_1, H_2', H_2'', H_3', H_4$ where H_1, H_4 are just as of CL-PKE1 and H_2', H_2'', H_3' are defined as follows:

$$\begin{aligned} &H_2' : \mathbb{G}_t \times \rightarrow \{0,1\}^n, \\ &H_2'' : \mathbb{G}_1 \times \rightarrow \{0,1\}^n, \\ &H_3' : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_p, \end{aligned}$$

AP-CL-PKE2's encryption and decryption algorithm are as follows:

CL.Encrypt$(M_{\mathtt{pk}}, \mathtt{ID}_A, P_A, m)$. Given a plaintext $m \in \{0,1\}^n$, the identity $\mathtt{ID}_A$ of entity A, the system parameters $M_{\mathtt{pk}}$ and the public key P_A of the entity, the following steps are performed.

1. Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3'(\sigma, m)$.
2. Compute $Q_A = H_1(\mathtt{ID}_A)$, $\xi = \hat{e}(P_{pub}, Q_A)^r$ and $f = rP_A$.
3. Set the ciphertext to $C = \langle rP_1, \sigma \oplus H_2'(\xi) \oplus H_2''(f), m \oplus H_4(\sigma)\rangle$.

CL.Decrypt$(M_{\mathfrak{pk}}, \mathtt{ID}_A, P_A, S_A, C)$. Given a ciphertext $C = \langle U, V, W\rangle \in \mathbb{C}$, the private key $S_A = (D_A, X_A)$, the identifier $\mathtt{ID}_A$ and $M_{\mathfrak{pk}}$, the algorithm takes the following steps:

1. Compute $\xi' = \hat{e}(U, D_A)$, $f' = X_A U$ and $\sigma' = V \oplus H_2'(\xi') \oplus H_2''(f')$.
2. Compute $m' = W \oplus H_4(\sigma')$ and $r' = H_3'(\sigma', m')$.
3. If $U \neq r'P_1$, output $\perp$, else return m' as the plaintext.

Though AP-CL-PKE2 shares very similarity with CL-PKE1, the scheme suffers from two attacks: a Type-I^C attack and a Type-II attack.

An adversary $\mathcal{A}$ launches the Type-I^C attack as follows:

1. $\mathcal{A}$ randomly chooses $\mathtt{ID}^*$ and two messages m_0, m_1, and passes them to the challenger $\mathcal{T}$ to get the ciphertext $C^* = \langle U^*, V^*, W^*\rangle$ of m_b for $b \in \{0,1\}$.
2. $\mathcal{A}$ issues *Secret-Value-Extract*$(\mathtt{ID}^*)$ to get the secret value X^*.
3. $\mathcal{A}$ randomly chooses $X' \in \mathbb{Z}_p$ and issues *Public-Key-Replace*$(\mathtt{ID}^*, X'P_1)$.
4. $\mathcal{A}$ generates a ciphertext $C' = \langle U^*, V^* \oplus H_2''(X^*U^*) \oplus H_2''(X'U^*), W^*\rangle$. Note that C' is a valid ciphertext of m_b for $\mathtt{ID}^*$ with the public key $X'P_1$.
5. $\mathcal{A}$ issues *Decrypt*$^C(\mathtt{ID}^*, C', X')$ to get the plaintext m_b to win Game 1.

Note that in Al-Riyami-Paterson's CL-PKE formulation [4], the oracle *Secret-Value-Extract* does not exist. But the attack still works by $\mathcal{A}$ first choosing $X^* \in Z_p$ and replacing $\mathtt{ID}^*$'s public key with X^*P_1 before issuing the challenge [32]. It is easy to see merely adding the message recipient's identifier in H_3' does not defend the attack.

An adversary $\mathcal{A}$ can launch a trivial Type-II attack as follows:

1. $\mathcal{A}$ issues *Public-Key-Broadcast*$(\mathtt{ID}^*)$ to get the public key value P^*.
2. $\mathcal{A}$ randomly chooses two messages m_0, m_1, and passes $\mathtt{ID}^*$ and the messages to the challenger $\mathcal{T}$ to get the ciphertext $C^* = \langle U^*, V^*, W^*\rangle$ of m_b for $b \in \{0,1\}$.
3. $\mathcal{A}$ randomly chooses $\mathtt{ID}'$ and issues *Public-Key-Replace*$(\mathtt{ID}', P^*)$.
4. $\mathcal{A}$ generates a ciphertext $C' = \langle U^*, V^* \oplus H_2'(\xi') \oplus H_2'(\xi''), W^*\rangle$ with $\xi' = \hat{e}(U^*, D_{\mathtt{ID}^*})$ and $\xi'' = \hat{e}(U^*, D_{\mathtt{ID}'})$ where $D_{\mathtt{ID}^*}, D_{\mathtt{ID}'}$ are the partial key of $\mathtt{ID}^*, \mathtt{ID}'$ respectively. $\mathcal{A}$ can compute both values with the master secret key $M_{\mathfrak{sk}}$. Note that C' is a valid ciphertext of m_b for $\mathtt{ID}'$ with the public key P^*.
5. $\mathcal{A}$ issues *Decrypt*$^S(\mathtt{ID}', C', X')$ to get the plaintext m_b to win Game 2.

It is sample to find out that merely adding the message recipient's public key in H_3' does not prevent the attack.

By applying Theorem 1 it is straightforward to prove that including the message recipient's identifier and public key in H_3' as CL-PKE1 does, the modified scheme is secure.

A.3 Proof of Theorem 3

Proof: We first deal with the Type-I adversary $\mathcal{A}_I$. Given a BDH problem $(P_1, aP_1, rP_1, P_2, aP_2, bP_2)$, we can construct an algorithm $\mathcal{B}_I$ to compute $\hat{e}(P_1, P_2)^{abr}$ by making use of $\mathcal{A}_I$ as a subroutine. $\mathcal{B}_I$ randomly chooses $1 \leq I \leq q_1$ and starts to simulate the CL-IND-CCA2 Game 1 as follows:

- **CL.Gen:** $\mathcal{B}_I$ simulates **CL.Gen** as follows:
  - Set $Q_1 = aP_1, \hat{Q}_1 = aP_2, \hat{Q}_2 = bP_2$.
  - Randomly sample $u, h \in \mathbb{Z}_p$ and set $c = h - ua \mod p$ and compute $Q_3 = hP_1 - uaP_1 = cP_1$ and $\hat{Q}_3 = hP_2 - uaP_2 = cP_2$.
  - Compute $v_0 = \hat{e}(aP_1, bP_2) = \hat{e}(P_1, P_2)^{ab}$.
  - Output the master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, \hat{e}, n, P_1, Q_1, Q_3, v_0, H_1, H_2, H_3)$ where H_i are random oracles controlled by $\mathcal{B}_I$ and set the master secret key $M_{\mathfrak{sk}} = (P_2, a, b, c)$.
- $H_1(\mathtt{ID}_i)$**:** $\mathcal{B}_I$ maintains a list H_1^{list} of tuples $\langle \mathtt{ID}_j, u_j \rangle$ as explained below. The list is initially empty. When $\mathcal{A}_I$ makes a query on $\mathtt{ID}_i$, $\mathcal{B}_I$ responds as follows:
  - If $\mathtt{ID}_i$ is on H_1^{list} in a tuple $\langle \mathtt{ID}_i, u_i \rangle$, then $\mathcal{B}_I$ responds with u_i.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}_I$ stores $\langle \mathtt{ID}_I, u \rangle$ into the tuple list and responds with u.
  - Otherwise, $\mathcal{B}_I$ selects a random integer $u_i \in \mathbb{Z}_p$ and stores $\langle \mathtt{ID}_i, u_i \rangle$ into the tuple list. $\mathcal{B}_I$ responds with u_i.
- **Public-Key-Broadcast(**$\mathtt{ID}_i$**):** $\mathcal{B}_I$ maintains a list P^{list} with tuples of the form $\langle \mathtt{ID}_i, U_i, X_i \rangle$ and responds to the queries as follows:
  - If a tuple exits for $\mathtt{ID}_i$, then $\mathcal{B}_I$ returns the corresponding U_i.
  - Otherwise, $\mathcal{B}_I$ randomly samples $X_i \in \mathbb{Z}_p$ and inserts a tuple $\langle \mathtt{ID}_i, X_iP_1, X_i \rangle$ into P^{list} and returns X_iP_1.
- **Public-Key-Replace(**$\mathtt{ID}_i, P$**):** $\mathcal{B}_I$ replace the tuple on P^{list} for $\mathtt{ID}_i$ with a new tuple $\langle \mathtt{ID}_i, P, \perp \rangle$.
- **Partial-Private-Key-Extract(**$\mathtt{ID}_i$**):** $\mathcal{B}_I$ responds to the query as follows
  - $\mathcal{B}_I$ first searches H_1^{list} for the tuple with $\mathtt{ID}_i$. If no such tuple is found, then $H_1(\mathtt{ID}_i)$ is queried. Assume u_i is on the found tuple in H_1^{list} for $\mathtt{ID}_i$.
  - If $u_i = u$, $\mathcal{B}_I$ aborts the game.
  - $\mathcal{B}_I$ randomly samples $t_i \in \mathbb{Z}_p$ and sets $\tilde{t}_i = t_i - b/(u_i - u)$ and computes

$$\begin{aligned} D_1 &= \tfrac{-h}{u_i-u}\hat{Q}_2 + t_i((u_i-u)\hat{Q}_1 + hP_2) \\ &= \tfrac{-h}{u_i-u}bP_2 + \tfrac{b}{u_i-u}((u_i-u)\hat{Q}_1 + hP_2) + \tilde{t}_i((u_i-u)\hat{Q}_1 + hP_2) \\ &= abP_2 + \tilde{t}_i(u_iaP_2 + cP_2) = (ab + (aH_1(\mathtt{ID}_i) + c)\tilde{t}_i)P_2 \\ D_2 &= t_iP_2 - \tfrac{1}{u_i-u}bP_2 = \tilde{t}_iP_2 \end{aligned}$$

    $\mathcal{B}_I$ outputs (D_1, D_2) as the answer.
- **Secret-Value-Extract(**$\mathtt{ID}_i$**):** $\mathcal{B}_I$ searches P^{list} for $\mathtt{ID}_i$, and if no tuple is found, then *Public-Key-Broadcast*($\mathtt{ID}_i$) is queried first. $\mathcal{B}_I$ returns X_i found on P^{list} with $\mathtt{ID}_i$.

- $H_2(K_i, F_i)$**:** To respond to the query $\mathcal{B}_I$ maintains a list H_2^{list} with tuples of the form $\langle K_i, F_i, \zeta_i \rangle$. On the query, $\mathcal{B}_I$ does the following operations:
  - If a tuple (K_i, F_i, ζ_i) is on the list, then $\mathcal{B}_I$ responds with ζ_i.
  - Otherwise, $\mathcal{B}_I$ randomly chooses $\zeta_i \in \{0,1\}^n$ and adds the tuple $\langle K_i, F_i, \zeta_i \rangle$ to the list. It responds to $\mathcal{A}_I$ with ζ_i.
- $H_3(K_i, F_i, C_0^i, C_1^i, C_2^i)$**:** To respond to the query $\mathcal{B}_I$ maintains a list H_3^{list} with tuples of the form $\langle K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i \rangle$. On the query, $\mathcal{B}_I$ does the following operations:
  - If a tuple $(K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i)$ is on the list, then $\mathcal{B}_I$ responds with ξ_i.
  - Otherwise, $\mathcal{B}_I$ randomly chooses $\xi_i \in \mathbb{Z}_p$ and adds the tuple $\langle K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i \rangle$ to the list. It responds to $\mathcal{A}_I$ with ξ_i.
- **Decrypt**$^S(\mathtt{ID}_i, C_i)$**:** $\mathcal{B}_I$ takes the following steps to respond to the query
  - $\mathcal{B}_I$ queries the current public key P_i associated with $\mathtt{ID}_i$ by *Public-Key-Broadcast*($\mathtt{ID}_i$).
  - $\mathcal{B}_I$ parses C_i as $\langle C_0^i, C_1^i, C_2^i, \sigma_i \rangle$ and searches H_3^{list} to finds tuples $(K_i, F_i, C_0^i, C_1^i, C_2^i, \xi_i)$ and puts them into set S_0.
  - If S_0 is empty, then $\mathcal{B}_I$ outputs $\perp$.
  - For each tuple in S_0,
    - Compute $r_i = \sigma_i - \xi_i \mod p$.
    - If $C_1^i = r_i P_1, C_2^i = r_i Q_3 + r_i H_1(\mathtt{ID}_i) Q_1, K_i = v_0^{r_i}$ and $F_i = r_i P_i$, then return $C_0 \oplus H_2(K_i, F_i)$.
  - If no tuple is found to pass the above check, then $\mathcal{B}_I$ outputs $\perp$.
- **Challenge:** Once $\mathcal{A}_I$ decides that Phase 1 is over it outputs identity $\mathtt{ID}^*$ and two messages m_0, m_1 on which it wishes to be challenged.
  - If $H_1(\mathtt{ID}^*) \neq u$, then $\mathcal{B}_I$ aborts the game.
  - $\mathcal{B}_I$ randomly samples $\sigma^* \in Z_p$ and $C_0^* \in \{0,1\}^n$. $\mathcal{B}_I$ sets $C_1^* = rP_1$ and $C_2^* = rcP_1 + ruaP_1 = hrP_1$.
  - $\mathcal{B}_I$ returns $C^* = \langle C_0^*, C_1^*, C_2^*, \sigma^* \rangle$ as the challenge ciphertext.
- **Guess:** Once $\mathcal{A}_I$ outputs its guess b'. $\mathcal{B}_I$ randomly chooses a K_i from H_2^{list} or H_3^{list} and outputs K_i as the answer of the BDH problem.

Now we analyse $\mathcal{B}_I$'s probability of outputting the correct answer to the BDH problem. We define three events. Event 1, denoted by $\mathcal{H}_1$, is that $\mathcal{B}_I$ aborts the game prematurely which could happen when $\mathtt{ID}^* \neq \mathtt{ID}_I$ or $H_1(\mathtt{ID}_i) = u$ with $\mathtt{ID}_i \neq \mathtt{ID}_I$. Event 2, denoted by $\mathcal{H}_2$, is that $H_2(\hat{e}(P_1, P_2)^{abr}, *)$ or $H_3(\hat{e}(P_1, P_2)^{abr}, *, *, *, *)$ is queried at some point during the simulation above. Event 3, denoted by $\mathcal{H}_3$, is that in the game $\mathcal{A}_I$ differentiates $\mathcal{B}_I$ from a real world before Event 2 happens.

Through a standard argument, we have

$$\begin{aligned} \Pr[\overline{\mathcal{H}_1}] &\geq (1 - q_1/p)^{q_1}/q_1 \\ \Pr[\mathcal{H}_2] &\geq \epsilon(k) \\ \Pr[\overline{\mathcal{H}_3}] &\geq (1 - 1/p)^{q_D} \end{aligned}$$

Overall we have

$$\mathtt{Adv}_{\mathcal{B}_I}^{\mathtt{BDH}}(k) \geq \frac{1}{q_2 + q_3} \Pr[\overline{\mathcal{H}_1} \wedge \mathcal{H}_2 \wedge \overline{\mathcal{H}_3}] = \frac{(1 - q_1/p)^{q_1} \epsilon(k)}{q_1(q_2 + q_3)} (1 - 1/p)^{q_D}.$$

The Type-II security can be proved with similar strategy used in the Type-I security proof. Due to lack of space, the details are omitted.

Hyperelliptic Pairings

Steven D. Galbraith[1], Florian Hess[2], and Frederik Vercauteren[3,⋆]

[1] Mathematics Department,
Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, UK
steven.galbraith@rhul.ac.uk
[2] Technische Universität Berlin,
Fakultät II, Institut für Mathematik Sekr. MA 8-1,
Strasse des 17. Juni 136, D-10623 Berlin, Germany
hess@math.tu-berlin.de
[3] Department of Electrical Engineering,
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
frederik.vercauteren@esat.kuleuven.be

Abstract. We survey recent research on pairings on hyperelliptic curves and present a comparison of the performance characteristics of pairings on elliptic curves and hyperelliptic curves. Our analysis indicates that hyperelliptic curves are not more efficient than elliptic curves for general pairing applications.

1 Introduction

The original work on pairing-based cryptography [46,47,27,6] used pairings on elliptic curves. It was then natural to suggest using higher genus curves for pairing applications [18]. The motivation given in [18] for considering higher genus curves was to have a wider choice of possible embedding degrees k. This motivation was supported by [42,45] who showed that, for supersingular abelian varieties, one can always get larger security parameter in dimension 4 than using elliptic curves.

Koblitz [30] was the first to propose using divisor class groups of hyperelliptic curves for cryptosystems based on the discrete logarithm problem. Since that time there has been much research on comparing the speed of elliptic curves and hyperelliptic curves for cryptography. The current state of the art (see [7,34] for a survey) suggests that in some situations genus 2 curves can be faster than elliptic curves. This gives further motivation for considering pairings on hyperelliptic curves.

Duursma and Lee [11] were the first to give fast algorithms for computing pairings on curves of genus ≥ 2. Their loop shortening idea was generalised in [5,26,23]. Some other papers on hyperelliptic pairings are [10,12,35,40].

It is therefore natural to explore whether pairings on hyperelliptic curves can be competitive/faster than pairings on elliptic curves, or whether there are any

⋆ Postdoctoral Fellow of the Research Foundation - Flanders (FWO).

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 108–131, 2007.

other advantages. This paper surveys the current state of knowledge on pairings on curves. We discuss possible advantages and disadvantages of using curves of genus $g > 1$ and we present a number of open problems for future research.

The plan of the paper is as follows. We recall some background on hyperelliptic curves, divisor class groups, and representation of divisor classes. We discuss the supposed advantages of hyperelliptic vs. elliptic curves in standard cryptography (throughout the paper we use the phrase 'non-pairing cryptography' to denote the other applications of elliptic and hyperelliptic curves). We then recall the Tate-Lichtenbaum and ate pairings and present some implementation details for pairings on hyperelliptic curves, including a discussion of the critical computational task of evaluating a function at a divisor. We discuss the use of degenerate divisors for pairings, give some results on distortion maps for supersingular genus 2 curves and recall the Rubin-Silverberg point compression method. Finally, we give a thorough comparison of the performance characteristics of elliptic and hyperelliptic curves.

Our conclusion is that, for most applications, elliptic curves provide more efficient solutions than hyperelliptic curves. Nevertheless, there are many interesting open questions relating to pairings on hyperelliptic curves. In order to encourage research on this important topic we provide a list of problems for further study.

2 Background on Curves

A good reference is [1]. An affine elliptic curve E over a finite field $\mathbb{F}_q$ is given by an equation of the form

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_3, a_2, a_4, a_6 \in \mathbb{F}_q$ are such that E is non-singular. An elliptic curve $\mathcal{E}$ over $\mathbb{F}_q$ is the associated projective curve of an affine elliptic curve E. It is a non-singular projective curve of genus one and has, in addition to the points of E, one extra point (called the point at infinity and denoted ∞), which is $\mathbb{F}_q$-rational. The set of points $\mathcal{E}(\mathbb{F}_q)$ forms a group, where ∞ is the identity element and the group operation is given by the chord-and-tangent rule.

We define an affine hyperelliptic curve over $\mathbb{F}_q$ to be a non-singular affine curve of the form

$$C : y^2 + H(x)y = F(x)$$

where $H(x), F(x) \in \mathbb{F}_q[x]$. We denote by g the genus of C, in which case we may assume that $\deg(H(x)) \leq g + 1$ and $\deg(F(x)) \leq 2g + 2$. There is a single point at infinity on the associated projective curve $\mathcal{C}_0$, but it is now singular and there may be one $\mathbb{F}_q$-rational or two, not necessarily $\mathbb{F}_q$-rational points above this on the normalisation $\mathcal{C}$ (i.e., desingularisation) of $\mathcal{C}_0$. We call $\mathcal{C}$ a hyperelliptic curve over $\mathbb{F}_q$ of genus g. For any extension field K of $\mathbb{F}_q$ we denote by $\mathcal{C}(K)$ the set of points on $\mathcal{C}$ with coordinates in K.

2.1 The Divisor Class Group

The points of $\mathcal{C}$ no longer form a group, if $g \geq 2$. Instead, one works with the divisor class group of the curve.

The divisor group is defined as

$$\mathrm{Div}(\mathcal{C}) = \left\{ \sum_{P \in \mathcal{C}(\overline{\mathbb{F}}_q)} n_P(P) : n_P \in \mathbb{Z} \text{ and all but finitely many } n_P = 0 \right\},$$

where the sum is a formal sum over symbols (P), and addition is carried out coefficientwise. For $D \in \mathrm{Div}(\mathcal{C})$, define $\deg(D) = \sum_{P \in \mathcal{C}(\overline{\mathbb{F}}_q)} n_P \in \mathbb{Z}$ and $v_P(D) = n_P$, so we can write $D = \sum_{P \in \mathcal{C}(\overline{\mathbb{F}}_q)} v_P(D)P$. Let $\mathrm{Div}^0(\mathcal{C}) = \{D \in \mathrm{Div}(\mathcal{C}) : \deg(D) = 0\}$ which is a subgroup of $\mathrm{Div}(\mathcal{C})$. The support of a divisor D, denoted $\mathrm{supp}(D)$, is the set $\{P \in \mathcal{C}(\overline{\mathbb{F}}_q) : v_P(D) \neq 0\}$. A divisor D is called effective if $v_P(D) \geq 0$ for all P.

For any algebraic extension field K of $\mathbb{F}_q$ a K-rational function on $\mathcal{C}$ is a function $f : \mathcal{C}(\overline{\mathbb{F}}_q) \to \overline{\mathbb{F}}_q \cup \{\infty\}$ that can be represented by a fraction g/h of homogeneous polynomials of the same degree defined over K. This means that for all $P \in \mathcal{C}(\overline{\mathbb{F}}_q)$ we have either $f(P) = g(P)/h(P)$ (evaluation of g, h at the coordinates of P) or $g(P) = h(P) = 0$, and the latter happens for at most finitely many P. It can be shown that for $P \in C(\overline{\mathbb{F}}_q)$ with $g(P) = h(P) = 0$ one can choose an alternative representation $\tilde{g}/\tilde{h}$ of f such that $\tilde{g}(P) \neq 0$ or $\tilde{h}(P) \neq 0$, hence $f(P) = \tilde{g}(P)/\tilde{h}(P)$.

The K-rational functions form a field $K(\mathcal{C})$, which is called the function field of $\mathcal{C}$ over K. The function evaluation $f(P)$ for $f \in K(\mathcal{C})$ can be either zero, a non-zero value from $\overline{\mathbb{F}}_q$, or ∞. It is possible to associate a multiplicity $v_P(f) \in \mathbb{Z}$ of zero (or pole), which satisfies the expected properties known from Laurent series. Equivalently, the function v_P is the valuation of the algebraic function field $K(\mathcal{C})$ at the place P.

If $f \in \overline{\mathbb{F}}_q(\mathcal{C})$ then one can define the divisor

$$\mathrm{div}(f) = \sum_{P \in \mathcal{C}(\overline{\mathbb{F}}_q)} v_P(f)(P).$$

It is a standard result that $\deg(\mathrm{div}(f)) = 0$, since $\mathcal{C}$ is projective. The degree of f is defined as $\deg(f) = \sum_{v_P(f)>0} v_P(f) = -\sum_{v_P(f)<0} v_P(f)$.

The group of principal divisors is

$$\mathrm{Prin}(\mathcal{C}) = \{\mathrm{div}(f) : f \in \overline{\mathbb{F}}_q(\mathcal{C})\}$$

which is a subgroup of $\mathrm{Div}^0(\mathcal{C})$. The divisor class group is defined to be the quotient group

$$\mathrm{Pic}^0(\mathcal{C}) = \mathrm{Div}^0(\mathcal{C})/\mathrm{Prin}(\mathcal{C}).$$

Some authors write $D_1 \sim D_2$ or $D_1 \equiv D_2$ to represent equivalence of divisors in the quotient, in other words that there exists a function f such that

$D_1 = D_2 + \mathrm{div}(f)$. The equivalence class containing a divisor D is called a divisor class and is denoted $\overline{D}$.

To obtain a finite group, we must consider divisor classes over $\mathbb{F}_q$, not $\overline{\mathbb{F}}_q$. A divisor D is said to be K-rational if $D^\sigma = D$ for all $\sigma \in \mathrm{Gal}(\overline{\mathbb{F}}_q/K)$. We define

$$\mathrm{Div}^0_K(\mathcal{C}) = \{D \in \mathrm{Div}^0(\mathcal{C}) : D^\sigma = D \ \ \forall \sigma \in \mathrm{Gal}(\overline{\mathbb{F}}_q/K)\}.$$

We define $\mathrm{Prin}_K(\mathcal{C}) = \{\mathrm{div}(f) : f \in K(\mathcal{C})\}$ and then

$$\mathrm{Pic}^0_K(\mathcal{C}) = \mathrm{Div}^0_K(\mathcal{C})/\mathrm{Prin}_K(\mathcal{C})$$

and one can prove that this is isomorphic to the subgroup of $\mathrm{Pic}^0(\mathcal{C})$ which is invariant under $\mathrm{Gal}(\overline{\mathbb{F}}_q/K)$. For $r \in \mathbb{N}$ we define

$$\mathrm{Pic}^0_K(\mathcal{C})[r] = \{D \in \mathrm{Pic}^0_K(\mathcal{C}) : rD = 0\}.$$

The Riemann hypothesis for function fields (proved by Weil) implies that $\#\mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C}) \approx q^g$ for q large and g small. This is the primary motivation for considering hyperelliptic curves for cryptography: for a k-bit group size one can work over a finite field of about k/g bits.

The Jacobian of a curve $\mathcal{C}$ is the abelian variety $\mathrm{Jac}(\mathcal{C})$ that contains $\mathcal{C}$ and has the property that, for every extension field K of $\mathbb{F}_q$, the groups $\mathrm{Jac}(\mathcal{C})(K)$ and $\mathrm{Pic}^0_K(\mathcal{C})$ are isomorphic.

2.2 Mumford Representation

In this section we assume $\mathcal{C}$ is a hyperelliptic curve of genus g over K with a single K-rational point ∞ at infinity. To be able to explicitly compute with elements of $\mathrm{Pic}^0_K(\mathcal{C})$ we have to choose a compact representation of the elements. To this end we introduce the following notion.

Definition 1. *A divisor $D \in \mathrm{Div}^0_K(\mathcal{C})$ on a hyperelliptic curve $\mathcal{C}$ of genus g is called* semi-reduced *if it can be written as $D = E - d(\infty)$ with E effective and for $P = (x, y)$ with $2y + H(x) = 0$ one has $v_P(E) \in \{0, 1\}$, and for $P = (x, y)$ and $P' = (x, -y - H(x))$ with $P \neq P'$ (equivalently $2y + H(x) \neq 0$) one has $v_P(E)v_{P'}(E) = 0$. If moreover $\deg(E) \leq g$ then D is called* reduced.

Every divisor class in $\mathrm{Pic}^0_K(\mathcal{C})$ contains exactly one reduced divisor. A reduced divisor $D = E - d(\infty)$ is represented in Mumford representation as a pair $[u(x), v(x)]$ of polynomials in $K[x]$ such that: $u(x)$ is monic, $u(x)$ divides $F(x) - H(x)v(x) - v(x)^2$, $\deg(v(x)) < \deg(u(x)) \leq g$. Subject to these conditions, the relation between $E = \sum_{i=1}^{d}(x_i, y_i)$ and $[u(x), v(x)]$ is as follows: $u(x) = \prod_{i=1}^{d}(x - x_i)$ and $v(x_i) = y_i$. This yields a 1–1 correspondence between reduced divisors and their Mumford representation.

2.3 Cantor's Algorithm

We continue to assume that $\mathcal{C}$ is a hyperelliptic curve over K with a single K-rational point at infinity. An algorithm for adding general divisor classes in

Algorithm 1. Cantor Addition

INPUT: Divisors $D_1 = [u_1, v_1]$ and $D_2 = [u_2, v_2]$
OUTPUT: A reduced divisor D representing the sum $D_1 + D_2$ in $\mathrm{Pic}^0_K(\mathcal{C})$.
1: $d_1 \leftarrow \gcd(u_1, u_2) = e_1 u_1 + e_2 u_2$
2: $d \leftarrow \gcd(d_1, v_1 + v_2 + H) = c_1 d_1 + c_2(v_1 + v_2 + H)$
3: $s_1 \leftarrow c_1 e_1$, $s_2 \leftarrow c_1 e_2$, $s_3 \leftarrow c_2$
4: $u \leftarrow (u_1 u_2)/d^2$, $v \leftarrow (s_1 u_1 v_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + F))/d \bmod u$
5: **while** $\deg(u) > g$ **do**
6: $\quad u' \leftarrow \mathrm{Monic}((F - vH - v^2)/u)$, $v' \leftarrow (-H - v) \bmod u'$
7: $\quad u \leftarrow u', v \leftarrow v'$
8: **end while**
9: **return** $D = [u', v']$.

this case was developed by Cantor [9] for the case that $H(x) = 0$ and the characteristic of K is not 2. The general case was worked out by Koblitz in [30].

Cantor's algorithm is given in Algorithm 1, but as presented here, the addition algorithm is not very efficient, since it requires an extended Euclidean gcd computation in Steps 1 and 2. However if one fixes the genus g, one can work out specific algorithms dedicated to the various possible values of $\deg(u_1)$ and $\deg(u_2)$ [25,32,33]. This way one can formulate algorithms that are much more efficient, and that avoid high-level operations like Euclidean algorithms.

The relation with divisor equivalence is as follows: in Step 4 we obtain a semi-reduced divisor D represented by $[u(x), v(x)]$ that satisfies

$$D = D_1 + D_2 - \mathrm{div}(d(x)) \,. \tag{1}$$

The divisor D is then further reduced in the loop in Step 5 to a divisor D' represented by $[u'(x), v'(x)]$ which satisfies

$$D' = D - \mathrm{div}((y - v(x))/u'(x)) \,. \tag{2}$$

3 Elliptic Versus Hyperelliptic Curves

The primary motivation for considering curves of genus $g > 1$ for cryptography is the fact that the group size grows as q^g. In other words, with genus 2 one can get a given group size by working over a field $\mathbb{F}_q$ where q has half the bit length needed if working with elliptic curves.

There has been much discussion about whether or not genus 2 curves can be faster for non-pairing cryptography than elliptic curves. To get comparable timings it is crucial to replace Cantor's algorithm with some optimised formulae [25,32,33]. Nevertheless, it seems that fully general implementations of genus 2 curves are slower than general implementations of elliptic curves.

However, there are several special tricks available for genus 2 curves, which have no counterpart for elliptic curves. We will briefly mention two such tricks here.

The first is to use curves (in characteristic 2) of the special form

$$y^2 + xy = x^5 + F_3x^3 + F_2x^2 + F_0.$$

For these curves, the optimised group law formulae require one inversion, 6 squarings and 5 multiplications [33,34], which is very competitive compared with elliptic curves. For pairings using supersingular curves in characteristic 2 one has curves which are even more special, and thus faster, than this. One can avoid inversions if some form of projective coordinates is used.

A second trick is to utilise special divisors. Recall that a general reduced divisor D on a genus 2 curve has support consisting of two affine points (i.e., $D = (P_1) + (P_2) - 2(\infty)$). Following [28,29] one can exploit the benefits of using *degenerate divisors* of the form $D = (P) - (\infty)$. (Note that the definition of degenerate divisors in [28,29] is that they have less than g points in their support, whereas our definition is more restrictive when $g > 2$ in that we insist on having exactly one affine point in the support.) The addition operations are faster when adding a general divisor to a degenerate divisor, than when adding two general divisors.

To summarise, hyperelliptic curves can be competitive with elliptic curves (sometimes, even faster) due to the smaller field size, as long as one exploits optimised addition formulae for special curves and one uses special divisors.

4 A World of Pairings

4.1 Weil and Tate-Lichtenbaum Pairings

Let $\mathcal{C}$ be a non-singular projective curve of genus g over $\mathbb{F}_q$. Let r be coprime to q. It is typical for cryptographic applications to take r to be a (large) prime divisor of $\#\mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})$. It is often the case that $r \approx q^g$, but in some situations it is necessary to take r smaller. The embedding degree is defined to be the smallest positive integer k such that $r \mid (q^k - 1)$. Note that the embedding degree is a function of q and r. The subgroup of r-th roots of unity of $\mathbb{F}^\times_{q^k}$ is denoted by $\mu_r = \{z \in \mathbb{F}^\times_{q^k} : z^r = 1\}$.

The Weil pairing [52,39] is defined to be a non-degenerate bilinear map

$$\mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})[r] \times \mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})[r] \longrightarrow \mu_r$$

which is denoted $e_r(\overline{D}_1, \overline{D}_2)$.

The Tate-Lichtenbaum pairing [49,36,16] is defined to be a non-degenerate bilinear map

$$\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})[r] \times \mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})/r\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C}) \longrightarrow \mathbb{F}^\times_{q^k}/(\mathbb{F}^\times_{q^k})^r$$

which is denoted $\langle \overline{D}_1, \overline{D}_2 \rangle_r$.

The domain and range of the Weil pairing are better suited for cryptographic applications, since the pairing arguments and values are given by points and finite

field elements instead of equivalence classes. For many cryptographic applications it is necessary to work with unique representatives (e.g., Alice and Bob may perform different calculations but must obtain the same element). To achieve this for the Tate-Lichtenbaum pairing, two simplifications are usually made. First, if we assume that $\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})$ contains no elements of order r^2 then we may identify $\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})[r]$ with $\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})/r\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})$, via the map $\overline{D}_2 \mapsto \overline{D}_2 + r\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})$. Second, we can map into the subgroup μ_r by raising to the power $(q^k-1)/r$. This is called the final exponentiation. Hence, we consider the reduced (or modified) Tate-Lichtenbaum pairing

$$t(\overline{D}_1, \overline{D}_2) = \langle \overline{D}_1, \overline{D}_2 \rangle_r^{(q^k-1)/r}.$$

The mathematical definition of the Tate-Lichtenbaum pairing is as follows. The argument on the left hand side of the Tate-Lichtenbaum pairing is represented by an $\mathbb{F}_{q^k}$-rational divisor D_1 of degree zero. Since $\overline{D}_1$ is a divisor class of order r, there is a function f_{r,D_1} with divisor

$$\mathrm{div}(f_{r,D_1}) = rD_1.$$

The argument of the right hand side of the Tate-Lichtenbaum pairing can be represented by an $\mathbb{F}_{q^k}$-rational divisor D_2 of degree zero such that the supports of D_1 and D_2 are disjoint. Then the Tate-Lichtenbaum pairing is defined to be

$$\langle \overline{D}_1, \overline{D}_2 + r\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C}) \rangle_r = f_{r,D_1}(D_2) = \prod_P f_{r,D_1}(P)^{v_P(D_2)}.$$

The Weil pairing usually offers inferior efficiency and flexibility in comparison with the reduced Tate-Lichtenbaum pairing (see for example [24] or [1, Section 16.1.5]). In the remainder of the paper we will therefore consider the reduced Tate pairing only. More information about the Weil pairing and its efficient computation can be found in [38].

Finally, we note that $f_{r,D}$ with $\mathrm{div}(f_{r,D}) = rD$ is only defined up to scalar multiples from $\overline{\mathbb{F}}_q^{\times}$. It is possible to find $f_{r,D}$ which is defined over the field of definition of D and we assume this in the following. We will need to impose some additional normalisation conditions on $f_{r,D}$ later.

4.2 Ate Pairings

For cryptographic purposes one applies one further simplification to the reduced Tate-Lichtenbaum pairing by restricting the pairing to certain cyclic subgroups G_1 and G_2 of $\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})[r]$ that are Frobenius eigenspaces. Write π for the q-power Frobenius map on $\mathcal{C}$ and the Frobenius endomorphism on $\mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})$. Then we define

$$G_1 = \mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})[r],$$

for which the eigenvalue of π is 1. We also define

$$G_2 = \mathrm{Pic}^0_{\mathbb{F}_{q^k}}(\mathcal{C})[r] \cap \ker(\pi - q).$$

The Weil and Tate-Lichtenbaum pairings are the two known, essentially different pairings available for curves. The elliptic and hyperelliptic ate pairings can be regarded as special, low degree variants of the Tate-Lichtenbaum pairing. Note that while the Weil and Tate-Lichtenbaum pairings are named after their inventors and are already quite classical, the ate pairing is much more recent and carries an artificial name.

Ate Pairings on Elliptic Curves. Let $\mathcal{E}$ be an ordinary elliptic curve over $\mathbb{F}_q$. Let t be the trace of the q-power Frobenius endomorphism π of $\mathcal{E}$, such that $\#\mathcal{E}(\mathbb{F}_q) = q - t + 1$. We assume that $r \geq 5$ is a sufficiently large prime factor of $\#\mathcal{E}(\mathbb{F}_q)$ and that k is minimal such that $r \| (q^k - 1)$.

If $P \in \mathcal{E}(\overline{\mathbb{F}}_q)$ has order r, then $(P) - (\infty)$ is a divisor of degree zero representing a divisor class of order r. We define $f_{r,P} = f_{r,(P)-(\infty)}$. Recall our assumption that the field of definition of $f_{r,P}$ be that of P. In addition to this, we need to normalise $f_{r,P}$ as follows. Let $z \in \mathbb{F}_q(\mathcal{E})$ be a local uniformizer at ∞, that is z satisfies $v_\infty(z) = 1$. Then we define $\mathrm{lc}_\infty(f_{r,P}) = (z^r f_{r,P})(\infty)$ and $f_{r,P}^{\mathrm{norm}} = f_{r,P}/\mathrm{lc}_\infty(f_{r,P})$. The function $f_{r,P}^{\mathrm{norm}}$ is defined over the field of definition K of P and is uniquely determined by r and P up to non-zero rth-power multiples from K.

The following theorem is from [26], using a slightly more general formulation given in [37].

Theorem 1. *Let S be an integer with $S \equiv q \bmod r$. Define $N = \gcd(S^k - 1, q^k - 1)$ and $L = (S^k - 1)/N$. Let $c_S = \sum_{i=0}^{k-1} S^{k-1-i} q^i \bmod N$. Then*

$$a_S : G_2 \times G_1 \to \mu_r, \quad (Q, P) \mapsto f_{S,Q}^{\mathrm{norm}}(P)^{c_S(q^k-1)/N}$$

defines a bilinear pairing, called the elliptic ate pairing. *If $k \mid \#\mathrm{Aut}(\mathcal{E})$ then*

$$a_S^{\mathrm{twist}} : G_1 \times G_2 \to \mu_r, \quad (P, Q) \mapsto f_{S,P}(Q)^{c_S(q^k-1)/N}$$

also defines a bilinear pairing, called the twisted ate pairing. *Both pairings a_S and a_S^{twist} are non-degenerate if and only if $r \nmid L$.*

The relation with the reduced Tate-Lichtenbaum pairing is

$$a_S(Q, P) = t(Q, P)^L \quad \text{and} \quad a_S^{\mathrm{twist}}(P, Q) = t(P, Q)^L.$$

We remark that the condition $k \mid \#\mathrm{Aut}(\mathcal{E})$ holds true if and only if $\mathcal{E}$ admits a twist of degree k. We say that $\mathcal{E}$ admits a twist of degree d if there is an elliptic curve $\mathcal{E}'$ defined over $\mathbb{F}_q$ and an isomorphism $\psi : \mathcal{E}' \to \mathcal{E}$ defined over $\mathbb{F}_{q^d}$, and d is minimal with this property. If $k \mid \#\mathrm{Aut}(\mathcal{E})$ does not hold one may still apply the theorem for a divisor e of k using a base extension of degree k/e.

One can take $S = q$ in Theorem 1, but the usual choice is $S = t - 1$, which has half the bit length of $\#\mathcal{E}(\mathbb{F}_q)$ and thus yields half the loop length of the standard reduced Tate-Lichtenbaum pairing, if $r \approx q$. In certain cases it may be possible to choose S strictly smaller than $t - 1$, which yields an even more efficient computation [37].

The Duursma-Lee pairing [11] and the η_T-pairing from [5] can be regarded as a special form of the twisted ate pairing on supersingular elliptic curves.

Ate Pairings on Hyperelliptic Curves. For hyperelliptic curves the situation is somewhat different. Indeed, if $\mathcal{C}$ is a hyperelliptic curve then $r \mid \#\mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C}) = q^g + a_1(q^{g-1}+1) + a_2(q^{g-2}+1) + \cdots + a_g$. If $r \approx \#\mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})$ then the bit length of q is already g times shorter than the bit length of r, so we may try to mimic Theorem 1 with $S = q$ (this idea, in a special case, first appears in [11]).

In order to formulate the main results from [23], we fix some notation. Let $\mathcal{C}$ be a hyperelliptic curve with a single point ∞ at infinity. For any divisor class $\overline{D}$ we denote by $\rho(\overline{D})$ the unique reduced divisor in $\overline{D}$ and by $\epsilon(\overline{D})$ the effective part of $\rho(\overline{D})$ so that we have $\rho(\overline{D}) = \epsilon(\overline{D}) - d(\infty)$. We apply the same normalisation to the function $f_{r,D}$ as above, namely $f^{\mathrm{norm}}_{r,D} = f_{r,D}/(\mathrm{lc}_\infty(f_{r,D}))$ for $\mathrm{lc}_\infty(f_{r,P}) = (z^r f_{r,P})(\infty)$ and $z \in \mathbb{F}_q(\mathcal{C})$ a local uniformizer at ∞ over $\mathbb{F}_q$. A curve is called superspecial if its Jacobian is isomorphic to E^g with E a supersingular elliptic curve. The Jacobian of superspecial curves is hence also supersingular, and in particular has p-rank zero.

Theorem 2. *([23]) With the above notation and assumptions,*

$$a : G_2 \times G_1 \to \mu_r : (\overline{D}_2, \overline{D}_1) \mapsto f^{\mathrm{norm}}_{q,\rho(\overline{D}_2)}(\epsilon(\overline{D}_1))$$

defines a non-degenerate, bilinear pairing called the hyperelliptic ate pairing. *If $\mathcal{C}$ is superspecial and $d = \gcd(k, q^k - 1)$ then*

$$\hat{a} : G_1 \times G_2 \to \mu_r : (\overline{D}_1, \overline{D}_2) \mapsto f^{\mathrm{norm}}_{q,\rho(\overline{D}_1)}(\epsilon(\overline{D}_2))^d$$

defines a non-degenerate, bilinear pairing.

If in any of the two pairings we have $\mathrm{supp}(\epsilon(\overline{D}_i)) \cap \mathrm{supp}(\rho(\overline{D}_j)) \neq \emptyset$, *then* $\epsilon(\overline{D}_i)$ *needs to be replaced by any* $D \in \overline{D}_i$ *with* $\mathrm{supp}(D) \cap \mathrm{supp}(\rho(\overline{D}_j)) = \emptyset$.

The relation with the reduced Tate-Lichtenbaum pairing is

$$t(\overline{D}_2, \overline{D}_1) = a(\overline{D}_2, \overline{D}_1)^{kq^{k-1}} \quad \textit{and} \quad t(\overline{D}_1, \overline{D}_1) = \hat{a}(\overline{D}_1, \overline{D}_2)^{(k/d)q^{k-1}}.$$

One feature of the hyperelliptic ate pairing is that the final exponentiation is very simple.

5 Pairing Friendly Curves

A curve $\mathcal{C}$ over $\mathbb{F}_q$ of genus g is called pairing friendly if there is a large prime $r \mid \#\mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})$ with relatively small embedding degree k (say, $2 \leq k \leq 30g$) and such that μ_r does not lie in a proper subfield of $\mathbb{F}_{q^k}$. Supersingular curves are pairing friendly. In this section we list some curves which are particularly suitable for efficient pairing implementation.

5.1 Elliptic Curves

The most useful cases are ordinary curves for which the value S in the elliptic ate pairing is small, and supersingular curves. An example in the supersingular

case is the curve $E : y^2 + y = x^3 + x + b$ over $\mathbb{F}_{2^m}$ where $b = 0, 1$ and m is odd. The group order is $2^m + 1 \pm 2^{(m+1)/2}$ and the embedding degree is $k = 4$. See [18,5] for more details.

Elliptic curves suitable for the elliptic ate pairing can be constructed using the method of Brezing and Weng [8]. A family which is especially adapted to the ate pairing is given by the following parameterisation (which for $n = 1$ was given already in [3]): let $r(x) = \Phi_{12n}(x)$ for any positive integer n coprime to 3, $t(x) = x + 1$ and $s(x) = (2x^{2n} - 1)$. Following the method of [8] we define $p(x) = (t(x)^2 - s(x)^2(t(x) - 2)^2/D)/4$ with $D = -3$. If x is such that $p(x)$ and $r(x)$ are prime, then one can easily construct an elliptic curve $y^2 = x^3 + b$ with embedding degree $12n$. For $n = 1$ then $\deg(r(x)) = 4$ and $\deg(p(x)) = 6$, but for larger values of n one can get $\deg(r(x)) \approx \deg(p(x))$. Note that the trace is as small as possible compared to r and that the elliptic curve admits sextic twists; both these features are desirable for fast implementations of the ate pairings.

5.2 Hyperelliptic Curves

An example of a supersingular genus 2 curve is $C_d : y^2 + y = x^5 + x^3 + d$ with $d = 0$ or 1 over $\mathbb{F}_{2^m}$, where m is coprime to 6. This curve has embedding degree 12 (see [18,5] for more details).

An example of a family of superspecial hyperelliptic curves was studied by Duursma and Lee [11]. They considered the curves $C : y^2 = x^p - x + d$ over $\mathbb{F}_{p^m}$ where $d = \pm 1$. The genus of C is $(p - 1)/2$. When $p \equiv 3 \bmod 4$ the embedding degree is $k = 2p$. If $p \equiv 1 \bmod 4$ then the embedding degree is $k = p$. It is worth noting that the fast pairing algorithms of [11,5] required the condition $p \equiv 3 \bmod 4$, but the ate method works for all cases.

In characteristic $p \geq 5$, the best one can do with supersingular genus 2 curves is $k = 6$. In [20] it is shown how to obtain suitable curves for any $p \equiv 2 \bmod 3$ by taking twists of the supersingular curve $y^2 = x^6 + 1$.

It is natural to try to use non-supersingular curves of genus $g \geq 2$ for pairing-based cryptography. However, it seems to be hard to generate suitable curves in this case. The paper [19] gives some first steps towards solving this problem, by presenting some quadratic polynomial families of abelian surfaces with given embedding degree. However, [19] proves that for some embedding degrees (e.g., $k = 8$ and $k = 12$) there are no such quadratic families. These results suggest that there is less structure in the genus 2 case (or, at least, that the structure is more complicated in the genus 2 case) than in the elliptic case. For the polynomial families found in [19] the authors were unable to generate any curves using the CM method. Indeed, the CM method for curves of genus ≥ 2 is much less well developed than the CM method in the elliptic case.

The first examples of non-supersingular pairing-friendly genus 2 curves are due to Freeman [14]. The parameters for these curves are not very attractive for fast pairing implementation (precisely, the size of r is too small compared with the size of q). More research is needed on methods to generate such curves with parameters suitable for cryptography.

Note that many of the computational assumptions in pairing based cryptography can be solved in subexponential time, hence it may not be necessary to restrict to very small genus g, as is usually done for non-pairing cryptography. Nevertheless, in our comparison with elliptic curves we will assume that $g \leq 5$.

6 Miller's Algorithm

This section reviews the generalisation of Miller's algorithm for computing Tate-Lichtenbaum or ate pairings efficiently on hyperelliptic curves $y^2+H(x)y = F(x)$ over $\mathbb{F}_q$ with a single point at infinity. We make the following assumptions:

- We are pairing two reduced divisors D_1 and D_2 with Frobenius eigenvalues 1 and q (for the Tate-Lichtenbaum pairing, D_1 has eigenvalue 1 and D_2 has eigenvalue q, for the ate pairing it is the other way around).
- The pairing is computed as $f_{S,D_1}(\epsilon(\overline{D}_2))^d$ for some integers S and d.

6.1 Hyperelliptic Miller

Miller's algorithm computes functions $f_{n,D}$ with divisor $\mathrm{div}(f_{n,D}) = nD - D_n$, where $D_n = \rho(n\overline{D})$, i.e. the unique reduced divisor equivalent to nD. It basically consists of a double and add algorithm exploiting the fact that one can define

$$f_{k+l,D} = f_{k,D} \cdot f_{l,D} \cdot h_{D_k,D_l} ,$$

with $\mathrm{div}(h_{D_k,D_l}) = D_k + D_l - \rho(\overline{D}_k + \overline{D}_l)$. These functions h_{D_k,D_l} are obtained easily from Cantor's algorithm by equations (1) and (2).

However, we are not really interested in the functions f_{n,D_1} themselves, but only in the evaluation $f_{n,D_1}(E)$ for some effective divisor E, so we need a method to evaluate h_{D_k,D_l} at E. Since this evaluation step is the crucial part of Miller's algorithm, we describe two different methods in detail (namely using a norm computation and using resultants). The papers [23,35] use resultants for pairing computation.

Since any function can be written as a fraction of two polynomials, we can limit ourselves to evaluating a polynomial $h(x, y) \in \mathbb{F}_q[x, y]$ at E for some $\mathbb{F}_q$-rational divisor $D = E - d(\infty)$, with Mumford representation $[u_E(x), v_E(x)]$.

The first method is a simple optimisation of the definition of function evaluation at a divisor. Let $E = \sum_{i=1}^{d}(x_i, y_i)$, then we can compute the support of E by factoring $u_E(x)$ as $\prod_{i=1}^{d}(x - x_i)$ and setting $y_i = v_E(x_i)$. In general each (x_i, y_i) will be defined over some extension field $\mathbb{F}_{q^{e_i}}$, where $e_i \leq g$. Of course, we could then compute $h(E)$ as $\prod_{i=1}^{d} h(x_i, y_i)$, but in general this is not the best method, since we are not exploiting the fact that the result has to be in $\mathbb{F}_q$. Instead, one could partition the support into distinct Galois orbits

$$\left\{(x_i, y_i), (x_i^q, y_i^q), \ldots, (x_i^{q^{e_i-1}}, y_i^{q^{e_i-1}})\right\}$$

and compute the product of the evaluations at these points simply by computing $N_{\mathbb{F}_{q^{e_i}}/\mathbb{F}_q}(h(x_i, y_i))$. The above method is in general suboptimal due to the

Algorithm 2. Miller Step

INPUT: $D_1 = [u_1, v_1]$, $D_2 = [u_2, v_2]$, $E = [u_E, v_E]$.
OUTPUT: Reduced divisor $\rho(\overline{D_1} + \overline{D_2})$ and evaluation $h^{\mathrm{norm}}_{D_1,D_2}(E)$, represented by $[\tilde{h}_1(x), \tilde{h}_2(x), h_3]$.

$d_1 \leftarrow \gcd(u_1, u_2) = e_1 u_1 + e_2 u_2$
$d \leftarrow \gcd(d_1, v_1 + v_2 + H) = c_1 d_1 + c_2(v_1 + v_2 + H)$
$\tilde{h}_1 \leftarrow d \bmod u_E$, $\tilde{h}_2 \leftarrow 1$, $h_3 \leftarrow 1$
$s_1 \leftarrow c_1 e_1$, $s_2 \leftarrow c_1 e_2$, $s_3 \leftarrow c_2$
$u \leftarrow (u_1 u_2)/d^2$, $v \leftarrow (s_1 u_1 v_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + F))/d \bmod u$
while $\deg(u) > g$ **do**
    $u' \leftarrow \mathrm{Monic}((F - vH - v^2)/u)$, $v' \leftarrow (-H - v) \bmod u'$
    $\tilde{h}_1 \leftarrow \tilde{h}_1 \cdot (v_E - v) \bmod u_E$
    $\tilde{h}_2 \leftarrow \tilde{h}_2 \cdot u' \bmod u_E$
    if $\deg(v) > g$ **then**
        $h_3 \leftarrow -\mathrm{lc}(v) \cdot h_3$ ▷ lc = leading coefficient
    end if
    $u \leftarrow u', v \leftarrow v'$
end while
return $[u, v], [\tilde{h}_1, \tilde{h}_2, h_3]$

Algorithm 3. Miller's algorithm (base 2)

INPUT: $S = \sum_{i=0}^{B} S_i 2^i$, d, $D_1 = [u_1, v_1]$, $D_2 = [u_2, v_2]$.
OUTPUT: Pairing value $f_{S,D_1}(\epsilon(\overline{D_2}))^d$

$D \leftarrow [u_1, v_1]$
$f \leftarrow 1$, $f_1 \leftarrow 1$, $f_2 \leftarrow 1$, $f_3 \leftarrow 1$
for $i \leftarrow B - 1$ **downto** 0 **do**
    $f_1 \leftarrow f_1^2 \bmod u_2$, $f_2 \leftarrow f_2^2 \bmod u_2$, $f_3 \leftarrow f_3^2$
    $D, [h_1, h_2, h_3] \leftarrow$ Miller Step(D, D, D_2)
    $f_1 \leftarrow f_1 \cdot h_1 \bmod u_2$, $f_2 \leftarrow f_2 \cdot h_2 \bmod u_2$, $f_3 \leftarrow f_3 \cdot h_3$
    if $S_i = 1$ **then**
        $D, [h_1, h_2, h_3] \leftarrow$ Miller Step(D, D_1, D_2)
        $f_1 \leftarrow f_1 \cdot h_1 \bmod u_2$, $f_2 \leftarrow f_2 \cdot h_2 \bmod u_2$, $f_3 \leftarrow f_3 \cdot h_3$
    end if
end for
$f \leftarrow \mathrm{Res}(u_2, f_1)/(f_3^{\deg(u_2)} \cdot \mathrm{Res}(u_2, f_2))$
return f^d

overhead of factoring the polynomial $u_E(x)$. Therefore, we advise to restrict its use to the case of degenerate divisors.

The second method is in general faster than the first, since it does not involve any polynomial factorisations (nor any explicit arithmetic in extension fields the way we have described it). It is based on the following basic observation: the univariate polynomial $\tilde{h}(x) = h(x, v_E(x))$ satisfies $\tilde{h}(x_i) = h(x_i, y_i)$, so we have reduced the problem to computing $\prod_{i=1}^{d} \tilde{h}(x_i)$ where the x_i are the zeros of

the monic polynomial $u_E(x)$. But this corresponds to the very definition of the resultant of the two polynomials $u_E(x)$ and $\tilde{h}(x)$, so we conclude

$$h(E) = \mathrm{Res}(u_E(x), h(x, v_E(x))) .$$

Note that the above resultant also equals $\mathrm{Res}(u_E(x), \tilde{h}(x) \bmod u_E(x))$, so it suffices to work with polynomials of degree smaller than g. In more mathematical terms, we still compute $h(E)$ by using a norm, namely $h(E) = N_{A/\mathbb{F}_q}(h(x, y))$ with A the finite $\mathbb{F}_q$-algebra $\mathbb{F}_q[x, y]/(u_E(x), y - v_E(x))$.

Algorithm 2 below executes one step in Miller's algorithm and is a simple adaptation of Cantor's algorithm. It computes the evaluation of $h^{\mathrm{norm}}_{D_1,D_2}(E)$ represented as follows: assume that $h_{D_1,D_2} = h_1(x, y)/h_2(x, y)$, then the algorithm returns $\tilde{h}_1(x) = h_1(x, v_E(x)) \bmod u_E(x)$ and $\tilde{h}_2(x) = h_2(x, v_E(x)) \bmod u_E(x)$ and the constant $h_3 = \mathrm{lc}_\infty(h_{D_1,D_2})$, so we conclude that

$$h^{\mathrm{norm}}_{D_1,D_2}(E) = \mathrm{Res}(u_E(x), \tilde{h}_1(x))/(h_3^{\deg(u_E)} \cdot \mathrm{Res}(u_E(x), \tilde{h}_2(x))) .$$

We assume in Algorithm 2 that all the intermediate functions are defined on $\epsilon(E)$ (this is a reasonable assumption for the cryptographic applications).

These partial evaluations are then combined in Algorithm 3 below using a double and add strategy. Note that the resultant computation is postponed to the very end of the algorithm. The alternate strategy of computing the resultant each time instead of computing modulo $u_E(x)$ is faster only in the genus 2 case.

6.2 Improvements to Miller's Algorithm

There are a number of standard implementation techniques to speed up pairing computation (see [2,4,11,17,22]). We always assume that the embedding degree satisfies $k > 1$. The improvements include the following.

- Using suitable representations for $\mathbb{F}_{q^k}$, such as pairing friendly fields [31], i.e. $\mathbb{F}_q$ is a prime field with $q \equiv 1 \bmod 12$ and k of the form $2^i 3^j$. Using a combination of Karatsuba and Toom-Cook, multiplication in such fields takes $3^i 5^j$ multiplications in the base field $\mathbb{F}_q$.
- Changing the base in Miller's algorithm. For example, base 3 is used in [2,17] and base 8 is used in [5] in genus 2.
- Working with divisors of the form $\epsilon(\overline{D}_2)$ and computing leading coefficients instead of evaluating functions on the reduced divisor D_2 itself. Since the support of D_2 has ∞ in common with the support of D_1, the alternative is to "shift" D_2 and this leads to extra function evaluations.
- Denominator elimination [11,5]. If a final exponentiation is performed then one can omit all terms lying in a proper subfield K such that $\mathbb{F}_q \subseteq K \subset \mathbb{F}_{q^k}$. In many cases (e.g. if one has even embedding degree, and D_1 and D_2 lie in the 1-eigenspace and q-eigenspace of Frobenius respectively) then the denominator f_2 and leading coefficient f_3 in Algorithm 3 are of this form, hence the name denominator elimination.

- The loop length and number of additions in Miller's algorithm depend on the bit-length of S and its Hamming weight. Therefore, it is convenient to choose S as small as possible and to have low Hamming weight.
- Pairing value compression, using trace or torus methods [22,43].
- Speeding up the final exponentiation, by exploiting the special form of the integer d and/or using trace or torus methods [4,22,24]. For instance, it is not difficult to prove that $r \mid \Phi_k(q)$, so a final exponentiation of the form $(q^k - 1)/r$ can be written as

$$\frac{q^k - 1}{r} = \frac{\Phi_k(q)}{r} \cdot \prod_{s|k, s<k} \Phi_s(q)$$

  where exponentiation by the last factor can be computed extremely fast using q-th power Frobenius operations.

The techniques mentioned above give impressive results for pairing implementation and they generalise trivially to hyperelliptic curves of genus $g \geq 2$.

7 Degenerate Divisors Versus General Divisors

In non-pairing cryptography it was noted by Katagi et al. [28,29] that using degenerate divisors can give performance advantages. In [5,15] it is explained how degenerate divisors can be used to speed up pairing-based cryptosystems. For example, in the Boneh-Franklin identity-based encryption scheme [6], one can choose D_{pub} to be degenerate (i.e., choose $D_{\text{pub}} = (P) - (\infty)$ first and then set $D = [s^{-1}]D_{\text{pub}}$). One can also choose $H(ID)$ to be degenerate, so that we hash to points rather than general divisors. Encryption in the Boneh-Franklin scheme then involves a pairing of two degenerate divisors, so is fast. Decryption still requires pairings of general divisors.

Several practical issues arise. First, can one choose a divisor D_{pub} of prime order of the form $(P) - (\infty)$? If one is choosing elements of G_1 and the divisor class group has prime order then this is automatic. The general case is discussed by Frey and Lange [15]. They also note that, when k is even, one might choose the second pairing argument from

$$\{(x, y) \in \mathbb{F}_{q^k} : x \in \mathbb{F}_{q^{k/2}},\ y \in \mathbb{F}_{q^k} \backslash \mathbb{F}_{q^{k/2}}\},$$

which is not a group but which is a much bigger set than the usual choice of G_2.

Second, is there a loss of security from using degenerate divisors? It is easy to show that the DLP for degenerate divisors is not easier than the general DLP [29]. However, some protocols require more for security than just hardness of the DLP. For example, in the Boneh-Franklin scheme it must be hard to find collisions $H(ID_1) = H(ID_2)$. The presentation in [5] is disingenuous in this regard: since the number of degenerate divisors is roughly q, one can find collisions in time $O(q^{1/2})$, yet [5] gives running times for examples where the security level is supposed to be $O(q)$.

To summarise, the three advantages of using degenerate divisors in pairing based cryptography are: faster computation of divisor multiplication and pairings, simpler hashing into the group, and reduced bandwidth for transmitting group elements. Only the first of these is really an advantage when compared with elliptic curves. Hashing for elliptic curves is always to a single point, so is easy (and we can often exploit twists to make it faster [26]). Regarding small representatives of group elements, for elliptic curves over $\mathbb{F}_p$ one could use points $(x, y) \in E(\mathbb{F}_p)$ where $1 < x < p^{1/g}$ (with a single bit to determine y) and so the bandwidth requirement is the same as the hyperelliptic case.

8 Torsion and Distortion

The results of this section are taken from [20,41]. Let $\mathcal{C}$ be a supersingular curve of genus g over $\mathbb{F}_q$ and let $r > 2$ be prime and coprime to q. It is a standard fact that $\mathrm{Pic}^0_{\mathbb{F}_q}(\mathcal{C})[r]$ is isomorphic to $(\mathbb{Z}/r\mathbb{Z})^{2g}$. We let e be any non-degenerate, bilinear, Galois-invariant pairing which maps to μ_r.

For pairings on elliptic curves with $k > 1$ one typically has $\mathcal{E}(\overline{\mathbb{F}}_q)[r] = \langle P, Q\rangle$ (here the notation $\langle P, Q\rangle$ means the subgroup generated by P and Q) where P is defined over $\mathbb{F}_q$ and Q is defined over $\mathbb{F}_{q^k}$. It follows that $e(P, P) = 1$ and hence, by non-degeneracy, $e(P, Q) \neq 1$. In other words, it is relatively simple to classify pairs of points whose pairing is non-trivial. In particular, if P is $\mathbb{F}_q$-rational and ψ is any endomorphism of $\mathcal{E}$ such that $\psi(P) \notin \langle P\rangle$ then it follows that $e(P, \psi(P)) \neq 1$.

In the higher genus case things are more complicated. Since a pairing $e(D, \cdot)$ defines a linear map from $(\mathbb{Z}/r\mathbb{Z})^{2g}$ to $\mathbb{Z}/r\mathbb{Z}$ it follows that the kernel of $e(D, \cdot)$ is $(2g-1)$-dimensional. Hence, the condition $\psi(D) \notin \langle D\rangle$ is not sufficient to imply that $e(D, \psi(D)) \neq 1$.

Definition 2. *([20,50,51]) A* distortion map *for a non-degenerate pairing e and non-zero divisor classes D_1, D_2 of prime order r on $\mathcal{C}$ is an endomorphism ψ of* $\mathrm{Jac}(\mathcal{C})$ *such that $e(D_1, \psi(D_2)) \neq 1$.*

It was proved in [51] that distortion maps always exist for supersingular elliptic curves over $\mathbb{F}_q$. The result was generalised in [20]. We denote by $\mathrm{End}(A)$ the ring of endomorphisms on the abelian variety A defined over $\overline{\mathbb{F}}_q$.

Theorem 3. *([20]) Let A be a supersingular abelian variety of dimension g over $\mathbb{F}_q$. Let $r \mid \#A(\mathbb{F}_q)$ be prime. Let D_1, D_2 be non-trivial elements of $A(\overline{\mathbb{F}}_q)$ of order r. Then there is an element $\psi \in \mathrm{End}(A)$ such that $e(D_1, \psi(D_2)) \neq 1$.*

Furthermore, it is shown in [20] that, to have distortion maps for every non-trivial pair of divisors, it is necessary that the $\mathbb{Z}$-rank of the endomorphism ring is equal to $(2g)^2$. In other words, if $\mathrm{Jac}(\mathcal{C})$ has endomorphism ring which has $\mathbb{Z}$-rank strictly less than $(2g)^2$ then there will exist non-zero divisor classes D_1 and D_2 in $\mathrm{Jac}(\mathcal{C})[r]$ such that $e(D_1, \psi(D_2)) = 1$ for all $\psi \in \mathrm{End}(\mathrm{Jac}(\mathcal{C}))$. In other words, if $\mathcal{C}$ is not supersingular then there cannot be distortion maps for every pair (D_1, D_2).

In practice, as in Section 4.2, one tends to choose divisors D_1 and D_2 which lie in eigenspaces of the q-power Frobenius π. The following results may be of practical relevance in this setting.

Lemma 1. *Let A be a supersingular abelian variety over $\mathbb{F}_q$ with characteristic polynomial of Frobenius equal to $T^4 + aT^2 + q^2$ and suppose $r \mid \#A(\mathbb{F}_q) = (q^2 + a + 1)$. Then then Frobenius eigenvalues on $A[r]$ are $1, -1, q, -q$.*

Proof. We have $a \equiv -(q^2 + 1) \bmod r$. Hence,

$$\begin{aligned}(T-1)(T-q)(T+1)(T+q) &= (T^2 - (q+1)T + q)(T^2 + (q+1)T + q) \\ &= T^4 - (q^2+1)T^2 + q^2 \\ &\equiv T^4 + aT^2 + q^2 \bmod r.\end{aligned}$$

Since the splitting of the characteristic polynomial of Frobenius is of this form, then the eigenvalues of Frobenius on $A[r]$ are $(1, -1, q, -q)$. □

Lemma 2. *With notation as above, let (D_1, D_2, D_3, D_4) be an ordered π-eigenbasis for $A[r]$ with eigenvalues $(1, -1, q, -q)$ respectively. Suppose $r \nmid (q^2 - 1)$. Then if $1 \le i, j \le 4$, we have $e(D_i, D_j) = 1$ unless $(i, j) = (1, 3), (3, 1), (2, 4)$ or $(4, 2)$.*

Proof. We use Galois invariance of e. For example, for D_1 one has

$$\pi(e(D_1, D_1)) = e(\pi(D_1), \pi(D_1)) = e(D_1, D_1).$$

This implies $e(D_1, D_1) \in \mathbb{F}_q \cap \mu_r$ (recall that μ_r is the group of r-th roots of unity and $r \nmid (q-1)$) and hence $e(D_1, D_1) = 1$.

Similarly,

$$e(D_1, D_2)^q = \pi(e(D_1, D_2)) = e(\pi(D_1), \pi(D_2)) = e(D_1, -D_2) = e(D_1, D_2)^{-1}.$$

Since, $r \nmid (q+1)$ this implies $e(D_1, D_2) = 1$.

Similarly,

$$e(D_1, D_4)^q = \pi(e(D_1, D_4)) = e(\pi(D_1), \pi(D_4)) = e(D_1, -qD_4) = e(D_1, D_4)^{-q}.$$

Since $r \nmid 2q$ it follows that $e(D_1, D_4) = 1$. By non-degeneracy of e, one must have $e(D_1, D_3) \neq 1$.

The other cases are similar. □

It would be interesting to develop cryptographic protocols which utilise this torsion structure and the properties of pairings stated in the above Lemma.

We see that π can be used as a distortion map. For example, suppose $D = D_1 + D_2$ and $D' = D_3 + mD_4$, with respect to the basis above, where $m \in \mathbb{Z}$ is such that $e(D, D') = e(D_1, D_3)e(D_2, D_4)^m = 1$. Then we have

$$e(D, \pi(D')) = e(D_1, qD_3)e(D_2, -qmD_4) = e(D_1, D_3)^q e(D_2, D_4)^{-qm}$$

and this is not equal to 1 if $m \not\equiv 0 \bmod r$. Note that, for efficient implementation, there are often reasons to prefer the trace map $\mathrm{Tr}(D) = \sum_{i=0}^{3} \pi^i(D)$ to π, though in the above example we have $\mathrm{Tr}(D') = 0$ so in this particular case the trace map is not useful.

9 Rubin-Silverberg Point Compression

Rubin and Silverberg [42] (also see [44,45,48]) have proposed an alternative way to view pairings on abelian varieties. They observe that many supersingular abelian varieties can be identified with subvarieties of Weil restrictions of supersingular elliptic curves. An alternative way to view their method is as a form of point compression for elliptic curves.

For example, the supersingular genus 2 curve $\mathcal{C}$ over $\mathbb{F}_2$ considered in [5] (also see Section 5.2) has $k = 12$. Working over $\mathbb{F}_{2^m}$, where m is coprime to 12, the number of points on the Jacobian of $\mathcal{C}$ is $N = 2^{2m} \pm 2^{(3m+1)/2} + 2^m \pm 2^{(m+1)/2} + 1$. As mentioned above, one can use this curve for pairing-based cryptography and the finite field security comes from $\mathbb{F}_{2^{12m}}$. Alternatively, one can consider the supersingular elliptic curve $E_b : y^2 + y = x^3 + x + b$ with $b = 0, 1$ over $\mathbb{F}_{2^{3m}}$. This curve has $k = 4$, so we also map into $\mathbb{F}_{2^{12m}}$. Furthermore, the order of $E_b(\mathbb{F}_{2^{3m}})$ (for the right choice of b) is divisible by N. Indeed, $\mathrm{Jac}(\mathcal{C})$ can be identified with the trace zero part of the Weil restriction of scalars E_b with respect to $\mathbb{F}_{2^{3m}}/\mathbb{F}_{2^m}$.

In the above example, from a security point of view, there is no difference between computing pairings on $E_b(\mathbb{F}_{2^{3m}})$ and $\mathrm{Jac}(C)(\mathbb{F}_{2^m})$. However, one can represent a general divisor on the genus 2 curve over $\mathbb{F}_{2^m}$ with about $2m$ bits, whereas it requires about $3m$ bits to represent a point in $E_b(\mathbb{F}_{2^{3m}})$. The contribution of Rubin and Silverberg is to give a method to represent elements of order N in $E_b(\mathbb{F}_{2^{3m}})$ using only $2m$ bits. In other words, the "per bit" security of the genus 2 curve is attained using elliptic curves. For precise details in this case see [5,48].

The above is an example of the Rubin-Silverberg method with, in their notation, $r = 3$. The compression method is trivial (and fast). The decompression method involves solving some non-linear equations over the finite field, and it is practical only for small values of r (e.g., $r = 3$ or 5). Note that the method can be used for any elliptic curve, not just supersingular ones.

In [5] a performance comparison is given for the above example. When pairing degenerate divisors on the genus 2 curve (such elements do not correspond to special points on the elliptic curve side) the timings in [5] show that using genus 2 curves in this setting is faster than the Rubin-Silverberg approach. However, if required to compute the pairing of general divisors on the genus 2 curve, then the timings indicate that the Rubin-Silverberg approach using the eta/ate pairing on supersingular elliptic curves is faster.

It therefore seems that, for parameters of current practical interest, it is more efficient to use elliptic curves with the Rubin-Silverberg compression method, than to work with divisor class groups of curves of genus $g \geq 2$. However, it should be noted that there are three possible exceptions to this statement: when degenerate divisors can be exploited the pairing may be faster for curves of genus $g \geq 2$; the decompression method is only practical for small values of r (whereas comparable performance with some high genus curves could require larger r); Abelian varieties have a richer torsion structure which may be useful for some applications.

10 Comparison of Pairings on Elliptic and Hyperelliptic Curves

In this section we compare the characteristics of pairings on elliptic and hyperelliptic curves of genus g. We write

$$e : G_1 \times G_2 \to G_T \subseteq \mathbb{F}_{q^k}^{\times}$$

for any of the pairings considered above. Here we assume that G_1 is the subgroup defined over the small field (so it has the more compact representation) and G_2 is the subgroup potentially defined over a larger field (hence, if using the ate pairing then $e(P,Q) = a_S(Q,P)$).

The main criteria for our comparison are:

- Computation time (for operations in G_1, G_2, G_T as well as for computing e).
- Size of representation (for G_1, G_2 and G_T).
- Flexibility and efficiency of parameter generation.
- Any other special properties.

10.1 Computation Time

In most situations we may assume that G_1 is a group of prime order r, consisting of elements defined over $\mathbb{F}_q$, with $r \approx q^g$. In this case, according to [33,34], hyperelliptic curves can be as much as twice as fast as elliptic curves. If the hyperelliptic curve is more general (e.g., has 2 points at infinity) then the computation times in G_1 may be a little slower for the hyperelliptic curve than the elliptic case.

If G_1 is a rather small subgroup of the whole curve group then define $\rho = g\log(q)/\log(r)$. If the elliptic and hyperelliptic cases have similar values for ρ then the performance should be comparable. But operations on a hyperelliptic curve with $\rho \geq 2$ (for example, Freeman's genus 2 curves [14]) will be slower than an elliptic curve with $\rho \approx 1$.

We now consider operations in G_2. In the supersingular case, $G_2 = G_1$ and the above remarks apply. More generally, we aim to take G_2 to be a subgroup of the divisor class group of a twist of the curve. The field of definition of G_2 depends on the field $\mathbb{F}_{q^k}$ (i.e., it is a function of the embedding degree). The crucial observation is that the field $\mathbb{F}_{q^k}$ is required to be the same size regardless of the genus. Hence, the group G_2 will be defined over a finite field of size independent of the genus. We therefore expect that operations in G_2 will be slower in the hyperelliptic case than in the elliptic case, unless one can exploit twists of high degree (e.g., degree $> 6g$).

The field $\mathbb{F}_{q^k}$ is the same size whether using elliptic or hyperelliptic curves. So the computation time in G_T is the same in both cases.

We now consider the cost of computing pairings. First we present an oversimplified analysis: the dominant part of a standard Tate-Lichtenbaum pairing computation (on either an elliptic or hyperelliptic curve) is $\log_2(r)$ iterations

of computing $h(D)$ for some function h and some divisor D (and accumulating the product of these values). As mentioned above, the field of definition of D depends on the embedding degree and so is similar in both cases. Furthermore, the functions h are more complicated for hyperelliptic curves than for elliptic curves. Hence, in general we expect pairing computation on hyperelliptic curves to be slower than elliptic curves.

The above analysis is extremely oversimplified and, indeed, is contradicted by the fact that the fastest known pairing computation is the η_T pairing on a supersingular genus 2 curve in characteristic 2 with embedding degree 12 (see the timings in Section 10 of [5]). The speed in that case is due to the implementation tricks available (including exploiting features of the processor architecture which are favourable to the genus 2 case). Also, the quoted timing is when using degenerate divisors; if general divisors are used then one gets faster timings using elliptic curves.

With hyperelliptic ate pairings over $\mathbb{F}_q$ one has a loop of length approximately $\log_2(q)$ in genus g rather than $g \log_2(q)$. This can be matched using the elliptic ate pairing if one can construct an elliptic curve over $\mathbb{F}_p$ (for $\log_2(p) \geq g \log_2(q)$) with trace of Frobenius $t \approx q$. The Brezing-Weng method [8] can be used to construct such curves. The example in Section 5.1 with $n = 1$ gives $\log_2(t) \approx \log_2(r)/4$, which matches the loop shortening obtained by using genus 4 curves. Hence, despite the attractive properties of the hyperelliptic ate pairing, it seems that in practice one can always match the speed of pairing computation by using elliptic curves.

10.2 Size of Representation

In general, we expect the size of the representation of G_1 to be the same for both elliptic and hyperelliptic curves (at least, as long as the ρ values are comparable in both cases). As noted earlier, degenerate divisors in G_1 require less storage than general elements, but there are potential security issues to take into account here. Also, as mentioned, the same effect can be achieved using elliptic curves by selecting points with short representations.

One issue here is comparing differing embedding degrees. For example, with supersingular curves in characteristic 2 we have $k = 4$ for elliptic curves and $k = 12$ for genus 2. One therefore would expect more compact representations in genus 2. However, the Rubin-Silverberg point compression method for elliptic curves can be applied. Hence, in practice, it seems that the size of elements of G_1 is always no worse for elliptic curves than hyperelliptic curves.

We now consider the size of elements of G_2. In the supersingular case, $G_1 = G_2$ and so the above remarks apply. If not, as mentioned above, the field of definition of G_2 depends on the embedding degree and so we expect the representation of G_2 to be larger in the hyperelliptic case than the elliptic case.

The representation of G_T is usually the same in both cases. Trace or torus methods can be used to compress these values [22,43,44].

10.3 Flexibility and Efficiency of Parameter Generation

The main concerns in this section are as follows. Are there pairing-friendly curves with useful parameters for secure and efficient implementation? Are there many possible examples or just a few? Is it easy to construct equations for such curves? Does the user have very fine control over the parameters?

If supersingular curves are used then there is roughly the same amount of flexibility in parameter generation for both elliptic and hyperelliptic curves.

There have been a number of results on constructing ordinary pairing-friendly elliptic curves using the CM method (see [13] for a survey). The outcome of this research is that there are many polynomial families of suitable curves. Further, one can generate curves with relatively small values of t, which are attractive due to the elliptic ate pairing [26]. So there is plenty of flexibility when choosing elliptic curves for pairing-based cryptography. In genus $g \geq 2$ the situation is much less satisfactory as was discussed in Section 5.2.

10.4 Special Properties

For pairings on elliptic curves there are great efficiency savings for computations in G_2 (including hashing to G_2) by using twists [26]. If non-supersingular hyperelliptic curves are to be competitive with ordinary elliptic curves then it is likely that similar techniques would have to be developed.

As noted in Section 8, in genus ≥ 2 there are larger torsion structures available and there is interesting pairing behaviour (see Lemma 2). It is natural to ask whether there might be novel cryptosystems which exploit this structure. Such applications would give renewed motivation for using hyperelliptic curves in pairing-based cryptography.

11 Conclusions and Open Problems

For non-pairing cryptography there are potential advantages of using hyperelliptic curves of genus $g \geq 2$. It is thus natural to consider hyperelliptic curves for pairing-based cryptography. We have surveyed work in this area. Our analysis indicates that, in practice, hyperelliptic curves are not more efficient than elliptic curves for general pairing applications. The only potentially significant advantage of hyperelliptic curves in pairing-based cryptosystems seems to be the speed of operations in G_1. Hence, hyperelliptic curves may be preferable for protocols with few pairing computations but many operations in G_1.

We conclude with a list of open problems.

- Can further loop shortening (as in [26,37]) be performed for the hyperelliptic ate pairing?
- A major problem is to give methods to construct non-supersingular pairing friendly curves of genus $g \geq 2$ and k in the range, say, $6g \leq k \leq 30g$. Ideally, these curves would have a single point at infinity and would have useful twists (as in [26]).

- Give fast methods to compute pairings on hyperelliptic curves with two points at infinity [21] or on non-hyperelliptic curves.
- Consider whether efficient and secure pairing-based cryptosystems can be developed for curves of genus $g \geq 3$, in spite of the index calculus attacks on curves in this case.
- Exploit the richer torsion structure available for abelian varieties. In particular, find cryptographic applications of pairings on groups which require 3 or more generators.
  A related problem is to give efficient methods to choose divisors in the particular subgroups.
- Improve the efficiency of the Rubin-Silverberg elliptic curve point decompression method. Generalise the Rubin-Silverberg method to divisor class groups of curves of genus $g \geq 2$.
- In Section 9 we recalled the identification of certain abelian varieties with subvarieties of the Weil restriction of supersingular curves. In the case where the abelian variety is a Jacobian, is there a way to compute explicit homomorphisms between the elliptic curve representation and the Jacobian representation?

Acknowledgements

We thank Tanja Lange, Eunjeong Lee, Jordi Pujolas, Mike Scott and Alice Silverberg for comments on a draft of this article.

The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The first author also thanks the EPSRC for support.

References

1. Avanzi, R., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., Vercauteren, F.: Handbook of elliptic and hyperelliptic curve cryptography. In: Discrete Mathematics and its Applications, Chapman & Hall/CRC, Sydney (2006)
2. 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)
3. 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)
4. Barreto, P.S.L.M., Lynn, B., Scott, M.: Efficient implementation of pairing-based cryptosystems. Journal of Cryptology 17(4), 321–334 (2004)
5. Barreto, P.S.L.M., Galbraith, S.D., ÓhÉigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular Abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007)
6. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)
7. Bernstein, D.: Elliptic vs hyperelliptic, part 1. Talk at ECC (2006)

8. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Designs, Codes and Cryptography 37, 133–141 (2005)
9. Cantor, D.G.: Computing in the Jacobian of a hyperelliptic curve. Math. Comp. 48(177), 95–101 (1987)
10. Choie, Y.-J., Lee, E.: Implementation of Tate pairing on hyperelliptic curves of genus 2. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 97–111. Springer, Heidelberg (2004)
11. Duursma, I., Lee, H.-S.: Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
12. Eisentraeger, K., Lauter, K., Montgomery, P.L.: Improved Weil and Tate pairings for elliptic and hyperelliptic curves. In: Buell, D.A. (ed.) Algorithmic Number Theory. LNCS, vol. 3076, pp. 169–183. Springer, Heidelberg (2004)
13. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372 (2006) Available from http://eprint.iacr.org/2006/372
14. Freeman, D.: Constructing pairing-friendly genus 2 curves over prime fields with ordinary Jacobians. In: proceedings of Pairing 2007, LNCS 4575, pp. 152–176, Springer, Heidelberg (to appear)
15. Frey, G., Lange, T.: Fast bilinear maps from the Tate-Lichtenbaum pairing on hyperelliptic curves. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. LNCS, vol. 4076, pp. 466–479. Springer, Heidelberg (2006)
16. Frey, G., Rück, H.-G.: A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. Math. Comp. 52, 865–874 (1994)
17. Galbraith, S.D., Harrison, K., Soldera, D.: Implementing the Tate pairing. In: Fieker, C., Kohel, D.R. (eds.) Algorithmic Number Theory. LNCS, vol. 2369, pp. 324–337. Springer, Heidelberg (2002)
18. Galbraith, S.D.: Supersingular curves in cryptography. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 495–513. Springer, Heidelberg (2001)
19. Galbraith, S.D., McKee, J.F., Valença, P.C.: Ordinary abelian varieties having small embedding degree. Finite Fields and Their Applications doi:10.1016/j.ffa.2007.02.003
20. Galbraith, S.D., Pujolàs, J., Ritzenthaler, C., Smith, B.: Distortion maps for genus two curves (2006) preprint: arXiv:math/0611471
21. Galbraith, S.D., Mireles, D.: Computing pairings on general genus 2 curves (preprint 2007)
22. Granger, R., Page, D., Stam, M.: On small characteristic algebraic tori in pairing-based cryptography. LMS Journal of Computation and Mathematics 9, 64–85 (2006)
23. Granger, R., Hess, F., Oyono, R., Thériault, N., Vercauteren, F.: ate pairing on hyperelliptic curves. In: Advances in Cryptology – EUROCRYPT 2007. LNCS, vol. 4515, pp. 419–436. Springer, Heidelberg (2007)
24. Granger, R., Page, D., Smart, N.P.: High security pairing-based cryptography revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. LNCS, vol. 4076, pp. 480–494. Springer, Heidelberg (2006)
25. Harley, R.: Fast arithmetic on genus 2 curves (2000) Available at http://cristal.inria.fr/harley/~hyper
26. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. IEEE Trans. Information Theory 52(10), 4595–4602 (2006)

27. Joux, A.: A one round protocol for tripartite Diffie-Hellman. Journal of Cryptology 17(4), 263–276 (2004)
28. Katagi, M., Akishita, T., Kitamura, I., Takagi, T.: Some improved algorithms for hyperelliptic curve cryptosystems using degenerate divisors. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 296–312. Springer, Heidelberg (2005)
29. Katagi, M., Kitamura, I., Akishita, T., Takagi, T.: Novel efficient implementations of hyperelliptic curve cryptosystems using degenerate divisors. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 345–359. Springer, Heidelberg (2005)
30. Koblitz, N.: Hyperelliptic cryptosystems. Journal of Cryptology 1(3), 139–150 (1989)
31. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)
32. Lange, T.: Formulae for arithmetic on genus 2 hyperelliptic curves. Appl. Algebra Eng. Commun. Comput. 15(5), 295–328 (2005)
33. Lange, T., Stevens, M.: Efficient doubling on genus two curves over binary fields. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 170–181. Springer, Heidelberg (2004)
34. Lange, T.: Elliptic vs. hyperelliptic, part 2. Talk at ECC 2006 (2006)
35. Lee, E., Lee, H.-S., Lee, Y.: Eta pairing computation on general divisors over hyperelliptic curves $y^2 = x^7 - x \pm 1$. In: proceedings of Pairing 2007, vol. 4575, pp. 349–366 Springer, Heidelberg (to appear)
36. Lichtenbaum, S.: Duality theorems for curves over p-adic fields. Invent. Math. 7, 120–136 (1969)
37. Matsuda, S., Kanayama, N., Hess, F., Okamoto, E.: Optimized versions of the ate and twisted ate pairings. Cryptology ePrint Archive, Report, 2007/013 (2007) Available from http://eprint.iacr.org/2007/013
38. Miller, V.S.: The Weil pairing and its efficient calculation. Journal of Cryptology 17(4), 235–261 (2004)
39. Mumford, D.: Abelian Varieties. Oxford University Press, London (1970)
40. Ó hÉigeartaigh, C., Scott, M.: Pairing calculation on supersingular genus 2 curves. In: Selected Areas in Cryptography 2006 (to appear)
41. Pujolas, J.: On the decisional Diffie-Hellman problem in genus 2, PhD thesis, Universitat Politècnica de Catalunya (2006)
42. Rubin, K., Silverberg, A.: Supersingular abelian varieties in cryptology. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 336–353. Springer, Heidelberg (2002)
43. Rubin, K., Silverberg, A.: Torus-based cryptography. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 349–365. Springer, Heidelberg (2003)
44. Rubin, K., Silverberg, A.: Using primitive subgroups to do more with fewer bits. In: Buell, D.A. (ed.) Algorithmic Number Theory. LNCS, vol. 3076, pp. 18–41. Springer, Heidelberg (2004)
45. Rubin, K., Silverberg, A.: Using abelian varieties to improve pairing based cryptography (Preprint, 2007)
46. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: The 2000 Symposium on Cryptography and Information Security (SCIS 2000) January 2000, Okinawa, Japan (2000)
47. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing over elliptic curve (in Japanese). In: The 2001 Symposium on Cryptography and Information Security, Oiso, Japan, January 2001 (2001)
48. Silverberg, A.: Compression for trace zero subgroups of elliptic curves. Trends in Mathematics 8, 93–100 (2005)

49. Tate, J.: WC group over p-adic fields. Séminaire Bourbaki (1958)
50. Verheul, E.: Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 195–210. Springer, Heidelberg (2001)
51. Verheul, E.: Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. Journal of Cryptology 17(4), 277–296 (2004)
52. Weil, A.: Sur les fonctions algebriques à corps de constantes finis. C. R. Acad. Sci. Paris, 210, 592–594 (1940) (= Oeuvres Scientifiques, vol. I, pp. 257–259)

Zeta Function and Cryptographic Exponent of Supersingular Curves of Genus 2

Gabriel Cardona[1] and Enric Nart[2,⋆]

[1] Dept. Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears,
E-07122 - Palma de Mallorca, Spain
gabriel.cardona@uib.es
[2] Departament de Matemàtiques, Universitat Autònoma de Barcelona,
E-08193 Bellaterra, Barcelona, Spain
nart@mat.uab.cat

Abstract. We compute in a direct (not algorithmic) way the zeta function of all supersingular curves of genus 2 over a finite field k, with many geometric automorphisms. We display these computations in an appendix where we select a family of representatives of all these curves up to $\overline{k}$-isomorphism and we exhibit equations and the zeta function of all their $\overline{k}/k$-twists. As an application we obtain a direct computation of the cryptographic exponent of the Jacobians of these curves.

Introduction

One-round tripartite Diffie-Hellman, identity based encryption, and short digital signatures are some problems for which good solutions have recently been found, making critical use of pairings on supersingular abelian varieties over a finite field k. The cryptographic exponent c_A of a supersingular abelian variety A is a half-integer that measures the security against an attack on the DL problem based on the Weil or the Tate pairings. Also, it is relevant to determine when pairings can be efficiently computed. Rubin and Silverberg showed in [RS04] that this invariant is determined by the zeta function of A.

In this paper we give a direct, non-algorithmic procedure to compute the zeta function of a supersingular curve of genus 2, providing thus a direct computation of the cryptographic exponent of its Jacobian. This is achieved in Sect. 1. For even characteristic the results are based on [MN06] and are summarized in Table 2; for odd characteristic we use results of Xing and Zhu on the structure of the group of k-rational points of a supersingular abelian surface and we almost determine the zeta function in terms of the Galois structure of the set of Weierstrass points of the curve (Tables 3, 4). In the rest of the paper we obtain a complete answer in the case of curves with many automorphisms. In Sect. 2 we study the extra information provided by these automorphisms and we show how to obtain the relevant data to compute the zeta funtion of a twisted curve in terms of data of the original curve

⋆ The authors acknowledge support from the projects MTM2006-15038-C02-01 and MTM2006-11391 from the Spanish MEC.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 132–151, 2007.

and the 1-cocycle defining the twist. In Sect. 3 we select a family of representatives of these curves up to $\overline{k}$-isomorphism and we apply the techniques of the previous section to deal with each curve and all its $\overline{k}/k$-twists. The results are displayed in an Appendix in the form of tables.

In what cryptographic applications of pairings concerns, curves with many automorphisms are interesting too because they are natural candidates to provide distortion maps on the Jacobian. In this regard the computation of the zeta function is a necessary step to study the structure of the endomorphism ring of the Jacobian (cf. [GPRS06]).

1 Zeta Function and Cryptographic Exponent

Let p be a prime number and let $k = \mathbb{F}_q$ be a finite field of characteristic p. We denote by k_n the extension of degree n of k in a fixed algebraic closure $\overline{k}$, $G_k := \mathrm{Gal}(\overline{k}/k)$ is the absolute Galois group of k, and $\sigma \in G_k$ the Frobenius automorphism.

Let C be a projective, smooth, geometrically irreducible, supersingular curve of genus 2 defined over k. The Jacobian J of C is a supersingular abelian surface over k (the p-torsion subgroup of $J(\overline{k})$ is trivial). Let us recall how supersingularity is reflected in a model of the curve C:

Theorem 1. *If p is odd, any curve of genus 2 defined over k admits an affine Weierstrass model $y^2 = f(x)$, with $f(x)$ a separable polynomial in $k[x]$ of degree 5 or 6. The curve is supersingular if and only if $M^{(p)}M = 0$, where M, $M^{(p)}$ are the matrices:*

$$M = \begin{pmatrix} c_{p-1} & c_{p-2} \\ c_{2p-1} & c_{2p-2} \end{pmatrix}, \quad M^{(p)} = \begin{pmatrix} c_{p-1}^p & c_{p-2}^p \\ c_{2p-1}^p & c_{2p-2}^p \end{pmatrix}, \quad f(x)^{(p-1)/2} = \sum_{j \geq 0} c_j x^j \ .$$

If $p = 2$ a curve of genus 2 defined over k is supersingular if and only if it admits an affine Artin-Schreier model $y^2 + y = f(x)$, with $f(x)$ an arbitrary polynomial in $k[x]$ of degree 5.

For the first statement see [Yui78] or [IKO86], for the second see [VV92].

For any simple supersingular abelian variety A defined over k, Rubin and Silverberg computed in [RS04] the *cryptographic exponent* c_A, defined as the half-integer such that q^{c_A} is the size of the smallest field F such that every cyclic subgroup of $A(k)$ can be embedded in F^*. This invariant refines the concept of *embedding degree*, formerly introduced as a measure of the security of the abelian variety against the attacks to the DLP by using the Weil pairing [MOV93] or the Tate pairing [FR94] (see for instance [Gal01]).

Let us recall the result of Rubin-Silverberg, adapted to the dimension two case. After classical results of Tate and Honda, the isogeny class of A is determined by the Weil polynomial of A, $f_A(x) = x^4 + rx^3 + sx^2 + qrx + q^2 \in \mathbb{Z}[x]$, which is the characteristic polynomial of the Frobenius endomorphism of the surface. For A supersingular the roots of $f_A(x)$ in $\overline{\mathbb{Q}}$ are of the form $\sqrt{q}\,\zeta$, where $\sqrt{q}$ is the positive square root of q and ζ is a primitive m-th root of unity.

Theorem 2. *Suppose A is a simple supersingular abelian surface over $\mathbb{F}_q$ and let $\ell > 5$ be any prime number dividing $|A(\mathbb{F}_q)|$. Then, the smallest half-integer c_A such that $q^{c_A} - 1$ is an integer divisible by ℓ is given by*

$$c_A = \begin{cases} m/2, & \text{if } q \text{ is a square,} \\ m/(2,m), & \text{if } q \text{ is not a square .} \end{cases}$$

In particular, the cryptographic exponent c_A is an invariant of the isogeny class of A. The complete list of simple supersingular isogeny classes of abelian surfaces can be found in [MN02, Thm. 2.9]. It is straightforward to find out the m-th root of unity in each case. We display the computation of c_A in Table 1.

Table 1. Cryptographic exponent c_A of the simple supersingular abelian surface A with Weil polynomial $f_A(x) = x^4 + rx^3 + sx^2 + qrx + q^2$

(r,s)	conditions on p and q	c_A
$(0,-2q)$	q nonsquare	1
$(0,2q)$	q square, $p \equiv 1 \pmod 4$	2
$(2\sqrt{q},3q)$	q square, $p \equiv 1 \pmod 3$	3/2
$(-2\sqrt{q},3q)$	q square, $p \equiv 1 \pmod 3$	3
$(0,0)$	(q nonsquare, $p \neq 2$) or (q square, $p \not\equiv 1 \pmod 8$)	4
$(0,q)$	q nonsquare	3
$(0,-q)$	(q nonsquare, $p \neq 3$) or (q square, $p \not\equiv 1 \pmod{12}$)	6
$(\sqrt{q},q)$	q square, $p \not\equiv 1 \pmod 5$	5/2
$(-\sqrt{q},q)$	q square, $p \not\equiv 1 \pmod 5$	5
$(\pm\sqrt{5q},3q)$	q nonsquare, $p = 5$	5
$(\pm\sqrt{2q},q)$	q nonsquare, $p = 2$	12

Therefore, the computation of the cryptographic exponent of the Jacobian J of a supersingular curve C amounts to the computation of the Weil polynomial of J, which is related in a well-known way to the zeta function of C. We shall call $f_J(x)$ the *Weil polynomial of* C too.

The computation of $f_J(x)$ has deserved a lot of attention because for the cryptographic applications one needs to know the cardinality $|J(\mathbb{F}_q)| = f_J(1)$ of the group of rational points of the Jacobian. However, in the supersingular case the current "counting points" algorithms are not necessary because there are more direct ways to compute the polynomial $f_J(x)$.

The aim of this section is to present these explicit methods, which take a different form for p odd or even. For $p = 2$ the computation of $f_J(x)$ is an immediate consequence of the methods of [MN06], based on ideas of van der Geer-van der Vlugt; for $p > 2$ we derive our results from the group structure of $J(\mathbb{F}_q)$, determined in [Xin96], [Zhu00], and from the exact knowledge of what isogeny classes of abelian surfaces do contain Jacobians [HNR06]. In both cases we shall show that $f_J(x)$ is almost determined by the structure as a Galois set of a finite subset of $\overline{k}$, easy to compute from the defining equation of C.

1.1 Computation of the Zeta Function When $p = 2$

We denote simply by tr the absolute trace $\mathrm{tr}_{k/\mathbb{F}_2}$. Recall that $\ker(\mathrm{tr}) = \{x + x^2 \mid x \in k\}$ is an $\mathbb{F}_2$-linear subspace of k of codimension 1.

Every projective smooth geometrically irreducible supersingular curve C of genus 2 defined over k admits an affine Artin-Schreier model of the type:

$$C: \quad y^2 + y = ax^5 + bx^3 + cx + d, \quad a \in k^*,\ b,\ c,\ d \in k,$$

which has only one point at infinity [VV92]. The change of variables $y = y+u$, $u \in k$, allows us to suppose that $d = 0$ or $d = d_0$, with $d_0 \in k \setminus \ker(\mathrm{tr})$ fixed. Twisting C by the hyperelliptic twist consists in adding d_0 to the defining equation. If we denote by J' the Jacobian of the twisted curve we have $f_{J'}(x) = f_J(-x)$. Thus, for the computation of $f_J(x)$ we can assume that $d = 0$.

The structure as a G_k-set of the set of roots in $\overline{k}$ of the polynomial $P(x) = a^2x^5 + b^2x + a \in k[x]$ almost determines the zeta function of C [MN06, Sect.3].

Table 2. Weil polynomial $x^4+rx^3+sx^2+qrx+q^2$ of the curve $y^2+y = ax^5+bx^3+cx$, for q nonsquare (left) and q square (right)

$P(x)$	$N,\ M$	(r, s)
$(1)(4)$	$N = 0$	$(\pm\sqrt{2q}, 2q)$
	$N = 1$	$(0, 0)$
$(2)(3)$	$M = 0$	$(\pm\sqrt{2q}, q)$
	$M = 1$	$(0, q)$
$(1)^3(2)$	$N = 0$	$(\pm 2\sqrt{2q}, 4q)$
	$N = 1$	$(0, 2q)$
	$N = 2$	$(0, 0)$
	$N = 3$	$(0, -2q)$

$P(x)$	$N,\ M$	(r, s)
(5)		$(\pm\sqrt{q}, q)$
$(1)^2(3)$	$N = 0$	$(0, -q)$
	$N = 1$	$(0, q)$
	$N = 2$	$(\pm 2\sqrt{q}, 3q)$
$(1)(2)^2$	$M = 0$	$(\pm 2\sqrt{q}, 2q)$
	$M = 1$	$(0, 0)$
	$M = 2$	$(0, 2q)$
$(1)^5$	$N = 1$	$(0, -2q)$
	$N = 3$	$(0, 2q)$
	$N = 5$	$(\pm 4\sqrt{q}, 6q)$

In Table 2 we write $P(x) = (n_1)^{r_1}(n_2)^{r_2}\cdots(n_m)^{r_m}$ to indicate that r_i of the irreducible factors of $P(x)$ have degree n_i. Also, we consider the linear operator $T(x) := \mathrm{tr}((c + b^2a^{-1})x)$ and we define

$N :=$ number of roots $z \in k$ of $P(x)$ s.t. $T(z) = 0$,
$M :=$ number of irred. quadratic factors $x^2 + vx + w$ of $P(x)$ s.t. $T(v) = 0$.

The ambiguity of the sign of r can be solved by computing nD in the Jacobian, where n is one of the presumed values of $|J(\mathbb{F}_q)|$ and D is a random rational divisor of degree 0.

1.2 Computation of the Zeta Function When p Is Odd

Let A be a supersingular abelian surface over k and let $\mathrm{rk}_2(A) := \dim_{\mathbb{F}_2}(A[2](k))$.

The structure of $A(k)$ as an abelian group was studied in [Xin96], [Zhu00], where it is proven that it is almost determined by the isogeny class of A. In fact, if $F_i(x)$ are the different irreducible factors of $f_A(x)$ in $\mathbb{Z}[x]$:

$$f_A(x) = \prod_{i=1}^{s} F_i(x)^{e_i},\ 1 \leq s \leq 2 \quad \Longrightarrow \quad A(k) \simeq \oplus_{i=1}^{s} \left(\mathbb{Z}/F_i(1)\mathbb{Z}\right)^{e_i},$$

except for the following cases:

(a) $p \equiv 3 \pmod 4$, q is not a square and $f_A(x) = (x^2+q)^2$,
(b) $p \equiv 1 \pmod 4$, q is not a square and $f_A(x) = (x^2-q)^2$.
(c) q is a square and $f_A(x) = (x^2-q)^2$.

The possible structure of $A(k)$ in cases (a) and (b) is:

$$A(k) \simeq \left(\mathbb{Z}/F(1)\mathbb{Z}\right)^m \oplus \left(\mathbb{Z}/(F(1)/2)\mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}\right)^n,$$

where $F(x)$ denotes respectively x^2+q, x^2-q, and m, n are non-negative integers such that $m+n=2$ [Zhu00, Thm. 1.1]. In case (c) we have either:

$$A(k) \simeq \left(\mathbb{Z}/((q-1)/2)\mathbb{Z}\right)^2 \oplus \left(\mathbb{Z}/2\mathbb{Z}\right)^2, \qquad \text{or}$$
$$A(k) \simeq \left(\mathbb{Z}/((q-1)/2^m)\mathbb{Z}\right) \oplus \left(\mathbb{Z}/((q-1)/2^n)\mathbb{Z}\right) \oplus \left(\mathbb{Z}/2^{m+n}\mathbb{Z}\right),$$

where $0 \leq m,\ n \leq v_2(q-1)$ [Xin96, Thm. 3]. In this last case we have $\mathrm{rk}_2(A) > 1$; in fact, $v_2(1-\sqrt{q}) + v_2(1+\sqrt{q}) = v_2(1-q) = (1/2)v_2(F(1))$ and we can apply [Xin96, Lem. 4] to conclude that $A(k)$ has a subgroup isomorphic to $(\mathbb{Z}/2\mathbb{Z})^2$.

Consider now a supersingular curve C of genus 2 defined over k, given by a Weierstrass equation $y^2 = f(x)$, for some separable polynomial $f(x) \in k[x]$ of degree 5 or 6. Let J be its Jacobian variety, $W = \{P_0, P_1, P_2, P_3, P_4, P_5\} \subseteq C(\overline{k})$ the set of Weierstrass points of C, and $W(k) \subseteq W$ the subset of k-rational Weierstrass points. Our aim is to show that the structure of W as a G_k-set contains enough information on the 2-adic value of $|C(k)|$ and $|J(k)|$ to almost determine the polynomial $f_J(x) = x^4 + rx^3 + sx^2 + qrx + q^2$.

From the fundamental identities

$$|C(k)| = q + 1 + r, \qquad |J(k)| = f_J(1) = (q^2+1) + (q+1)r + s,$$

and the free action of the hyperelliptic involution on $C(k) \setminus W(k)$ we get

$$r \equiv |W(k)| \pmod 2, \qquad s \equiv |J(k)| \pmod 2\ . \tag{1}$$

On the other hand, $J[2]$ is represented by the classes of the 15 divisors:

$$P_i - P_0,\ \ 1 \leq i \leq 5, \qquad \text{and} \qquad P_i + P_j - 2P_0,\ \ 1 \leq i < j \leq 5,$$

together with the trivial class.

Lemma 3. *Let $D = P_i - P_j$, with $i \neq j$, or $D = P_i + P_j - 2P_0$, with $0, i, j$ pairwise different. Then, the class of the divisor D is k-rational if and only if P_i, P_j are both k-rational or quadratic conjugate.*

Hence, the Galois structure of W determines $\mathrm{rk}_2(J)$ and this limits the possible values of the zeta function of C. Our final results are given in Tables 3, 4, where we write $W = (n_1)^{r_1}(n_2)^{r_2}\cdots(n_m)^{r_m}$ to indicate that there are r_i G_k-orbits of length n_i of Weierstrass points. If $f(x)$ has degree 6 this Galois structure mimics the decomposition $f(x) = (n_1)^{r_1}(n_2)^{r_2}\cdots(n_m)^{r_m}$ (same notation as in Sect. 1.1) of $f(x)$ into a product of irreducible polynomials $k[x]$. If $f(x)$ has degree 5 then $W = (1)f(x)$, because in these models the point at infinity is a k-rational Weierstrass point.

Table 3. Weil polynomial $x^4+rx^3+sx^2+qrx+q^2$ of the curve C when q is nonsquare. The sign ϵ is the Legendre symbol $(-1/p)$.

W	p	$\mathrm{rk}_2(J)$	(r,s)
$(1)^6$ or $(1)^4(2)$		4, 3	$(0,-2\epsilon q)$
$(1)^2(2)^2$ or $(2)^3$		2	$(0,\pm 2q)$
$(1)^3(3)$		2	not possible
$(1)(2)(3)$	$p>3$	1	not possible
	$p=3$		$(\pm\sqrt{3q},2q)$
$(1)^2(4)$ or $(2)(4)$		1	$(0,0)$
$(1)(5)$	$p\neq 5$	0	not possible
	$p=5$		$(\pm\sqrt{5q},3q)$
$(3)^2$	$p\equiv 1 \pmod 3$	0	$(0,q)$
	$p\equiv -1 \pmod 3$		$(0,\epsilon q)$
	$p=3$		not possible
(6)	$p\equiv -1 \pmod 3$	0	$(0,\pm q)$
	$p\not\equiv -1 \pmod 3$		$(0,q)$

The proof of the content of Tables 3 and 4 is elementary, but long. Instead of giving all details we only sketch the main ideas:

(I) Waterhouse determined all possible isogeny classes of supersingular elliptic curves [Wat69]. Thus, it is possible to write down all isogeny classes of supersingular abelian surfaces by adding to the simple classes given in Table 1 the split isogeny classes. By [HNR06] we know exactly what isogeny classes of abelian surfaces do not contain Jacobians and they can be dropped from the list. By the results of Xing and Zhu we can distribute the remaining isogeny classes according to the possible values of rk_2.

(II) Each structure of W as a G_k-set determines the value of rk_2 and, after (I), it has a reduced number of possibilities for the isogeny classes. By using (1) and looking for some incoherence in the behaviour under scalar extension to k_2 or k_3 of both, the Galois structure of W and the possible associated isogeny classes, we can still discard some of these possibilities.

In practice, among the few possibilities left in Tables 3 and 4 we can single out the isogeny class of the Jacobian of any given supersingular curve by computing iterates of random divisors of degree zero. However, if C has many automorphisms they provide enough extra information to completely determine the zeta

Table 4. Weil polynomial $x^4 + rx^3 + sx^2 + qrx + q^2$ of the curve C when q is a square

W	p	$\mathrm{rk}_2(J)$	(r,s)
$(1)^6$		4	$(0,-2q)$ or $(\pm 4\sqrt{q}, 6q)$
$(1)^4(2)$		3	$(0,-2q)$
$(1)^2(2)^2$ or $(2)^3$		2	$(0,\pm 2q)$
$(1)^3(3)$	$p>3$	2	not possible
	$p=3$		$(\pm\sqrt{q},0)$
$(1)(2)(3)$		1	not possible
$(1)^2(4)$ or $(2)(4)$	$p \equiv 1 \pmod 8$	1	not possible
	$p \not\equiv 1 \pmod 8$		$(0,0)$
$(1)(5)$	$p \equiv 1 \pmod 5$	0	not possible
	$p \not\equiv 1 \pmod 5$		$(\pm\sqrt{q}, q)$
$(3)^2$			$(0,q)$ or $(\pm 2\sqrt{q}, 3q)$
(6)	$p \equiv 5 \pmod{12}$	0	$(0,\pm q)$
	$p \not\equiv 5 \pmod{12}$		$(0,q)$

function. This will be carried out in the rest of the paper. In the Appendix we display equations of the supersingular curves with many automorphisms and their Weil polynomial.

2 Zeta Function of Twists

In this section we review some basic facts about twists and we show how to compute different properties of a twisted curve in terms of the defining 1-cocycle. From now on the ground field k will have odd characteristic.

Let C be a supersingular curve of genus 2 defined over k and let $W \subseteq C(\overline{k})$ be the set of Weierstrass points of C. We denote by $\mathrm{Aut}(C)$ the k-automorphism group of C and by $\mathrm{Aut}_{\overline{k}}(C)$ the full automorphism group of C.

Let $\phi\colon C \longrightarrow \mathbb{P}^1$ be a fixed k-morphism of degree 2 and consider the group of reduced geometric automorphisms of C:

$$\mathrm{Aut}'_{\overline{k}}(C) := \{u' \in \mathrm{Aut}_{\overline{k}}(\mathbb{P}^1) \mid u'(\phi(W)) = \phi(W)\} \ .$$

We denote by $\mathrm{Aut}'(C)$ the subgroup of reduced automorphisms defined over k.

Any automorphism u of C fits into a commutative diagram:

$$\begin{array}{ccc} C & \xrightarrow{u} & C \\ \downarrow{\scriptstyle \phi} & & \downarrow{\scriptstyle \phi} \\ \mathbb{P}^1 & \xrightarrow{u'} & \mathbb{P}^1 \end{array}$$

for certain uniquely determined reduced automorphism u'. The map $u \mapsto u'$ is a group homomorphism (depending on ϕ) and we have a central exact sequence of groups compatible with Galois action:

$$1 \longrightarrow \{1, \iota\} \longrightarrow \mathrm{Aut}_{\overline{k}}(C) \xrightarrow{\phi} \mathrm{Aut}'_{\overline{k}}(C) \longrightarrow 1,$$

where ι is the hyperelliptic involution. This leads to a long exact sequence of Galois cohomology sets:

$$1 \to \{1,\iota\} \to \mathrm{Aut}(C) \xrightarrow{\phi} \mathrm{Aut}'(C) \xrightarrow{\delta} H^1(G_k,\{1,\iota\}) \to H^1(G_k,\mathrm{Aut}_{\overline{k}}(C)) \to \\ \to H^1(G_k,\mathrm{Aut}'_{\overline{k}}(C)) \to H^2(G_k,\{1,\iota\}) \simeq \mathrm{Br}_2(k) = 0 \ . \quad (2)$$

The $\overline{k}/k$-twists of C are parameterized by the pointed set $H^1(G_k,\mathrm{Aut}_{\overline{k}}(C))$ and, since k is a finite field, a 1-cocycle is determined just by the choice of an automorphism $v \in \mathrm{Aut}_{\overline{k}}(C)$. The twisted curve C_v associated to v is defined over k and is determined, up to k-isomorphism, by the existence of a $\overline{k}$-isomorphism $f\colon C \longrightarrow C_v$, such that $f^{-1}f^\sigma = v$.

For instance, the choice $v = \iota$ corresponds to the hyperelliptic twist C'; if C is given by a Weierstrass equation $y^2 = f(x)$ then C' admits the model $y^2 = tf(x)$, for $t \in k^* \backslash (k^*)^2$. We say that C is *self-dual* if it is k-isomorphic to its hyperelliptc twist. If $f_J(x)$ is the Weil polynomial of C, the Weil polynomial of C' is $f_{J'}(x) = f_J(-x)$; in particular, for a self-dual curve one has $f_J(x) = x^4 + sx^2 + q^2$ for some integer s.

It is easy to deduce from (2) the following criterion for self-duality:

Lemma 4. *The curve C is self-dual if and only if $|\mathrm{Aut}'(C)| = |\mathrm{Aut}(C)|$.*

One can easily compute the data $\mathrm{Aut}(C_v)$, $\mathrm{Aut}'(C_v)$ of the twisted curve C_v, in terms of $\mathrm{Aut}_{\overline{k}}(C)$, $\mathrm{Aut}'_{\overline{k}}(C)$ and the 1-cocycle v.

Let $f\colon C \longrightarrow C_v$ be a geometric isomorphism such that $f^{-1}f^\sigma = v$. We have $\mathrm{Aut}_{\overline{k}}(C_v) = f\,\mathrm{Aut}_{\overline{k}}(C)f^{-1}$, and the k-automorphism group is

$$\mathrm{Aut}(C_v) = \{fuf^{-1} \mid u \in \mathrm{Aut}_{\overline{k}}(C),\, u\,v = v\,u^\sigma\} \ . \quad (3)$$

Once we fix any k-morphism of degree two, $\phi_v\colon C_v \longrightarrow \mathbb{P}^1$, it determines a unique geometric automorphism f' of $\mathbb{P}^1$ such that $\phi_v f = f'\phi$. The reduced group of k-automorphisms of C_v is

$$\mathrm{Aut}'(C_v) = \{f'u'(f')^{-1} \mid u' \in \mathrm{Aut}'_{\overline{k}}(C),\, u'\,v' = v'(u')^\sigma\} \ . \quad (4)$$

In order to compute the zeta function of C_v we consider the geometric isomorphism $f\colon J \longrightarrow J_v$ induced by f. We still have $f^{-1}f^\sigma = v_*$, where v_* is the automorphism of J induced by v. Clearly, $\pi_v f = f^\sigma \pi$, where π, π_v are the respective q-power Frobenius endomorphisms of J, J_v. Hence,

$$f^{-1}\pi_v f = f^{-1}f^\sigma (f^\sigma)^{-1}\pi_v f = v_*\pi \ .$$

In particular, π_v has the same characteristic polynomial than $v_*\pi$. From this fact one can deduce two crucial results (cf. [HNR06, Props.13.1,13.4]).

Proposition 5. *Suppose q is a square. Let C be a supersingular genus 2 curve over k with Weil polynomial $(x+\sqrt{q})^4$ and let v be a geometric automorphism of*

C, $v \neq 1, \iota$. Then, the Weil polynomial $x^4+rx^3+sx^2+rx+q^2$ of C_v is determined as follows in terms of v (in the column $v^6=1$ we suppose $v^2 \neq 1$, $v^3 \neq 1, \iota$):

v	$v^2=1$	$v^2=\iota$	$v^3=1$	$v^3=\iota$	$v^4=\iota$	$v^5=1$	$v^5=\iota$	$v^6=1$	$v^6=\iota$
(r,s)	$(0,-2q)$	$(0,2q)$	$(-2\sqrt{q},3q)$	$(2\sqrt{q},3q)$	$(0,0)$	$(-\sqrt{q},q)$	$(\sqrt{q},q)$	$(0,q)$	$(0,-q)$

Proposition 6. *Suppose q is nonsquare. Let C be a supersingular genus 2 curve over k with Weil polynomial $(x^2+\epsilon q)^2$, $\epsilon \in \{1,-1\}$, and let v be a geometric automorphism of C. Then, the Weil polynomial $x^4+rx^3+sx^2+rx+q^2$ of C_v is determined as follows in terms of the order n of the automorphism vv^σ:*

n	1	2	3	4	6
(r,s)	$(0,2\epsilon q)$	$(0,-2\epsilon q)$	$(0,-\epsilon q)$	$(0,0)$	$(0,\epsilon q)$

In applying these results the transitivity property of twists can be helpful.

Lemma 7. *Let u, v be automorphisms of C and let $f\colon C \to C_v$ be a geometric isomorphism with $f^{-1}f^\sigma = v$. Then the curve C_u is the twist of C_v associated to the automorphism $fuv^{-1}f^{-1}$ of C_v.*

For a curve with a large k-automorphism group the following remark, together with Tables 3 and 4, determines in some cases the zeta function:

Lemma 8. *Let $\mathcal{F} \subseteq C(k)$ be the subset of k-rational points of C that are fixed by some non-trivial k-automorphism of C. Then,*

$$|C(k)| \equiv |\mathcal{F}| \pmod{|\operatorname{Aut}(C)|} \ .$$

Proof. The group $\operatorname{Aut}(C)$ acts freely on $C(k) \setminus \mathcal{F}$.

Note that $\mathcal{F}$ contains the set $W(k)$ of k-rational Weierstrass points, all of them fixed by the hyperelliptic involution ι of C.

In order to apply this result to the twisted curve C_v we need to compute the G_k-set structure of W_v and $|\mathcal{F}_v|$ solely in terms of v.

Lemma 9. *(1) For any $P \in W$ the length of the G_k-orbit of $f(P) \in W_v$ is the minimum positive integer n such that $v\, v^\sigma \cdots v^{\sigma^{n-1}}(P^{\sigma^n}) = P$. In particular, $|W_v(k)| = |\{P \in W \mid v(P^\sigma) = P\}|$.*

(2) The map f^{-1} stablishes a bijection between $\mathcal{F}_v$ and the set $\{P \in C(\overline{k}) \mid v(P^\sigma) = P = u(P)$ for some $1 \neq u \in \operatorname{Aut}_{\overline{k}}(C)$, s.t. $u\, v = v\, u^\sigma\}$.

3 Supersingular Curves with Many Automorphisms

For several cryptographic applications of the Tate pairing the use of distortion maps is essential. A distortion map is an endomorphism ψ of the Jacobian J of C that provides an input for which the value of the pairing is non-trivial: $e_\ell(D_1, \psi(D_2)) \neq 1$ for some fixed ℓ-torsion divisors D_1, D_2. The existence of such a map is guaranteed, but in practice it is hard to find it in an efficient way. Usually, one can start with a nice curve C with many automorphisms, consider

a concrete automorphism $u \neq 1$, $u \neq \iota$, and look for a distortion map ψ in the subring $\mathbb{Z}[\pi, u_*] \subseteq \mathrm{End}(J)$, where π is the Frobenius endomorphism of J and u_* is the automorphism of the Jacobian induced by u. If $\mathbb{Z}[\pi, u_*] = \mathrm{End}(J)$ it is highly probable that a distortion map is found. If $\mathbb{Z}[\pi, u_*] \neq \mathrm{End}(J)$ it can be a hard problem to prove that some nice candidate is a distortion map, but at least one is able most of the time to find a "denominator" m such that $m\psi$ lies in the subring $\mathbb{Z}[\pi, u_*]$; in this case, if $\ell \nmid m$ one can use $m\psi$ as a distortion map on divisors of order ℓ. Several examples are discussed in [GPRS06].

The aim of this section is to exhibit all supersingular curves of genus 2 with many automorphisms, describe their automorphisms, and compute the characteristic polynomial of π, which is always a necessary ingredient in order to analyze the structure of the ring $\mathbb{Z}[\pi, u_*]$. Recall that a curve C is said to have *many automorphisms* if it has some geometric automorphism other than the identity and the hyperelliptic involution; in other words, if $|\mathrm{Aut}_{\overline{k}}(C)| > 2$.

Igusa found equations for all geometric curves of genus 2 with many automorphisms, and he grouped these curves in six families according to the possible structure of the automorphism group [Igu60], [IKO86]. Cardona and Quer found a faithful and complete system of representatives of all these curves up to $\overline{k}$-isomorphism and they gave conditions to ensure the exact structure of the automorphism group of each concrete model [Car03], [CQ06]. The following theorem sums up these results.

Theorem 10. *Any curve of genus* 2 *with many automorphisms is geometrically isomorphic to one and only one of the curves in these six families:*

Equation of C		$\mathrm{Aut}'_{\overline{k}}(C)$	$\mathrm{Aut}_{\overline{k}}(C)$
$y^2 = x^6 + ax^4 + bx^2 + 1$	a, b *satisfy* (5)	C_2	$C_2 \times C_2$
$y^2 = x^5 + x^3 + ax$	$a \neq 0,\ 1/4,\ 9/100$	$C_2 \times C_2$	D_8
$y^2 = x^6 + x^3 + a$	$p \neq 3,\ a \neq 0,\ 1/4,\ -1/50$	S_3	D_{12}
$y^2 = ax^6 + x^4 + x^2 + 1$	$p = 3,\ a \neq 0$	S_3	D_{12}
$y^2 = x^6 - 1$	$p \neq 3,\ 5$	D_{12}	$2D_{12}$
$y^2 = x^5 - x$	$p \neq 5$	S_4	$\tilde{S}_4$
	$p = 5$	$\mathrm{PGL}_2(\mathbb{F}_5)$	$\tilde{S}_5$
$y^2 = x^5 - 1$	$p \neq 5$	C_5	C_{10}

$$(4c^3 - d^2)(c^2 - 4d + 18c - 27)(c^2 - 4d - 110c + 1125) \neq 0,\ c := ab,\ d := a^3 + b^3. \quad (5)$$

Ibukiyama-Katsura-Oort determined, using Theorem 1, when the last three curves are supersingular [IKO86, Props. 1.11, 1.12, 1.13]:

$$\begin{aligned} &y^2 = x^6 - 1 \text{ is supersingular iff } \ p \equiv -1 \pmod 3 \\ &y^2 = x^5 - x \text{ is supersingular iff } \ p \equiv 5, 7 \pmod 8 \\ &y^2 = x^5 - 1 \text{ is supersingular iff } \ p \equiv 2, 3, 4 \pmod 5 \end{aligned}$$

It is immediate to check that $y^2 = ax^6 + x^4 + x^2 + 1$ is never supersingular if $p = 3$. One can apply Theorem 1 to the other curves in the first three families to distinguish the supersingular ones.

Theorem 11. *Suppose q is a square and let C be a supersingular curve belonging to one of the first five families of Theorem 10. Then there a twist of C with Weil polynomial $(x+\sqrt{q})^4$, and this twist is unique.*

Proof. Let E be a supersingular elliptic curve defined over $\mathbb{F}_p$. By [IKO86, Prop. 1.3] the Jacobian J of C is geometrically isomorphic to the product of two supersingular elliptic curves, which is in turn isomorphic to $E\times E$ by a well-known theorem of Deligne. The principally polarized surface (J,Θ) is thus geometrically isomorphic to $(E\times E,\lambda)$ for some principal polarization λ. Since E has all endomorphisms defined over $\mathbb{F}_{p^2}$, $(E\times E,\lambda)$ is defined over $\mathbb{F}_{p^2}$ and by a classical result of Weil it is $\mathbb{F}_{p^2}$-isomorphic to the canonically polarized Jacobian of a curve C_0 defined over $\mathbb{F}_{p^2}$. By Torelli, C_0 is a twist of C. The Weil polynomial of C_0 is $(x\pm\sqrt{q})^4$ because the Frobenius polynomial of E is x^2+p. The fact that C_0 and C_0' are the unique twists of C_0 with Weil polynomial $(x\pm\sqrt{q})^4$ is consequence of Proposition 5. □

Corollary 12. *Under the same assumptions:*

1. *The Weil polynomial of C is $(x\pm\sqrt{q})^4$ if and only if $W=(1)^6$.*
2. *If C belongs to one of the first three families of Theorem 10, then it admits no twist with Weil polynomial $x^4\pm qx^2+q^2$ or x^4+q^2.*
3. *If any of the curves $y^2=x^5+x^3+ax$, $y^2=x^6+x^3+a$ is supersingular then $a\in\mathbb{F}_{p^2}$.*

Proof. (1) By Table 4, the set W_0 of Weierstrass points of C_0 has G_k-structure $W_0=(1)^6$ and Lemma 9 shows that for all automorphisms $v\neq 1,\iota$ one has $W_v\neq(1)^6$; thus, only the twists C_0 and C_0' have $W=(1)^6$.

(2) The geometric automorphisms v of C_0 satisfy neither $v^6=1$, $v^2\neq 1$, $v^3\neq 1,\iota$, nor $v^6=\iota$, nor $v^4=\iota$; thus, by Proposition 5 the Weil polynomial of a twist of C_0 is neither $x^4\pm qx^2+q^2$ nor x^4+q^2.

(3) The Igusa invariants of C_0 take values in $\mathbb{F}_{p^2}$ and a can be expressed in terms of these invariants [CQ05]. □

In a series of papers Cardona and Quer studied the possible structures of the pointed sets $H^1(G_k,\mathrm{Aut}_{\overline{k}}(C))$ and found representatives $v\in\mathrm{Aut}_{\overline{k}}(C)$ (identified to 1-cocycles of $H^1(G_k,\mathrm{Aut}_{\overline{k}}(C))$) of the twists of all curves with many automorphisms [Car03], [CQ05], [Car06], [CQ06]. In the next subsections we compute the zeta function and the number of k-automorphisms of these curves when they are supersingular. A general strategy that works in most of the cases is to apply the techniques of Sect. 2 to find a twist of C with Weil polynomial $(x\pm\sqrt{q})^4$ (for q square) or $(x^2\pm q)^2$ (for q nonsquare) and apply then Propositions 5, 6 to obtain the zeta function of all other twists of C. The results are displayed in the Appendix in the form of Tables, where we exhibit moreover an equation of each curve.

3.1 Twists of the Curve $C\colon y^2=x^5-1$, for $p\not\equiv 0,1 \pmod 5$

We have $\phi(W)=\{\infty\}\cup\mu_5$ and $\mathrm{Aut}'_{\overline{k}}(C)\simeq\mu_5$. The zeta function of C can be computed from Tables 3,4 and Lemma 8 applied to $C\otimes k_2$. If $q\not\equiv 1 \pmod 5$ the

only twists are C, C'. If $q \equiv 1 \pmod 5$ there are ten twists and their zeta function can be deduced from Proposition 5. Table 5 summarizes all computations.

3.2 Twists of the Curve $C: y^2 = x^5 - x$, for $p \equiv 5, 7 \pmod 8$

Now $\phi(W) = \{\infty, 0, \pm 1, \pm i\}$. If $p = 5$ we have $\mathrm{Aut}'_{\overline{k}}(C) = \mathrm{Aut}(\mathbb{P}^1)$. If $p \neq 5$ the group $\mathrm{Aut}'_{\overline{k}}(C)$ is isomorphic to S_4 and it is generated by the transformations $T(x) = ix, S(x) = \frac{x-i}{x+i}$, with relations $S^3 = 1 = T^4, ST^3 = TS^2$. For q nonsquare the zeta function of C is determined by Table 3; since the curve is defined over $\mathbb{F}_p$ we obtain the zeta function of C over k by scalar extension.

In all cases we can apply Propositions 5 and 6 to determine the zeta function of the twists of C. Tables 6, 7, 8 summarize all computations.

3.3 Twists of the Curve $C: y^2 = x^6 - 1$, for $p \equiv -1 \pmod 3$, $p \neq 5$

We have $\phi(W) = \mu_6$ and $\mathrm{Aut}'_{\overline{k}}(C) = \{\pm x, \pm \eta x, \pm \eta^2 x, \pm \frac{1}{x}, \pm \frac{\eta}{x}, \pm \frac{\eta^2}{x}\}$, where $\eta \in \mathbb{F}_{p^2}$ is a primitive third root of unity.

The zeta function of C can be computed from Tables 3,4 and Lemma 8 applied to C and $C \otimes k_2$. The zeta function of all twists can be determined by Propositions 5, 6. Tables 9, 10 summarize all computations.

3.4 Twists of the Supersingular Curve $C: y^2 = x^6 + x^3 + a$, for $p > 3$

Recall that a is a special value making the curve C supersingular and $a \neq 0, 1/4, -1/50$. We have now

$$\phi(W) = \{\theta, \eta\theta, \eta^2\theta, \frac{A}{\theta}, \eta\frac{A}{\theta}, \eta^2\frac{A}{\theta}\}, \quad \mathrm{Aut}'_{\overline{k}}(C) = \{x, \eta x, \eta^2 x, \frac{A}{x}, \eta\frac{A}{x}, \eta^2\frac{A}{x}\},$$

where $A,\ z,\ \theta \in \overline{k}$ satisfy $A^3 = a,\ z^2 + z + a = 0,\ \theta^3 = z$.

The Galois action on W and on $\mathrm{Aut}'_{\overline{k}}(C)$ depends on z and a/z being cubes or not in their minimum field of definition k^* or $(k_2)^*$. This is determined by the fact that a is a cube or not.

Lemma 13. *If a is a cube in k^* then z, a/z are both cubes in k^* or in $(k_2)^*$, according to $1 - 4a \in (k^*)^2$ being a square or not.*

If a is not a cube in k^* then z, a/z are both noncubes in k^* or in $(k_2)^*$, according to $1 - 4a \in (k^*)^2$ being a square or not.

Proof. Let us check that all situations excluded by the statement lead to $W = (1)^3(3)$ or $W_v = (1)^3(3)$ for some twist, in contradiction with Tables 3, 4.

Suppose $q \equiv -1 \pmod 3$. If $1-4a$ is a square then a, z, a/z are all cubes in k^*. If $1 - 4a$ is not a square then a is a cube and if z, z^σ are not cubes in k_2 we have $\theta^\sigma = \omega(A/\theta)$, with $\omega^3 = 1$, $\omega \neq 1$, and the twist by $v = (\omega^{-1}(A/x), \sqrt{a}y/x^3)$ has $W_v = (1)^3(3)$ by Lemma 9.

Suppose $q \equiv 1 \pmod 3$. If $1 - 4a$ is not a square we have $z^{(q^2-1)/3} = a^{(q-1)/3}$, so that a is a cube in k^* if and only if z, z^σ are cubes in k_2^*. Suppose now that

$1-4a$ is a square. If exactly one of the two elements z, a/z is a cube we have $W=(1)^3(3)$; thus z, a/z are both cubes or noncubes in k^*. In particular, if a is not a cube then z, a/z are necessarily both noncubes. Finally, if a is a cube and z, a/z are noncubes in k^*, Lemma 9 shows that $W_v=(1)^3(3)$ for the twist corresponding to $v=(\eta x,y)$. □

For the computation of the zeta functions of the twists it is useful to detect that some of the combinations a square/nonsquare and $1-4a$ square/nonsquare are not possible.

Lemma 14. *Suppose* $q\equiv 1 \pmod 3$.

1. *If* $q\equiv -1 \pmod 4$ *then* $1-4a$ *is not a square.*
2. *If* q *is nonsquare then* a *is not a square.*
3. *If* q *is a square then* a *and* $1-4a$ *are both squares.*

Proof. Let C_v be the twist of C corresponding to $v(x,y)=(\eta x,y)$.

(1) Supose $1-4a$ is a square. If a is a cube we have $W=(1)^6$ and if a is not a cube we have $W_v=(1)^6$; by Table 3 we get $(r,s)=(0,-2\left(\frac{-1}{p}\right)q)$ in both cases. On the other hand, Lemmas 8 and 9 applied to $C\otimes_k k_2$ show in both cases that $s\equiv 1 \pmod 3$; thus, $p\equiv 1 \pmod 4$.

(2) Suppose a is a square. If a is a cube (respectively a is not a cube) we have $W=(1)^6$ or $W=(2)^3$ (respectively $W_v=(1)^6$ or $W_v=(2)^3$), according to $1-4a$ being a square or not. In all cases we have $(r,s)=(0,\pm 2q)$ by Table 3, and a straightforward application of Lemma 8 and (2) of Lemma 9 leads to $r\equiv -1 \pmod 3$, which is a contradiction.

(3) In all cases in which a or $1-4a$ are nonsquares we get $(r,s)=(0,q)$ either for the curve C or for the curve C_v. This contradicts Corollary 12. □

After these results one can apply the general strategy. The results are displayed in Tables 11, 12, 13.

3.5 Twists of the Supersingular Curve $C\colon y^2=x^5+x^3+ax$

Recall that a is a special value making C supersingular and $a\neq 0,\ 1/4,\ 9/100$. Given $z\in\overline{k}$ satisfying $z^2+z+a=0$ we have $\phi(W)=\{0,\ \infty,\ \pm\sqrt{z},\ \pm\sqrt{a/z}\}$,

$$\mathrm{Aut}_{\overline{k}}(C)=\left\{(\omega^2 x,\omega y)\mid \omega^4=1\right\}\cup\left\{\left(\frac{w^2}{x},\frac{w^3 y}{x^3}\right)\mid w^4=a\right\}\ .$$

Lemma 15. *If* $q\equiv 1 \pmod 4$ *then* a *and* $1-4a$ *are both squares or both nonsquares in* k^*. *If* q *is a square then necessarily* a *and* $1-4a$ *are both squares.*

Proof. If $a\notin (k^*)^2$, $1-4a\in (k^*)^2$, then $W=(1)^4(2)$ and $(r,s)=(0,-2q)$ by Tables 3,4; this contradicts Lemma 8 because $|\mathrm{Aut}(C)|=|\mathcal{F}|=4$ and $r\equiv 2 \pmod 4$.

Suppose now $a\in (k^*)^2$, $1-4a\notin (k^*)^2$. If $a\in (k^*)^4$ then $W=(1)^2(2)^2$ and $(r,s)=(0,\pm 2q)$; this contradicts Lemma 8 because $|\mathrm{Aut}(C)|=8$, $|\mathcal{F}|=6$ if

$q \equiv 1 \pmod 8$ and $|\mathcal{F}| = 2$ or 10 if $q \equiv 5 \pmod 8$, so that $r \equiv 4 \pmod 8$ in both cases. If $a \notin (k^*)^4$ we get a similar contradiction for the curve C_v for $v(x,y) = (-x, iy)$.

If a, $1-4a$ are nonsquares, then $W = (1)^2(4)$ and the Weil polynomial of C is $x^4 + q^2$ by Tables 3,4. If q is a square this contradicts Corollary 12. □

Lemma 16. *If q is a square then $a \in (k^*)^4$ if and only if $z \in (k^*)^2$.*

Proof. Suppose $a \in (k^*)^4$, $z \notin (k^*)^2$ and let us look for a contradiction. Consider the k-automorphisms $u(x,y) = (-x, iy)$, $v(x,y) = (\frac{w^2}{x}, \frac{w^3}{x^3}y)$ of C, where $w^4 = a$. By Lemma 9, $W_u = (1)^6$ and C_u has Weil polynomial $(x \pm \sqrt{q})^4$ by Corollary 12; since $u^2 = \iota$, the Weil polynomial of C is $(x^2+q)^2$ by Proposition 5 and Lemma 7. The quotient $E := C/v$ is an elliptic curve defined over k and the Frobenius endomorphism π of E must satisfy $\pi^2 = -q$. Since q is a square, E has four automorphisms and its j invariant is necessarily $j_E = 1728$. Now, E has a Weierstrass equation: $Y^2 = (X+2w)(X^2+1-2w^2)$, where $X = (x^2+w^2)x^{-1}$, $Y = y(x+w)x^{-2}$ are invariant under the action of v. The condition $j_E = 1728$ is equivalent to $a = 0$ (which was excluded from the beginning) or $a = (9/14)^2$; in this latter case z is a square in $\mathbb{F}_{p^2}$ and we get a contradiction.

Suppose now $a \notin (k^*)^4$, $z \in (k^*)^2$. We have $W = (1)^6$ and C has Weil polynomial $(x \pm \sqrt{q})^4$ by Corollary 12. By Proposition 5, the Weil polynomial of C_u is $(x^2+q)^2$. For any choice of $w = \sqrt[4]{a}$, the morphism $f(x,y) = (\frac{x+w}{x-w}, \frac{8\sqrt{w^3}}{\sqrt{1+2w^2}}\frac{y}{(x-w)^3})$ sets a k_2-isomorphism between C and the model:

$$C_u\colon y^2 = (x^2-1)(x^4+bx^2+1), \qquad b = (12\sqrt{a}-2)/(2\sqrt{a}+1),$$

of C_u. The quotient of this curve by the automorphism $(-x, y)$ is the elliptic curve $E\colon Y^2 = (X-1)(X^2+bX+1)$. Arguing as above, E has j-invariant 1728, and this leads to $a = 0$ (excluded from the beginning) or $a = (9/14)^2$, which is a contradiction since a would be a fourth power in $\mathbb{F}_{p^2}$. □

After these results one is able to determine the zeta function of all twists of C when q is a square; the results are displayed in Table 14. In the cases where the Weil polynomial is $(x - \epsilon\sqrt{q})^4$, $\epsilon = \pm 1$, the methods of section 2 are not sufficient to determine ϵ; our computation of this sign follows from a study of the 4-torsion of an elliptic quotient of the corresponding curve.

In order to deal with the case q nonsquare we need to discard more cases.

Lemma 17. *Suppose q nonsquare. If $q \equiv -1 \pmod 4$ then a and $1-4a$ cannot be both nonsquares.*

If $q \equiv 1 \pmod 4$ and $a \in (k^*)^2$ then $a \in (k^*)^4$ if and only if $z \notin (k^*)^2$.

Proof. If a, $1-4a$ are both nonsquares the polynomial x^4+x^2+a is irreducible and the Weil polynomial of C is x^4+q^2 by Table 3; hence, the Weil polynomial of $C \otimes_k k_2$ is $(x^2+q^2)^2$. If $q \equiv -1 \pmod 4$ we have $a \in k^* \subseteq (k_2^*)^4$ and this contradicts Table 14.

Suppose $q \equiv 1 \pmod 4$ and $a \in (k^*)^2$; by Lemma 15, $1-4a$ is also a square and $z \in k^*$. If $a \in (k^*)^4$ and $z \in (k^*)^2$ we get $W = (1)^6$, and $(r,s) = (0,-2q)$ by Table 3; we get a contradiction because the Jacobian J of C is simple ([MN02, Thm. 2.9]) and C has elliptic quotients over k because the automorphisms $(w^2/x, (w^3y)/x^3)$ are defined over k. If $a \notin (k^*)^4$ and $z \notin (k^*)^2$ we get an analogous contradiction for the curve C_u twisted by $u(x,y) = (-x, iy)$. □

The results for the case q nonsquare follow now by the usual arguments and they are displayed in Tables 15, 16.

3.6 Twists of the Supersingular Curve $C\colon y^2 = x^6 + ax^4 + bx^2 + 1$

Recall that $a, b \in k$ are special values satisfying (5) and making C supersingular; in particular $p > 3$. The curve C has four twists because $\mathrm{Aut}_{\overline{k}}(C) = \mathrm{Aut}(C) = \{(\pm x, \pm y)\}$ is commutative and has trivial Galois action. The Jacobian of C is k-isogenous to the product $E_1 \times E_2$ of the elliptic curves with Weierstrass equations $y^2 = x^3 + ax^2 + bx + 1$, $y^2 = x^3 + bx^2 + ax + 1$, obtained as the quotient of C by the respective automorphisms $\upsilon = (-x, y)$, $\iota\upsilon = (-x, -y)$. For q nonsquare, these elliptic curves have necessarily Weil polynomial $x^2 + q$ and the Weil polynomial of C is $(x^2+q)^2$.

Lemma 18. *If q is a square C has Weil polynomial $(x \pm \sqrt{q})^4$.*

Proof. By Theorem 11 and Proposition 5 C has Weil polynomial $(x \pm \sqrt{q})^4$ or $(x^2-q)^2$. In both cases the elliptic curves E_1, E_2 have Weil polynomial $(x \pm \sqrt{q})^2$ and we claim that they are isogenous. Since $E(k) \simeq (\mathbb{Z}/(1 \pm \sqrt{q})\mathbb{Z})^2$ as an abelian group, our elliptic curves have four rational 2-torsion points and the polynomial $x^3 + ax^2 + bx + 1$ has three roots e_1, e_2, $e_3 \in k$. Since $e_1e_2e_3 = 1$, either one or three of these roots are squares. If only one root is a square we have $W = (1)^2(2)^2$, $W_\upsilon = (1)^4(2)$ and C, C_υ have both Weil polynomial $(x^2 \pm q)^2$, in contradiction with Theorem 11. Hence, the three roots are squares, $W = (1)^6$, and C has Weil polynomial $(x \pm \sqrt{q})^4$ by Corollary 12. □

The zeta function of the twists of C is obtained from Propositions 5 and 6. The results are displayed in Table 17. For q square the sign of $(x \pm \sqrt{q})^4$ can be determined by analyzing the 4-torsion of the elliptic curve $y^2 = x^3 + ax^2 + bx + 1$.

Finally, there are special curves over k whose geometric model $y^2 = x^6 + ax^4 + bx^2 + 1$ is not defined over k (cf. [Car03, Sect.1]). It is straightforward to apply the techniques of this paper to determine their zeta function too.

4 Conclusion

We show that the zeta function of a supersingular curve of genus two is almost determined by the Galois structure of a finite set easy to describe in terms of a

defining equation. For curves with many automorphisms this result is refined to obtain a direct (non-algoritmic) computation of the zeta function in all cases. As an application one gets a direct computation of the cryptographic exponent of the Jacobian of these curves. Also, the computation of the zeta function is necessary to determine the structure of the endomorphism ring of the Jacobian and to compute distortion maps for the Weil and Tate pairings.

Acknowledgement. It is a pleasure to thank Christophe Ritzenthaler for his help in finding some of the equations of the twisted curves.

References

[Car03] Cardona, G.: On the number of curves of genus 2 over a finite field. Finite Fields and Their Applications 9, 505–526 (2003)

[CQ05] Cardona, G., Quer, J.: Field of moduli and field of definition for curves of genus 2. In: Shaska, T. (ed.) Computational aspects of algebraic curves, Lecture Notes Series on Computing 13, pp. 71–83, World Scientific

[Car06] Cardona, G.: Representations of G_k-groups and the genus 2 curve $y^2 = x^5 - x$. Journal of Algebra 303, 707–721 (2006)

[CQ06] Cardona, G., Quer, J.: Curves of genus 2 with group of automorphisms isomorphic to D_8 or D_{12}, Trans. Amer. Math. Soc. (to appear)

[FR94] Frey, G., Rück, H.-G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Mathematics of Computation 62, 865–874 (1994)

[Gal01] Galbraith, S.D.: Supersingular curves in cryptography. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 495–513. Springer, Heidelberg (2001)

[GPRS06] Galbraith, S.D., Pujolàs, J., Ritzenthaler, C., Smith, B.: Distortion maps for genus two curves `http://eprint.iacr.org/2006/375`

[HNR06] Howe, E.W., Nart, E., Ritzenthaler, C.: Jacobians in isogeny classes of abelian surfaces over finite fields `arXiv:math.NT/0607515`

[IKO86] Ibukiyama, T., Katsura, T., Oort, F.: Supersingular curves of genus two and class numbers. Compositio Math. 57, 127–152 (1986)

[Igu60] Igusa, J.-I.: Arithmetic variety of moduli for genus two. Annals of Mathematics 72, 612–649 (1960)

[MN02] Maisner, D., Nart, E., Howe, E.W.: Abelian surfaces over finite fields as jacobians. Experimental Mathematics 11, 321–337 (2002)

[MN06] Maisner, D., Nart, E.: Zeta functions of supersingular curves of genus 2. Canadian Journal of Mathematics 59, 372–392 (2007)

[MOV93] Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. IEEE Trans. on Information Theory 39, 1639–1646 (1993)

[RS04] Rubin, K., Siverberg, A.: Supersingular abelian varieties in cryptology. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 336–353. Springer, Heidelberg (2004)

[VV92] van der Geer, G., van der Vlugt, M.: Supersingular curves of genus 2 over finite fields of characteristic 2. Math. Nachrichten 159, 73–81 (1992)

[Wat69] Waterhouse, W.C.: Abelian varieties over finite fields. Annales Scientifiques de l'École Normale Supérieure 2(4), 521–560 (1969)
[Xin96] Xing, C.P.: On supersingular abelian varieties of dimension two over finite fields. Finite Fields and Their Applications 2, 407–421 (1996)
[Yui78] Yui, N.: On the Jacobian varieties of hyperelliptic curves over fields of characteristic $p > 2$. Journal of Algebra 52, 378–410 (1978)
[Zhu00] Zhu, H.J.: Group Structures of Elementary Supersingular Abelian Varieties over Finite Fields. Journal of Number Theory 81, 292–309 (2000)

Appendix

In this appendix we display in several tables the computation of the zeta function of the supersingular curves of genus 2 with many automorphisms. For each curve C_v, we exhibit the number of k-automorphisms and the pair of integers (r, s) determining the Weil polynomial $f_{J_v}(x) = x^4 + rx^3 + sx^2 + qrx + q^2$ of C_v. In the column labelled "s.d" we indicate if C is self-dual. For the non-self-dual curves we exhibit only one curve from the pair C_v, C'_v.

We denote by η, $i \in \overline{k}$ a primitive third, fourth root of unity. For n a positive integer and $x \in k^*$ we define

$$\nu_n(x) = 1 \ \text{ if } x \in (k^*)^n, \qquad \nu_n(x) = -1 \ \text{ otherwise } .$$

In all tables the parameters s, t take values in k^*.

Table 5. Twists of the curve $y^2 = x^5 - 1$ for $p \equiv 2, 3, 4 \pmod 5$. The sign $\epsilon = \pm 1$ is determined by $\sqrt{q} \equiv \epsilon \pmod 5$. The last row provides eight inequivalent twists corresponding to the four nontrivial values of $t \in k^*/(k^*)^5$.

C_v	v		(r, s)	s.d.	$\lvert \mathrm{Aut}(C_v) \rvert$
$y^2 = x^5 - 1$	(x, y)	$q \equiv \pm 2 \pmod 5$	$(0, 0)$	no	2
		$q \equiv -1 \pmod 5$	$(0, 2q)$		2
		$q \equiv 1 \pmod 5$	$(-4\epsilon\sqrt{q}, 6q)$		10
$y^2 = tx^5 - 1$, $t \notin (k^*)^5$	$(t^{\frac{1-q}{5}}x, y)$	$q \equiv 1 \pmod 5$	$(\epsilon\sqrt{q}, q)$	no	10

Table 6. Twists of the curve $y^2 = x^5 - x$ when $q \equiv -1 \pmod 8$

C_v	v	(r, s)	s.d.	$\lvert \mathrm{Aut}(C_v) \rvert$
$y^2 = x^5 - x$	(x, y)	$(0, 2q)$	yes	8
$y^2 = x^5 + x$	$(ix, \frac{1+i}{\sqrt{2}}y)$	$(0, 2q)$	yes	4
$y^2 = (x^2+1)(x^2 - 2tx - 1)(x^2 + \frac{2}{t}x - 1)$ $t^2 + 1 \notin (k^*)^2$	$(-\frac{1}{x}, \frac{y}{x^3})$	$(0, -2q)$	yes	24
$y^2 = (x^2+1)(x^4 - 4tx^3 - 6x^2 + 4tx + 1)$, $t^2 + 1 \notin (k^*)^2$	$\left(\frac{i}{x}, \frac{i-1}{\sqrt{2}x^3}y\right)$	$(0, 0)$	yes	4
$y^2 = x^6 - (t+3)x^5 + 5(\frac{2+t-s}{2})x^4 + 5(s-1)x^3$ $+5(\frac{2-t-s}{2})x^2 + (t-3)x + 1$ irred., $s^2 + t^2 = -2$	$\left(\frac{x-i}{x+i}, \frac{2(1-i)y}{(x+i)^3}\right)$	$(0, q)$	no	6

Table 7. Twists of the curve $y^2 = x^5 - x$ when $q \equiv 5 \pmod 8$

C_v	v	p	(r,s)	s.d.	$\lvert \mathrm{Aut}(C_v)\rvert$
$y^2 = x^5 - x$	(x,y)	$p>5$ $p=5$	$(0,-2q)$	yes	24 120
$y^2 = x^5 - 4x$	$(-x, iy)$		$(0,2q)$	yes	8
$y^2 = x^5 - 2x$	$(ix, \frac{1+i}{\sqrt{2}}y)$		$(0,0)$	yes	4
$y^2 = (x^2+2)(x^4-12x^2+4)$	$\left(\frac{i}{x}, \frac{i-1}{\sqrt{2}x^3}y\right)$	$p>5$ $p=5$	$(0,2q)$	yes	4 12
$y^2 = f(t,x)f(\frac{18+(5i-3)t}{(5i+3)-2t}, x)$ $f(t,x) = x^3 - tx^2 + (t-3)x + 1$ irred.	$\left(\frac{x-i}{x+i}, \frac{2(1-i)y}{(x+i)^3}\right)$	$p>5$ $p=5$	$(0,q)$	no yes	6
$y^2 = x^5 - x - t$, $\mathrm{tr}_{k/\mathbb{F}_5}(t) = 1$	$(x+1, y)$	$p=5$	$(\sqrt{5q}, 3q)$	no	10
$y^2 = x^6 + tx^5 + (1-t)x + 2$, irred.	$(\frac{3}{x-1}, \frac{\sqrt{2}y}{(x+1)^3})$	$p=5$	$(0,-q)$	yes	6

Table 8. Twists of the curve $y^2 = x^5 - x$ when $p \equiv 5,\ 7 \pmod 8$ and q is a square. Here $\epsilon = (-1/\sqrt{q})$ and $\epsilon' = (-3/\sqrt{q})$.

C_v	v	p	(r,s)	s.d.	$\lvert \mathrm{Aut}(C_v)\rvert$
$y^2 = x^5 - x$	(x,y)	$p>5$ $p=5$	$(-4\epsilon\sqrt{q}, 6q)$	no	48 240
$y^2 = x^5 - t^2x$, $t \notin (k^*)^2$	$(-x, iy)$		$(0,2q)$	yes	8
$y^2 = x^5 - tx$, $t \notin (k^*)^2$	$(ix, \frac{1+i}{\sqrt{2}}y)$		$(0,0)$	no	8
$y^2 = (x^2-t)(x^4+6tx^2+t^2)$, $t \notin (k^*)^2$	$\left(\frac{i}{x}, \frac{i-1}{\sqrt{2}x^3}y\right)$	$p>5$ $p=5$	$(0,-2q)$	yes	4 12
$y^2 = (x^3-t)(x^3-(15\sqrt{3}-26)t)$, $t \notin (k^*)^3$	$\left(\frac{x-i}{x+i}, \frac{2(1-i)y}{(x+i)^3}\right)$	$p>5$ $p=5$	$(2\epsilon'\sqrt{q}, 3q)$	no	6 12
$y^2 = x^5 - x - t$, $\mathrm{tr}_{k/\mathbb{F}_5}(t) = 1$	$(x+1, y)$	$p=5$	$(\sqrt{q}, q)$	no	10
$y^2 = x^6 + tx^5 + (1-t)x + 2$, irred.	$(\frac{3}{x-1}, \frac{\sqrt{2}y}{(x+1)^3})$	$p=5$	$(0,q)$	no	12

Table 9. Twists of the curve $y^2 = x^6 - 1$ when $q \equiv -1 \pmod 3$, $p \neq 5$. Here $\epsilon = (-1/p)$.

C_v	v	(r,s)	s.d.	$\lvert \mathrm{Aut}(C_v)\rvert$
$y^2 = x^6 - 1$	(x,y)	$(0,2q)$	iff $\epsilon = -1$	$6+2\epsilon$
$y^2 = x^6 - t$, $t \notin (k^*)^2$	$(-x,-y)$	$(0,2q)$	iff $\epsilon = 1$	$6-2\epsilon$
$y^2 = x(x^2-1)(x^2-9)$	$(\frac{1}{x}, \frac{iy}{x^3})$	$(0,-2\epsilon q)$	yes	12
$y^2 = (x^4 - 2stx^3 + (7s+1)x^2 + 2tsx + 1)\cdot$ $\cdot(x^2 - \frac{4}{t}x - 1)$, $t^2+4 \in k^* \setminus (k^*)^2$, $s^{-1} = t^2+3$	$(-\frac{1}{x}, \frac{iy}{x^3})$	$(0,2\epsilon q)$	yes	12
$y^2 = x^6 + 6tx^5 + 15sx^4 + 20tsx^3 + 15s^2x^2 +$ $+6ts^2x + s^3$, $s = t^2 - 4 \notin (k^*)^2$, $\gcd(x^{(q+1)/3} - 1, x^2 - tx + 1) = 1$	$(\frac{\eta}{x}, \frac{iy}{x^3})$	$(0,\epsilon q)$	yes	6
$y^2 = x^6 + 6x^5 + 15sx^4 + 20sx^3 + 15s^2x^2 +$ $+6s^2x + s^3$, $s = t^2/(t^2+4) \notin (k^*)^2$, $\gcd(x^{(q+1)/3} + 1, x^2 - tx - 1) = 1$	$(-\frac{\eta}{x}, \frac{iy}{x^3})$	$(0,-\epsilon q)$	yes	6

Table 10. Twists of the curve $y^2 = x^6 - 1$ when $p \equiv -1 \pmod 3$, $p \neq 5$ and q is a square. Here $\epsilon = (-3/\sqrt{q})$.

C_v	v	(r,s)	s.d.	$\lvert \mathrm{Aut}(C_v)\rvert$
$y^2 = x^6 - 1$	(x,y)	$(-4\epsilon\sqrt{q}, 6q)$	no	24
$y^2 = x^6 - t^3$, $\quad t \notin (k^*)^2$	$(-x,y)$	$(0,-2q)$	yes	12
$y^2 = x^6 - t^2$, $\quad t \notin (k^*)^3$	$(\eta x, y)$	$(2\epsilon\sqrt{q}, 3q)$	no	12
$y^2 = x^6 - t$, $\quad t \notin ((k^*)^2 \cup (k^*)^3)$	$(-\eta x, -y)$	$(0,q)$	no	12
$y^2 = x(x^2+3t)(x^2+\frac{t}{3})$, $\quad t \notin (k^*)^2$	$(\frac{1}{x}, \frac{iy}{x^3})$	$(0,2q)$	yes	4
$y^2 = x^6 + 15tx^4 + 15t^2x^2 + t^3$, $\quad t \notin (k^*)^2$	$(-\frac{1}{x}, \frac{iy}{x^3})$	$(0,-2q)$	yes	4

Table 11. Twists of the supersingular curve $y^2 = x^6 + x^3 + a$, $a \neq 0, 1/4, -1/50$, when $q \equiv -1 \pmod 3$. Here $\epsilon = \nu_2(a)$ and A is the cubic root of a in k.

C_v	v	(r,s)	s.d.	$\lvert \mathrm{Aut}(C_v)\rvert$
$y^2 = x^6 + x^3 + a$	(x,y)	$(0,2q)$	iff $\epsilon = -1$	$3+\epsilon$
$y^2 = \theta^{-3}(x-\theta)^6 - g(x)^3 + a\theta^3(x-\theta^\sigma)^6$ $g(x)$ min. polyn. of $\theta \in k_2 \setminus k$, $\mathrm{N}_{k_2/k}(\theta) = A^{-1}$	$(\frac{A}{x}, \frac{\sqrt{a}}{x^3}y)$	$(0,2\epsilon q)$	iff $\epsilon = -1$	$9+3\epsilon$
$y^2 = \theta(x-\eta)^6 - g(x)^3 + a\theta^{-1}(x-\eta^2)^6$ $g(x) = x^2+x+1$, $\theta \in k_2 \setminus (k_2^*)^3$, $\mathrm{N}_{k_2/k}(\theta) = a$	$(\eta\frac{A}{x}, \frac{\sqrt{a}}{x^3}y)$	$(0,-\epsilon q)$	no	6

Table 12. Twists of the supersingular curve $y^2 = x^6 + x^3 + a$, $a \neq 0, 1/4, -1/50$, when $q \equiv 1 \pmod 3$ and q is nonsquare. Here A is a cubic root of a in k and $n = 3$, if $a \in (k^*)^3$, whereas $A = a$, $n = 1$, if $a \notin (k^*)^3$.

C_v	v	$\nu_3(a)$	(r,s)	s.d.	$\lvert \mathrm{Aut}(C_v)\rvert$
$y^2 = x^6 + x^3 + a$	(x,y)	1 -1	$(0,-2q)$ $(0,q)$	yes no	6
$y^2 = x^6 + tx^3 + t^2a$, $\ t \notin (k^*)^3$ $y^2 = x^6 + ax^3 + a^3$	$(\eta x, y)$	1 -1	$(0,q)$ $(0,-2q)$	no yes	6
$y^2 = \theta^{-n}(x-\theta)^6 - g(x)^3 + a\theta^n(x-\theta^\sigma)^6$ $g(x)$ min. polyn. of $\theta \in k_2 \setminus k$, $\mathrm{N}_{k_2/k}(\theta) = A^{-1}$	$(\frac{\sqrt[3]{a}}{x}, \frac{\sqrt{a}}{x^3}y)$		$(0,2q)$	yes	2

Table 13. Twists of the supersingular curve $y^2 = x^6 + x^3 + a$, $a \neq 0, 1/4, -1/50$, when q is a square. Here $\epsilon = (-3/\sqrt{q})$. Also, A is a cubic root of a in k and $n = 3$, if $a \in (k^*)^3$, whereas $A = a$, $n = 1$, if $a \notin (k^*)^3$.

C_v	v	$\nu_3(a)$	(r,s)	s.d.	$\lvert \mathrm{Aut}(C_v)\rvert$
$y^2 = x^6 + x^3 + a$	(x,y)	1 -1	$(-4\epsilon\sqrt{q}, 6q)$ $(2\epsilon\sqrt{q}, 3q)$	no	12 6
$y^2 = x^6 + tx^3 + t^2a$, $\ t \notin (k^*)^3$ $y^2 = x^6 + ax^3 + a^3$	$(\eta x, y)$	1 -1	$(2\epsilon\sqrt{q}, 3q)$ $(-4\epsilon\sqrt{q}, 6q)$	no	6 12
$y^2 = \theta^{-n}(x-\theta)^6 - g(x)^3 + a\theta^n(x-\theta^\sigma)^6$ $g(x)$ min. polyn. of $\theta \in k_2 \setminus k$, $\mathrm{N}_{k_2/k}(\theta) = A^{-1}$	$(\frac{\sqrt[3]{a}}{x}, \frac{\sqrt{a}}{x^3}y)$		$(0,-2q)$	no	4

Table 14. Twists of the supersingular curve $y^2 = x^5 + x^3 + ax$, $a \neq 0, 1/4, 9/100$, when q is a square. The last row provides two inequivalent twists according to the two values of $\sqrt{a}$. Here $\epsilon = -(-1/\sqrt{q})\nu_4(z)$ and $\epsilon' = -(-1/\sqrt{q})\nu_4(tz)$, where $z^2 + z + a = 0$.

C_v	v	$\nu_4(a)$	(r, s)	s.d.	$\lvert \mathrm{Aut}(C_v) \rvert$
$y^2 = x^5 + x^3 + ax$	(x, y)	1	$(4\epsilon\sqrt{q}, 6q)$	no	8
		-1	$(0, 2q)$	yes	4
$y^2 = x^5 + tx^3 + at^2x, \quad t \notin (k^*)^2$	$(-x, t^{\frac{q-1}{4}}y)$	1	$(0, 2q)$	yes	4
		-1	$(4\epsilon'\sqrt{q}, 6q)$	no	8
$y^2 = g(x)\left(\theta^2(x-\theta^\sigma)^4 + g(x)^2 + a\theta^{-2}(x-\theta)^4\right)$, $\mathrm{N}_{k_2/k}(\theta) = \sqrt{a}$ $g(x)$ min. polyn. of $\theta \in k_2 \setminus k$	$(\frac{\sqrt{a}}{x}, \frac{\sqrt[4]{a^3}}{x^3}y)$		$(0, -2q)$	yes	4

Table 15. Twists of the supersingular curve $y^2 = x^5 + x^3 + ax$, $a \neq 0, 1/4, 9/100$, when q is nonsquare and $a \notin (k^*)^2$

C_v	v	$(-1/p)$	(r, s)	s.d.	$\lvert \mathrm{Aut}(C_v) \rvert$
$y^2 = x^5 + x^3 + ax$	(x, y)	1	$(0, 0)$	no	4
		-1	$(0, 2q)$	yes	2
$y^2 = (x^2 - a)\left(\theta(x - \sqrt{a})^4 + (x^2 - a)^2 + a\theta^{-1}(x + \sqrt{a})^4\right)$, $\theta \in k_2$, $\mathrm{N}_{k_2/k}(\theta) = a$	$(\frac{\sqrt{a}}{x}, \frac{\sqrt[4]{a^3}}{x^3}y)$	1	$(0, 2q)$	yes	2
		-1	$(0, 0)$	no	4

Table 16. Twists of the supersingular curve $y^2 = x^5 + x^3 + ax$, $a \neq 0, 1/4, 9/100$, when q is nonsquare and $a \in (k^*)^2$. Here $\epsilon = (-1/p)$. If $p \equiv -1 \pmod 4$ we assume that $\sqrt{a}$ belongs to $(k^*)^2$.

C_v	v	$\nu_4(a)$	(r, s)	s.d.	$\lvert \mathrm{Aut}(C_v) \rvert$
$y^2 = x^5 + x^3 + ax$	(x, y)	1	$(0, 2q)$	iff $\epsilon = -1$	$6 + 2\epsilon$
		-1	$(0, -2q)$	yes	4
$y^2 = x^5 + tx^3 + at^2x, \quad t \notin (k^*)^2$	$(-x, t^{\frac{q-1}{4}}y)$	1	$(0, -2\epsilon q)$	yes	4
		-1	$(0, 2q)$	no	8
$y^2 = g(x)\left(\theta^2(x-\theta^\sigma)^4 + g(x)^2 + a\theta^{-2}(x-\theta)^4\right)$, $\mathrm{N}_{k_2/k}(\theta) = \sqrt{a}$ $g(x)$ min. polyn. of $\theta \in k_2 \setminus k$	$(\frac{\sqrt{a}}{x}, \frac{\sqrt[4]{a^3}}{x^3}y)$		$(0, 2q)$	iff $\epsilon = 1$	$6 - 2\epsilon$
$y^2 = g(x)\left(\theta^2(x-\theta^\sigma)^4 + g(x)^2 + a\theta^{-2}(x-\theta)^4\right)$, $\mathrm{N}_{k_2/k}(\theta) = -\sqrt{a}$ $g(x)$ min. polyn. of $\theta \in k_2 \setminus k$	$(\frac{-\sqrt{a}}{x}, \frac{i\sqrt[4]{a^3}}{x^3}y)$		$(0, 2\epsilon q)$	yes	4

Table 17. Twists of the supersingular curve $y^2 = x^6 + ax^4 + bx^2 + 1$ satisfying (5)

C_v	v		(r, s)	s.d.	$\lvert \mathrm{Aut}(C_v) \rvert$
$y^2 = x^6 + ax^4 + bx^2 + 1$	(x, y)	q nonsq.	$(0, 2q)$	no	4
		q square	$(\pm 4\sqrt{q}, 6q)$		
$y^2 = x^6 + atx^4 + bt^2x^2 + t^3$ $t \notin (k^*)^2$	$(-x, -y)$	q nonsq.	$(0, 2q)$	no	4
		q square	$(0, -2q)$		

Constructing Pairing-Friendly Genus 2 Curves with Ordinary Jacobians

David Freeman

Department of Mathematics
University of California, Berkeley
Berkeley, CA 94720-3840
USA
dfreeman@math.berkeley.edu

Abstract. We provide the first explicit construction of genus 2 curves over finite fields whose Jacobians are ordinary, have large prime-order subgroups, and have small embedding degree. Our algorithm is modeled on the Cocks-Pinch method for constructing pairing-friendly elliptic curves [5], and works for arbitrary embedding degrees k and prime subgroup orders r. The resulting abelian surfaces are defined over prime fields $\mathbb{F}_q$ with $q \approx r^4$. We also provide an algorithm for constructing genus 2 curves over prime fields $\mathbb{F}_q$ with ordinary Jacobians J having the property that $J[r] \subset J(\mathbb{F}_q)$ or $J[r] \subset J(\mathbb{F}_{q^k})$ for any even k.

1 Introduction

In the last few years, many cryptographic protocols have been proposed that make use of bilinear pairings [25]. While the protocols are described in the language of abstract groups, in practice the pairings used are the Weil and Tate pairings on abelian varieties over finite fields. These pairings take as input two points on an abelian variety A defined over a finite field $\mathbb{F}_q$ and give as output an element of an extension field $\mathbb{F}_{q^k}$. The degree k of the extension field is known as the "embedding degree" of the abelian variety.

For pairing-based cryptosystems to be both efficient and secure, the embedding degree k should be chosen so that the discrete logarithm problem is equally difficult on the abelian variety (where only exponential-time discrete logarithm algorithms are known) and in the multiplicative group of the extension field (where there exist subexponential-time discrete logarithm algorithms). Since the optimal embedding degree will vary according to the desired level of security, in order to build systems with a variety of security levels we wish to have a supply of abelian varieties with various embedding degrees.

While elliptic curves remain the most common choice of a family of abelian varieties for cryptographic protocols, Bernstein [2] and Lange [18] have recently shown that for certain applications Jacobians of genus 2 curves are now competitive with elliptic curves in terms of performance and security. In addition, Frey and Lange [9] have shown that in many situations the Tate pairing can be computed more efficiently on Jacobians of hyperelliptic curves of genus $g > 1$

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 152–176, 2007.

than on elliptic curves. It is thus only natural that we should seek to construct "pairing-friendly" genus 2 curves, i.e. curves whose Jacobians have small embedding degree.

At present there exist very few constructions of pairing-friendly genus 2 curves. Rubin and Silverberg [26] showed that any supersingular Jacobian of a genus 2 curve has embedding degree at most 12. Galbraith, McKee, and Valença [11] demonstrated the existence of isogeny classes of ordinary abelian surfaces over prime fields with small embedding degree, and Hitt [14] demonstrated the existence of p-rank 1 abelian surfaces in characteristic 2 with small embedding degree. However, to date there exist no equations of genus 2 curves over fields of cryptographic size whose Jacobians are not supersingular and have small embedding degree.

In this paper we provide the first explicit construction of genus 2 curves whose Jacobians are ordinary, have large prime-order subgroups, and have prescribed embedding degree. Our construction is modeled on the Cocks-Pinch method for constructing pairing-friendly elliptic curves [5], and makes use of the Complex Multiplication (CM) method of curve construction. The outline of our algorithm is as follows:

1. Find primes q and r and a polynomial $h(x)$ such that if $h(x)$ is the characteristic polynomial of Frobenius of an abelian surface A over $\mathbb{F}_q$, then A has a subgroup of order r with embedding degree k.
2. Use the Igusa class polynomials for the quartic CM field $K = \mathbb{Q}[x]/(h(x))$ to construct a genus 2 curve C over $\mathbb{F}_q$ such that the Jacobian of C has characteristic polynomial of Frobenius $h(x)$.

The difficult part of the construction is ensuring that the polynomial $h(x)$ defines a CM field K for which the Igusa class polynomials can be computed efficiently using current methods. The solution to this problem is our most important theoretical contribution.

Our paper is structured as follows. Section 2 addresses Step 1 of the algorithm. In this section we give a precise definition of embedding degree, and we give a set of explicit conditions necessary for the prime q and the polynomial $h(x)$ to have the desired properties. We give separate sets of conditions that address two different notions of embedding degree: the standard notion as described above, and a new notion we call the "full embedding degree," which indicates the field over which the full set of r-torsion points of A is defined.

Section 3 addresses Step 2 of the algorithm. We find explicit formulas that relate the CM field K to the characteristic polynomial of Frobenius $h(x)$. We then use the theory of ordinary abelian varieties over finite fields to give conditions on $h(x)$ such that the desired genus 2 curve C exists over $\mathbb{F}_q$. Construction of C from roots of the Igusa class polynomials modulo q is then a standard procedure.

In Section 4 we give a complete algorithm for constructing genus 2 curves whose Jacobians are ordinary and have prescribed embedding degree. In Section 5 we give an analogous algorithm for constructing genus 2 curves whose Jacobians are ordinary and have prescribed full embedding degree; i.e. all

r-torsion points of the Jacobian are defined over a specified field extension. Examples of curves of cryptographic size constructed using these algorithms appear in the Appendices.

Finally, in Section 6 we consider possible extensions of our constructions. We show that our algorithms extend readily to produce abelian varieties that have small embedding degree with respect to subgroups of composite order; such varieties are required by a number of recent protocols. We also consider methods of generalizing the algorithm to improve the ratio between the sizes of the primes q and r. Our method produces varieties with $\log q / \log r \approx 4$; our hope is that the techniques presented here will lead to constructions that reduce this ratio to its theoretical minimum of $1/2$.

2 Pairing-Friendly Abelian Varieties

In this section we gather together facts about abelian varieties relevant to our construction. Good overviews of the subject can be found in the articles of Waterhouse and Milne [30], which focuses on varieties over finite fields, and Milne [22], which treats varieties over arbitrary fields.

An *abelian variety* A is a complete algebraic variety with a group structure whose operations are given by algebraic morphisms. An *elliptic curve* is a one-dimensional abelian variety, and an *abelian surface* is a two-dimensional abelian variety. If A is an abelian variety defined over a field K, we denote by $A(K)$ the set of K-rational points of A. If r is an integer, then $A[r]$ denotes the set of all r-torsion points of A, defined over an algebraic closure of K. We denote by $A(K)[r]$ the set of r-torsion points of A defined over K. If A has dimension g and r is prime to the characteristic of K, then $A[r] \cong (\mathbb{Z}/r\mathbb{Z})^{2g}$.

Every abelian variety A defined over a finite field $\mathbb{F}_q$ has an endomorphism called the *Frobenius endomorphism*, which operates by raising the coordinates of a point to the qth power. The Frobenius endomorphism satisfies an integer polynomial $h(x)$ known as the *characteristic polynomial of Frobenius*. By a theorem of Weil [22, Theorem 19.1], all of the complex roots of $h(x)$ have absolute value $\sqrt{q}$; such a polynomial is called a *q-Weil polynomial*. If A has dimension g, then this polynomial is of the form

$$\begin{aligned} h(x) = x^{2g} + a_1 x^{2g-1} + \ldots + a_{g-1} x^{g+1} + a_g x^g \\ + a_{g-1} q x^{g-1} + \ldots + a_1 q^{g-1} x + q^g, \end{aligned} \tag{2.1}$$

By Honda-Tate theory [28], q-Weil polynomials are in one-to-one correspondence with isogeny classes of abelian varieties over $\mathbb{F}_q$.

If A is an abelian variety with characteristic polynomial of Frobenius $h(x)$, then $\#A(\mathbb{F}_q) = h(1)$. We say that A is *ordinary* if the middle coefficient a_g of $h(x)$ is relatively prime to q, and A is *supersingular* if all of the complex roots of $h(x)$ are roots of unity times $\sqrt{q}$. (For other equivalent definitions of ordinary and supersingular, see [16, Definition 3.1] or [10, Theorem 1].) If $g \geq 2$ then there are g-dimensional abelian varieties that are neither ordinary nor supersingular.

The most common abelian varieties used in cryptography are Jacobians of hyperelliptic curves of genus $g \geq 1$. A hyperelliptic curve of genus g (over a field of characteristic $\neq 2$) is the normal projective closure of a nonsingular affine curve of the form $y^2 = f(x)$, with $\deg f = 2g + 1$ or $2g + 2$. The Jacobian of a projective genus g curve C, denoted $\mathrm{Jac}(C)$, is a g-dimensional principally polarized abelian variety whose points are degree zero divisors on C modulo principal divisors.

2.1 Pairings and Embedding Degrees

The two most common pairings on abelian varieties used in cryptography are the Weil and Tate pairings. Let A be an abelian variety defined over a field K, and let r be a positive integer. Let μ_r be the rth roots of unity in an algebraic closure of K. The Weil pairing is a nondegenerate bilinear map

$$e_{weil,r} : A[r] \times A[r] \to \mu_r,$$

while the Tate pairing is a nondegenerate bilinear map

$$e_{tate,r} : A(K)[r] \times A(K)/rA(K) \to K^\times/(K^\times)^r.$$

If $\mu_r \subset K$, then the target group $K^\times/(K^\times)^r$ is isomorphic to μ_r; otherwise it is isomorphic to μ_s for some $s \mid r$. Thus to obtain Weil or Tate pairing values of order r, we must work over a field containing the rth roots of unity. We define the embedding degree to be the extension degree of the smallest such field.

Definition 2.1. Let A be an abelian variety defined over a field K, and let r be a positive integer relatively prime to $\mathrm{char}(K)$. We say that A has *embedding degree k with respect to r* if

1. A has a K-rational point of order r, and
2. k is the smallest integer such that μ_r is contained in a degree-k extension of K.

If C is a projective nonsingular curve, then we say that C has embedding degree k with respect to r if and only if the Jacobian of C does.

Remark 2.2. If A is an abelian variety over a finite field $\mathbb{F}_q$ with an $\mathbb{F}_q$-rational point of order r, then the following conditions are equivalent:

1. A has embedding degree k with respect to r.
2. k is the smallest integer such that r divides $q^k - 1$.
3. k is the multiplicative order of q modulo r.

Furthermore, if r is square-free these conditions are equivalent to

4. $\Phi_k(q) \equiv 0 \pmod{r}$, where Φ_k is the kth cyclotomic polynomial. (Cf. [8, Proposition 2.4]).

The embedding degree gets its name because we can use a pairing to "embed" a cyclic subgroup of A of order r into the multiplicative group of the degree-k extension of K. The MOV attack on the discrete logarithm problem on supersingular elliptic curves [20] makes use of such an "embedding." If A is a g-dimensional abelian variety defined over $\mathbb{F}_q$ with $q = p^d$ and ℓ is the multiplicative order of p modulo r, then the quantity ℓ/dg is a good measure of the security of cryptosystems based on A [15]. If $\mathbb{F}_q$ is a prime field (i.e. $d = 1$) then this quantity is equal to k/g.

In general we expect a "random" abelian variety over $\mathbb{F}_q$ with a point of order r to have embedding degree $k \approx r$; this statement has been made more precise in the case of elliptic curves by Balasubramanian and Koblitz [1] and Luca, Mireles, and Shparlinski [19]. In cryptographic applications r will be at least 2^{160}, so computing pairings on random abelian varieties over $\mathbb{F}_q$ appears hopeless. Thus we wish to construct abelian varieties over finite fields that have points of large order r and small embedding degree with respect to r; such varieties are called "pairing-friendly."

Our first task is to give some conditions on the characteristic polynomial of Frobenius that are sufficient for A to have embedding degree k.

Proposition 2.3. *Let A be an abelian variety over $\mathbb{F}_q$, and let $h(x)$ be the characteristic polynomial of Frobenius of A. Let $r \nmid q$ be a prime number and k a positive integer, and suppose the following hold:*

$$h(1) \equiv 0 \pmod{r},$$
$$\Phi_k(q) \equiv 0 \pmod{r},$$

where Φ_k is the kth cyclotomic polynomial. Then A has embedding degree k with respect to r.

Furthermore, if $k > 1$ then $A(\mathbb{F}_{q^k})$ contains two linearly independent r-torsion points.

Proof. The condition $r \mid h(1)$ guarantees that A has an $\mathbb{F}_q$-rational point of order r, and by Remark 2.2 the condition $r \mid \Phi_k(q)$ implies that A has embedding degree k with respect to r.

The proof of the "furthermore" clause follows an argument of Balasubramanian and Koblitz [1, Theorem 1]. Let $\tilde{h}(x)$ be the reduction of $h(x)$ modulo r. The roots of $\tilde{h}(x)$ are the eigenvalues of the Frobenius endomorphism F on $A[r]$. From equation (2.1) we see that $h(x) = (x^2/q)^g h(q/x)$, so roots of $\tilde{h}(x)$ come in pairs $(\alpha, q/\alpha)$. The hypothesis $h(1) \equiv 0 \pmod{r}$ thus implies that $\tilde{h}(q)$ is also zero. The hypotheses $\tilde{h}'(1) \neq 0$ and $k > 1$ imply that 1 and q are distinct roots with multiplicity 1, so $A[r]$ has a one-dimensional eigenspace with eigenvalue 1, and a one-dimensional eigenspace with eigenvalue q. Since $q^k = 1 \pmod{r}$, F^k acts trivially on the two-dimensional span of these eigenspaces, so $\dim A[r](\mathbb{F}_{q^k}) \geq 2$. □

If $\dim A = 1$ (i.e. A is an elliptic curve) and the "furthermore" clause of Proposition 2.3 holds, then since $A[r]$ is two-dimensional we must have $A[r] \subset A(\mathbb{F}_{q^k})$.

However, if $\dim A > 1$, then in general $A[r]$ will not be contained in $A(\mathbb{F}_{q^k})$. Thus we define a second type of embedding degree, which indicates the extension degree of the smallest field over which all r-torsion points of A are defined.

Definition 2.4. Let A be an abelian variety defined over a field K, and let r be a positive integer relatively prime to $\mathrm{char}(K)$. We say that A has *full embedding degree k with respect to r* if

1. A has a K-rational point of order r, and
2. k is the smallest integer such that *all* r-torsion points of A are defined over a degree-k extension of K.

If C is a projective nonsingular curve, then we say that C has full embedding degree k with respect to r if and only if the Jacobian of C does.

Remark 2.5. The non-degeneracy of the Weil pairing [22, §16] implies that the full embedding degree is a multiple of the embedding degree.

We next give a criterion that determines when all of the r-torsion points are defined over a given extension field.

Proposition 2.6. *Let A be a g-dimensional abelian variety over $\mathbb{F}_q$, let R be the endomorphism ring of A, and suppose R is a Dedekind domain. Let F be the Frobenius endomorphism of A, and for $k \geq 1$ let $h_k(x)$ be the characteristic polynomial of F^k. Let $r \nmid q$ be a rational prime unramified in R, and suppose that for some k, $h_k(x) \equiv (x-1)^{2g} \pmod{r}$. Then $A[r] \subset A(\mathbb{F}_{q^k})$.*

Proof. Let $\pi \in R$ be the Frobenius endomorphism of A. By a result of Eisenträger and Lauter [6, Fact 10], $A[r] \subset A(\mathbb{F}_{q^k})$ if and only if $\pi^k - 1 \in rR$. Since R is a Dedekind domain and r is unramified in R, it suffices to show that $\pi^k - 1 \in \mathfrak{p}$ for every prime $\mathfrak{p}$ of R dividing r. Since π^k is a root of $h_k(x)$, the hypothesis $h_k(x) \equiv (x-1)^{2g} \pmod{r}$ implies that $(\pi^k - 1)^{2g} \in \mathfrak{p}$ for every $\mathfrak{p} \mid r$, and since $R/\mathfrak{p}$ is a field we conclude that $\pi^k - 1 \in \mathfrak{p}$ for every $\mathfrak{p} \mid r$. □

Using Proposition 2.6, we can now give a statement analogous to Proposition 2.3 that gives us sufficient conditions for A to have full embedding degree k.

Proposition 2.7. *Let A be an abelian variety over $\mathbb{F}_q$, and let $h(x)$ be the characteristic polynomial of Frobenius of A. Let $r \nmid q$ be a prime number, let $\tilde{h}(x) \in \mathbb{F}_r[x]$ be $h(x)$ modulo r, and suppose $\tilde{h}(1) = 0$. Let $\{\alpha_i\}$ be the roots of $\tilde{h}(x)$ in $\overline{\mathbb{F}}_r$, and suppose that k is the least common multiple of the multiplicative orders of all the α_i. Suppose that either (1) $\gcd(\tilde{h}(x), \tilde{h}'(x)) = 1$ or (2) $\mathrm{End}(A)$ is a Dedekind domain and r is unramified in $\mathrm{End}(A)$. Then A has full embedding degree k with respect to r.*

Proof. The condition $\tilde{h}(1) = 0$ guarantees that A has an $\mathbb{F}_q$-rational point of order r. The α_i are the eigenvalues of the Frobenius endomorphism F on $A[r]$. Since $\alpha_i^k = 1$, all of the $2g$ eigenvalues of F^k are 1, and by assumption k is the smallest integer with this property.

If $\gcd(\tilde{h}(x), \tilde{h}'(x)) = 1$, then all the eigenvalues of F are distinct, so F is diagonalizable. Thus F^k is the identity on $A[r]$, and we conclude that $A[r] \subset A(\mathbb{F}_{q^k})$. If $\mathrm{End}(A)$ is a Dedekind domain and r is unramified in $\mathrm{End}(A)$, then since the characteristic polynomial of F^k modulo r is $(x-1)^{2g}$, we may apply Proposition 2.6 to deduce that $A[r] \subset A(\mathbb{F}_{q^k})$. Thus in both cases we see that A has full embedding degree k with respect to r. □

The security of pairing-based cryptographic protocols depends on both the size r of the subgroup involved in the pairing and the size q^k of the finite field into which the pairing maps. Ideally r is very close to the total number of points on the abelian variety. However, many of the constructions of pairing-friendly varieties give values of r whose size is some fraction of the total number of points on the variety. We define a parameter ρ that measures this ratio. Since the Weil conjectures [22, Theorem 19.1] imply that $\#A(\mathbb{F}_q) = q^g + O(q^{g-1/2})$, it is reasonable to use q^g as an approximation to $\#A(\mathbb{F}_q)$ in our definition.

Definition 2.8. Let $A/\mathbb{F}_q$ be a g-dimensional abelian variety, and suppose r divides $\#A(\mathbb{F}_q)$. The *ρ-value of A* (with respect to r) is defined to be

$$\rho(A) = \frac{g \log q}{\log r}.$$

Varieties with a prime number of points, such as MNT elliptic curves [23], will have ρ-value very close to 1; this is the "ideal" case. The expression $k\rho/g$ measures the ratio of the size of the field into which the pairing maps to the size of the prime-order subgroup on the variety. Recommended values of this expression to achieve "balanced" security levels comparable to standard sizes of keys for symmetric encryption have been given by several authors; for a summary, see [8, Table 1.1].

3 Constructing Ordinary Abelian Surfaces Via the Genus 2 CM Method

In this section we consider the problem of generating a genus 2 curve C such that $\mathrm{Jac}(C)$ is ordinary and has characteristic polynomial of Frobenius equal to a specified q-Weil polynomial $h(x)$.

Let A be an absolutely simple ordinary g-dimensional abelian variety over a finite field $\mathbb{F}_q$. We denote by $\mathrm{End}(A)$ the ring of endomorphisms of A defined over $\overline{\mathbb{F}}_q$. This ring is a rank-$2g$ $\mathbb{Z}$-module that is isomorphic as a $\mathbb{Z}$-algebra to an order in the ring of integers in a number field K. We say that such a variety A has *complex multiplication by K* or *CM by K*. The field K has degree $2g$ and is an imaginary quadratic extension of a totally real number field of degree g; a field with these properties is called a *CM field*. A CM field K is *primitive* if it contains no proper CM subfields.

The fundamental fact we will use in our construction of pairing-friendly abelian surfaces is the following, which relates the CM field to the characteristic polynomial of Frobenius.

Fact 3.1 ([30, Theorem 8]). *Let K be a CM field. An ordinary abelian variety $A/\mathbb{F}_q$ has CM by K if and only if $K \cong \mathbb{Q}[x]/(h(x))$, where $h(x)$ is the characteristic polynomial of Frobenius of A.*

In the case of abelian surfaces, we can give a more explicit relation between the CM field and the characteristic polynomial of Frobenius.

Lemma 3.2. *Let $h(x) \in \mathbb{Z}[x]$ be a polynomial with integer coefficients of the form*

$$h(x) = x^4 - sx^3 + tx^2 - sqx + q^2. \tag{3.1}$$

Then the four complex roots of $h(x)$ are

$$\frac{s}{4} + \frac{1}{2}\sqrt{\frac{s^2}{4} - t + 2q} \pm \frac{1}{2}\sqrt{\left(\frac{s^2}{2} - t - 2q\right) + s\sqrt{\frac{s^2}{4} - t + 2q}},$$

$$\frac{s}{4} - \frac{1}{2}\sqrt{\frac{s^2}{4} - t + 2q} \pm \frac{1}{2}\sqrt{\left(\frac{s^2}{2} - t - 2q\right) - s\sqrt{\frac{s^2}{4} - t + 2q}}.$$

Proof. An easy calculation shows that if α is a root of $h(x)$, then $\alpha + q/\alpha$ is a root of $x^2 - sx + t - 2q$. The result then follows from two successive applications of the quadratic formula. □

Proposition 3.3. *Let $h(x)$ be a polynomial of the form (3.1). Let $\delta = s^2/4 - t + 2q$. Suppose the following hold:*

$$\delta > 0 \tag{3.2}$$

$$\frac{s^2}{2} - t - 2q \pm s\sqrt{\delta} < 0 \tag{3.3}$$

$$\gcd(t, q) = 1 \tag{3.4}$$

Then there is an abelian surface A such that the characteristic polynomial of Frobenius of A is equal to $h(x)$. Furthermore, A has CM by $\mathbb{Q}(\eta)$, where

$$\eta = \sqrt{\left(\frac{s^2}{2} - t - 2q\right) + s\sqrt{\frac{s^2}{4} - t + 2q}}. \tag{3.5}$$

Proof. By Lemma 3.2, $h(x)$ has a root in $K = \mathbb{Q}(\eta)$, so $K \cong \mathbb{Q}[x]/(h(x))$. Conditions (3.2) and (3.3) ensure that K is a purely imaginary quadratic extension of the real quadratic field $\mathbb{Q}(\sqrt{\delta})$. Under this hypothesis, one can compute that all four of the roots of $h(x)$ given by Lemma 3.2 have complex absolute value $\sqrt{q}$. Thus $h(x)$ is a q-Weil polynomial, so by Honda-Tate theory [28] there is an isogeny class $\mathcal{A}$ of abelian varieties over $\mathbb{F}_q$ with characteristic polynomial of Frobenius $h(x)$. Condition (3.4) implies that any $A \in \mathcal{A}$ is ordinary, so by Fact 3.1, any $A \in \mathcal{A}$ has CM by K. □

If the CM field $K = \mathbb{Q}(\eta)$ is primitive then we can go one step further and say that there is an abelian surface $A \in \mathcal{A}$ such that A is the Jacobian of a genus 2 curve over $\mathbb{F}_q$ and $\mathrm{End}(A)$ is the ring of integers of K.

Proposition 3.4. *Let $h(x)$ be a polynomial of the form (3.1) satisfying the conditions of Proposition 3.3. Let η be defined by equation (3.5), and suppose that the number field $K = \mathbb{Q}(\eta)$ is a primitive quartic CM field. Then there is a genus 2 curve $C/\mathbb{F}_q$ such that $\mathrm{Jac}(C)$ has characteristic polynomial of Frobenius $h(x)$ and $\mathrm{End}(\mathrm{Jac}(C)) \cong \mathcal{O}_K$, the ring of integers of K.*

Proof. Let $A \in \mathcal{A}$ be an abelian surface in the isogeny class of abelian varieties given by Proposition 3.3. Since K is primitive and A is ordinary, it follows from the Honda-Tate theorem [28] that A is absolutely simple. By a theorem of Weil (cf. [24]), an absolutely simple principally polarized abelian surface is the Jacobian of a genus 2 curve. It thus suffices to show that we can find an $A \in \mathcal{A}$ that is principally polarized and has endomorphism ring isomorphic to $\mathcal{O}_K$.

Let K_0 be the real quadratic subfield of K. By the work of Howe [16, Propositions 5.7 and 10.1], it suffices to show that there is a finite prime $\mathfrak{p}$ of K_0 that ramifies in K. A variation of Howe's proof of [16, Lemma 12.1] shows that if there is no finite prime $\mathfrak{p}$ of K_0 that ramifies in K, then K contains an imaginary quadratic subfield, contradicting the assumption that K is primitive. □

Our algorithms in Sections 4 and 5 for generating pairing-friendly abelian surfaces will produce primes q and q-Weil polynomials $h(x)$ satisfying the hypotheses of Proposition 3.4. To construct the genus 2 curves specified by the proposition, we turn to genus 2 invariant theory.

If $\mathbb{F}$ is a field with $\mathrm{char}(\mathbb{F}) \neq 2$, then $\overline{\mathbb{F}}$-isomorphism classes of genus 2 curves defined over $\mathbb{F}$ are in one-to-one correspondence with triples $(j_1, j_2, j_3) \in \mathbb{F}^3$. The triple (j_1, j_2, j_3) corresponding to a curve C is called the curve's *absolute invariants*. Let K be a primitive quartic CM field, and let $\mathscr{C}_K$ be the set of isomorphism classes of genus 2 curves over $\mathbb{C}$ such that $\mathrm{End}(\mathrm{Jac}(C)) \cong \mathcal{O}_K$, the ring of integers in K. The *Igusa class polynomials* of K are defined to be

$$H_i(x) = \prod_{C \in \mathscr{C}_K} (x - j_i(C))$$

for $i = 1, 2, 3$; these polynomials have rational coefficients. There are currently several methods for computing the Igusa class polynomials: a complex-analytic algorithm involving modular functions [27], [29], [31]; a Chinese Remainder Theorem algorithm that computes the $H_i(x)$ modulo many small primes [6], [7]; and a p-adic lifting algorithm [12]. All three methods are currently limited to CM fields K with small discriminant and class number.

By the Serre-Tate theory of canonical liftings [17], any ordinary abelian variety A over a finite field is the reduction modulo a suitable prime $\mathfrak{p}$ of an abelian variety $\tilde{A}$ over $\mathbb{C}$ with $\mathrm{End}(\tilde{A}) \cong \mathrm{End}(A)$. Furthermore, if A is the Jacobian of a genus 2 curve C, then $\tilde{A}$ is the Jacobian of a genus 2 curve $\tilde{C}$, and the absolute invariants of C are the reduction modulo $\mathfrak{p}$ of the absolute invariants of $\tilde{C}$. The requirement that A be ordinary is essential; see [13, Section 4] for further details. Combining these facts gives the following statement.

Fact 3.5. *Let K be a primitive quartic CM field, and let $H_i(x)$ be the Igusa class polynomials of K. Let $q = p^d$ be a prime power, and let $C/\mathbb{F}_q$ be a genus 2*

curve. Suppose that $\mathrm{End}(\mathrm{Jac}(C)) \cong \mathcal{O}_K$ *(so in particular,* $\mathrm{Jac}(C)$ *is ordinary). Then* p *does not divide the denominator of any coefficient of any* $H_i(x)$*, and for each* i *the absolute invariant* j_i *of* C *is a root of* $H_i(x)$ *modulo* p*.*

Thus given a q-Weil polynomial $h(x)$ satisfying the hypotheses of Proposition 3.4 with q prime, we can construct the absolute invariants of C by computing the Igusa class polynomials for K and finding roots of the polynomials modulo q. Constructing the curve C from its absolute invariants is then a standard procedure. Algorithm 4.3 below gives step by step instructions for this construction.

4 Constructing Genus 2 Curves with Prescribed Embedding Degree

We have seen in Sections 2 and 3 that to construct an abelian surface with prescribed embedding degree k, it suffices to find a q-Weil polynomial $h(x)$ satisfying the conditions of Propositions 2.3 and 3.4. Given such an $h(x)$, Fact 3.5 says that we can use the Igusa class polynomials of $K = \mathbb{Q}[x]/(h(x))$ to find a genus 2 curve C such that $\mathrm{Jac}(C)$ has the desired properties. Since current methods of computing the Igusa class polynomials are limited to a small range of quartic CM fields K, we will specify K as an input to our algorithm, and assume that K is chosen such that the Igusa class polynomials for K are known or can be easily computed.

Recall that a CM field is a purely imaginary quadratic extension of a totally real field. Thus all primitive CM fields K of degree 4 can be written in the form $K = \mathbb{Q}(\sqrt{-a + b\sqrt{d}})$ for some integers $a, b, d > 0$ with $a^2 - b^2 d > 0$. If $a^2 - b^2 d$ is not a square then K is primitive.

If we fix an element $\xi = \sqrt{-a + b\sqrt{d}}$ generating our CM field K and require that the element η given by Proposition 3.3 is equal to ξ, then we have three equations in the three variables q, s, t. By Proposition 2.3, requiring that an abelian surface with characteristic polynomial $h(x)$ has embedding degree k imposes two additional constraints on q, s, t, and it will almost certainly be impossible to find q, s, t satisfying all five equations. Thus we wish to add at least two degrees of freedom to our description of the CM field so that instead of requiring $\xi = \eta$, we only require $\mathbb{Q}(\xi) \cong \mathbb{Q}(\eta)$. The following proposition achieves this goal.

Proposition 4.1. *Suppose* $K = \mathbb{Q}(\sqrt{-a + b\sqrt{d}})$ *is a primitive quartic CM field. Let* u, v, w *be integers, and let*

$$\alpha = w^2(au^2 + adv^2 + 2bduv), \tag{4.1}$$

$$\beta = bu^2 + bdv^2 + 2auv, \tag{4.2}$$

$$\delta = dw^4. \tag{4.3}$$

Then K *is isomorphic to* $\mathbb{Q}(\sqrt{-\alpha + \beta\sqrt{\delta}})$*.*

Proof. Let $L = \mathbb{Q}(\sqrt{-\alpha + \beta\sqrt{\delta}})$. Since K is primitive it contains a unique quadratic subfield K_0 isomorphic to $\mathbb{Q}(\sqrt{d})$. Since $\delta = dw^4$, K_0 is a quadratic subfield of L, so it suffices to show that K and L are isomorphic as quadratic extensions of K_0. One can check that the choices of α, β, δ above satisfy

$$-\alpha + \beta\sqrt{\delta} = (-a + b\sqrt{d})(u - v\sqrt{d})^2 w^2,$$

so $(-a + b\sqrt{d})/(-\alpha + \beta\sqrt{\delta})$ is a square in K_0, and K and L are isomorphic. □

We now turn to the task of constructing the characteristic polynomial of Frobenius of a pairing-friendly abelian surface A. We will fix throughout a prime r and an embedding degree k, and look for a polynomial $h(x)$ satisfying the conditions of Proposition 2.3. As remarked above, we will also fix a CM field $K = \mathbb{Q}(\sqrt{-a + b\sqrt{d}})$ and require that $K \cong \mathbb{Q}[x]/(h(x))$.

Recall that $h(x)$ has the form

$$h(x) = x^4 - sx^3 + tx^2 - sqx + q^2. \tag{4.4}$$

If we are given $\eta = \sqrt{-a + b\sqrt{d}}$, Propositions 3.3 and 4.1 give a set of conditions sufficient for A to have CM by $\mathbb{Q}(\eta)$, namely, that for some u, v, w, the following hold:

$$\frac{s^2}{2} - t - 2q = -w^2(au^2 + adv^2 + 2bduv) \tag{4.5}$$

$$s = bu^2 + bdv^2 + 2auv \tag{4.6}$$

$$\frac{s^2}{4} - t + 2q = dw^4. \tag{4.7}$$

By Proposition 2.3, the condition that A has embedding degree k with respect to r is equivalent to

$$q^2 + 1 - s(q+1) + t \equiv 0 \pmod{r} \tag{4.8}$$

$$\Phi_k(q) \equiv 0 \pmod{r}, \tag{4.9}$$

where Φ_k is the kth cyclotomic polynomial.

Conditions (4.5) through (4.9) together comprise five equations in six variables over $\mathbb{F}_r$, so we can expect to find solutions (q', s', t', u', v', w') in $\mathbb{F}_r^6$ for some positive fraction of all primes r. Since we have an extra degree of freedom, we can loop on one variable and search for solutions to the five equations in the remaining five variables. When a solution is found, we can then lift u', v', w' to integers u, v, w and use equations (4.5) through (4.7) to compute q, s, t congruent to q', s', t' modulo r. Explicitly, we have s given by equation (4.6) and

$$t = \frac{1}{2}w^2(au^2 + adv^2 + 2bduv) - \frac{1}{2}dw^4 + \frac{3}{8}(bu^2 + bdv^2 + 2auv)^2 \tag{4.10}$$

$$q = \frac{1}{4}w^2(au^2 + adv^2 + 2bduv) + \frac{1}{4}dw^4 + \frac{1}{16}(bu^2 + bdv^2 + 2auv)^2. \tag{4.11}$$

We can choose different lifts u, v, w until the value of q computed is prime. (In theory we could allow q to be a prime power, but since almost all prime powers in a given interval are prime, in practice we find that q will always be prime.) We summarize the procedure in the following algorithm.

Algorithm 4.2. The following algorithm takes as input five positive integers a, b, d, k, r and a (finite) interval $I \subset \mathbb{Z}$, such that $K = \mathbb{Q}(\sqrt{-a + b\sqrt{d}})$ is a primitive quartic CM field, and r is a prime congruent to 1 (mod k). The algorithm outputs either the symbol $\perp$ or a prime q and a polynomial $h(x)$ of the form (4.4). If the output is not $\perp$, then there is a genus 2 curve $C/\mathbb{F}_q$ such that

- Jac(C) has characteristic polynomial of Frobenius $h(x)$,
- Jac(C) has endomorphism ring isomorphic to $\mathcal{O}_K$, and
- Jac(C) has embedding degree k with respect to r.

1. Set $v' \leftarrow 0$.
2. Using v' as the value of the variable v, find a simultaneous solution $(q', s', t', u', w') \in \mathbb{F}_r^5$ to equations (4.5) through (4.9) modulo r. If none exists, go to Step 5.
3. Let u_0, v_0, w_0 be the unique integers in $[0, r)$ congruent to u', v', w' respectively.
4. For each triple $(i_1, i_2, i_3) \in I^3$, do the following:
   (a) Set $u \leftarrow u_0 + i_1 r$, $v \leftarrow v_0 + i_2 r$, $w \leftarrow w_0 + i_3 r$.
   (b) Compute q, s, and t by equations (4.11), (4.6), and (4.10), respectively.
   (c) If t and q are integers, q is prime, and $q \nmid t$, go to Step 6.
5. Set $v' \leftarrow v' + 1$. If $v' \equiv 0 \pmod{r}$ then output $\perp$; otherwise go to Step 2.
6. Output q and the polynomial $h(x) = x^4 - sx^3 + tx^2 - sqx + q^2$.

By Fact 3.5, if q and $h(x)$ are outputs of Algorithm 4.2, we can construct the desired curve C by the following procedure:

Algorithm 4.3. The following algorithm takes as input a prime q and a q-Weil polynomial $h(x)$. Let $K = \mathbb{Q}[x]/(h(x))$. With high probability, the algorithm outputs a genus 2 curve $C/\mathbb{F}_q$ such that

- Jac(C) has characteristic polynomial of Frobenius $h(x)$, and
- Jac(C) has endomorphism ring isomorphic to $\mathcal{O}_K$.

1. Compute the Igusa class polynomials $H_i(x)$ for K, via e.g. [31], [6], or [12].
2. Let S_i for $i = 1, 2, 3$ be the sets of roots in $\mathbb{F}_q$ of the $H_i(x) \pmod{q}$.
3. Let $n_1 = h(1)$ and $n_2 = h(-1)$. For each $(j_1, j_2, j_3) \in S_1 \times S_2 \times S_3$, do the following:
   (a) Use Mestre's algorithm [21] to compute a curve $C/\mathbb{F}_q$ with absolute Igusa invariants (j_1, j_2, j_3).
   (b) Choose a random point $P \in \text{Jac}(C)(\mathbb{F}_q)$.
   (c) If $[n_1]P = O$, return C.
   (d) If $[n_2]P = O$, return the quadratic twist of C.
   (e) If $K \cong \mathbb{Q}(\zeta_5)$, repeat Steps 3b through 3d for each quintic twist of C.

We note that if the correct triple of invariants is tested in Step 3 then the algorithm will output the correct curve C. The algorithm will only output an incorrect curve C if the random point P has order dividing both $\#\operatorname{Jac}(C)$ and one of n_1 or n_2. If q is reasonably large then this event occurs with negligible probability; to further reduce the probability of error one could choose more random points P.

We have run Algorithm 4.2 for various prime values r of cryptographic size and various embedding degrees k, and used Algorithm 4.3 to generate pairing-friendly genus 2 curves C. Some examples appear in Appendix A.

4.1 Analysis of Algorithm 4.2

The success of Algorithm 4.2 depends on two factors: finding a valid solution (q', s', t', u', w') in Step 2, and finding lifts q and t in Step 4 such that q and t are integers and q is prime. In our analysis we treat these two factors independently.

A theoretical analysis of the probability of finding a valid solution in Step 2 is beyond our means at this time; instead, we provide some experimental data. We fixed an embedding degree k and a bit size μ and chose random primes $r \in [2^{\mu-1}, 2^{\mu}]$ as well as random integers $a, b, d \in [1, r]$ such that $a^2 - b^2 d > 0$; then $\mathbb{Q}(\sqrt{-a + b\sqrt{d}})$ is a quartic CM field. For each k between 2 and 30 we ran 1,000 such trials, increasing the value of v' until a solution was found. We found that for each k, between 35 and 40 percent of the trials produced a solution; these data appear to be independent of the bit size μ. In cases where a solution was found, the average number of v' tried was less than 2, with the number approaching 1 as $\varphi(k)$ grew. We thus aborted the trial if no solution was found for any $v' \leq 20$, and we recommend that others implementing the algorithm do the same.

Whether t and q are integers can be determined by an analysis modulo 4 of the parameters involved in equations (4.10) and (4.11). In all cases, for a fixed a, b, d, we see that q is an integer for at least $1/8$ of the possible choices of u, v, w, and t is an integer whenever q is. This analysis also leads to the following observation.

Remark 4.4. For the following choices of a, b, d, the value of q will never be an odd integer: $a \equiv b \equiv 0 \pmod 4$; and $(a, b, d) \equiv (0, 2, 0)$, $(1, 2, 1)$, or $(2, 2, 0) \pmod 4$. Thus for Algorithm 4.2 to produce prime values of q these choices must be avoided. In practice this restriction does not pose a problem, since if $b = 2^{\ell} b'$ then $\mathbb{Q}(\sqrt{-a + b\sqrt{d}})$ is isomorphic to $\mathbb{Q}(\sqrt{-a + b'\sqrt{4^{\ell} d}})$.

If we treat u' and w' as random integers in $[0, r)$, we see from equation (4.11) that we can expect q to be roughly the same size as r^4. Thus by the Prime Number Theorem, we should expect to try roughly $4 \log r$ integer values of q before we find one that is prime.

Since $q \approx r^4$, by Definition 2.8 the ρ-values of the varieties generated will be roughly 8; the examples in Appendix A bear this heuristic observation out in practice. As a consequence, to achieve comparable levels of security on the

abelian surface A and in the finite field $\mathbb{F}_{q^k}$, the chosen embedding degree k should be one eighth of the embedding degree of an "ideal" abelian surface of prime order $r \approx q^2$.

5 Constructing Genus 2 Curves with Prescribed Full Embedding Degree

Algorithm 4.2 constructs abelian surfaces $A/\mathbb{F}_q$ that have prescribed embedding degree k with respect to a subgroup of a given size r. By Proposition 2.3, this guarantees that two roots of the characteristic polynomial of Frobenius have order dividing k, so if $k > 1$ then two dimensions of $A[r]$ are contained in $A(\mathbb{F}_{q^k})$. However, the algorithm makes no claim about the remaining two roots of the characteristic polynomial, so we have no control over the full embedding degree of A, i.e. the field over which all of the points of $A[r]$ are defined. Such control would be necessary, for instance, in a protocol that required pairings involving three or four linearly independent r-torsion points. We thus seek a method of producing abelian surfaces with prescribed full embedding degree.

As in the previous section, we fix a prime r and an embedding degree k. To construct an abelian variety with prescribed full embedding degree k with respect to r, by Proposition 2.7 it suffices to produce a characteristic polynomial of Frobenius with four distinct roots, all of which have order dividing k in $\mathbb{F}_r^\times$. There are many possibilities for such a polynomial; for our construction we will choose the polynomial to have the form

$$h(x) \equiv x^4 - (q^2+1)x^2 + q^2 \pmod{r} \tag{5.1}$$

This polynomial has roots $1, -1, q, -q$, so choosing q to be a primitive kth root of unity modulo r will give us the desired condition on the orders of the roots. To ensure that all four roots are distinct we require $k \geq 3$. Since -1 has order 2, the full embedding degree k must be even.

Now suppose as in the previous section that the characteristic polynomial of Frobenius (over the integers) is given by

$$h(x) = x^4 - sx^3 + tx^2 - sqx + q^2. \tag{5.2}$$

Equation (5.1) and the requirement that q be a primitive kth root of unity modulo r tell us that

$$\Phi_k(q) \equiv 0 \pmod{r} \tag{5.3}$$

$$s \equiv 0 \pmod{r} \tag{5.4}$$

$$t \equiv -q^2 - 1 \pmod{r}, \tag{5.5}$$

where Φ_k is the kth cyclotomic polynomial.

As before, due to the limitations of the genus 2 CM method we will fix a quartic CM field $K = \mathbb{Q}(\sqrt{-a + b\sqrt{d}})$, and look for a polynomial $h(x)$ such that $K \cong \mathbb{Q}[x]/(h(x))$. By Propositions 3.3 and 4.1 it suffices to find u, v, and

w satisfying equations (4.5) through (4.7). These three equations together with equations (5.3) through (5.5) give six relations in six variables, so we can expect to find a valid solution (q', s', t', u', v', w') modulo r for some positive fraction of all r. As in Algorithm 4.2, if a solution exists we can then lift u', v', w' to integers u, v, w and use equations (4.11), (4.6), and (4.10) to compute q, s, t congruent to q', s', t' modulo r. We choose different lifts u, v, w until the value of q computed is prime. We summarize the procedure in the following algorithm.

Algorithm 5.1. The following algorithm takes as input five positive integers a, b, d, k, r and a (finite) interval $I \subset \mathbb{Z}$, such that $K = \mathbb{Q}(\sqrt{-a + b\sqrt{d}})$ is a primitive quartic CM field, $k \geq 4$ is even, and r is a prime congruent to 1 (mod k). The algorithm outputs either the symbol $\perp$ or a prime q and a polynomial $h(x)$ of the form (5.2). If the output is not $\perp$, then there is a genus 2 curve $C/\mathbb{F}_q$ such that

- Jac(C) has characteristic polynomial of Frobenius $h(x)$,
- Jac(C) has endomorphism ring isomorphic to $\mathcal{O}_K$, and
- Jac(C) has full embedding degree k with respect to r.

1. Find a simultaneous solution $(q', s', t', u', v', w') \in \mathbb{F}_r^6$ to equations (4.5) through (4.7) and (5.3) through (5.5). If none exists, output $\perp$.
2. Let u_0, v_0, w_0 be the unique integers in $[0, r)$ congruent to u', v', w' respectively.
3. For each triple $(i_1, i_2, i_3) \in I^3$, do the following:
   (a) Set $u \leftarrow u_0 + i_1 r$, $v \leftarrow v_0 + i_2 r$, $w \leftarrow w_0 + i_3 r$.
   (b) Compute q, s, and t by equations (4.11), (4.6), and (4.10), respectively.
   (c) If t and q are integers, q is prime, and $q \nmid t$, go to Step 4.
   (d) If every triple (i_1, i_2, i_3) has been tested, output $\perp$.
4. Output q and the polynomial $h(x) = x^4 - sx^3 + tx^2 - sqx + q^2$.

If q and $h(x)$ are outputs of Algorithm 5.1, we can use Algorithm 4.3 to construct the desired curve C. As noted in Remark 4.4, certain choices of (a, b, d) input into Algorithm 5.1 must be avoided. Since the algorithm requires solving six equations in six variables, there is no extra degree of freedom that we can use for a loop as in Algorithm 4.2. Thus if no solution to the system exists for a given r, we must try again with a different r.

We have run Algorithm 5.1 for various prime values r of cryptographic size and various embedding degrees k, and used Algorithm 4.3 to generate pairing-friendly genus 2 curves C. Some examples with r of cryptographic size appear in Appendix B. As before, the ρ-values of the varieties generated are roughly 8.

To determine the success probability of Algorithm 5.1 we ran Step 1 for random primes r of various sizes and integers a, b, d defining quartic CM fields. We found that for even values of k between 2 and 30, in 1,000 trials the system of equations (4.5)-(4.7) and (5.3)-(5.5) had a solution for between 8 and 14 percent of all inputs, independent of the size of r. The remainder of the analysis proceeds as in Section 4.1.

Remark 5.2. Since the endomorphism rings of the abelian varieties constructed by Algorithm 4.3 are Dedekind domains, by Proposition 2.6 if r is unramified in $\mathcal{O}_K$ then the variety constructed has full embedding degree k even if the characteristic polynomial of Frobenius has multiple roots. Thus while Algorithm 5.1 is stated for $k \geq 4$, it also works for $k = 2$.

5.1 Constructing Genus 2 Curves C with $\mathrm{Jac}(C)[r] \subset \mathrm{Jac}(C)(\mathbb{F}_q)$

It may happen that for some protocols we require all of the r-torsion points of our pairing-friendly abelian surface A to be defined over a prime field, i.e. A to have full embedding degree 1. Since Algorithm 5.1 requires k to be even, we must find a different means of constructing such abelian surfaces.

Since the abelian surfaces we construct via Algorithm 4.3 have endomorphism rings equal to rings of integers in number fields, we may apply Proposition 2.6 to determine when such a surface has embedding degree 1. Given a prime r, we seek characteristic polynomial $h(x)$ such that $h(x) \equiv (x-1)^4 \pmod{r}$. In the notation of (5.2), this means that we have $q \equiv 1$, $s \equiv 4$, and $t \equiv 6 \pmod{r}$. Substituting these values into expressions (4.5), (4.6), and (4.7), we see that the left hand sides of equations (4.5) and (4.7) both become zero. Thus if we set $w \equiv 0 \pmod{r}$, we need only find a solution (u, v) to

$$bu^2 + bdv^2 + 2auv = 4 \pmod{r}. \tag{5.6}$$

This single equation in two variables gives an extra degree of freedom as in Algorithm 4.2, allowing us to loop on the value of v until a solution is found. The complete algorithm is as follows.

Algorithm 5.3. The following algorithm takes as input four positive integers a, b, d, r and a (finite) interval $I \subset \mathbb{Z}$, such that $K = \mathbb{Q}(\sqrt{-a + b\sqrt{d}})$ is a primitive quartic CM field, and r is prime. The algorithm outputs either the symbol $\perp$ or a prime q and a polynomial $h(x)$ of the form (5.2). If the output is not $\perp$, then there is a genus 2 curve $C/\mathbb{F}_q$ such that

- $\mathrm{Jac}(C)$ has characteristic polynomial of Frobenius $h(x)$,
- $\mathrm{Jac}(C)$ has endomorphism ring isomorphic to $\mathcal{O}_K$, and
- $\mathrm{Jac}(C)[r] \subset \mathrm{Jac}(C)(\mathbb{F}_q)$.

1. Set $v' \leftarrow 0$.
2. Using v' as the value of the variable v, find a solution u' to equation (5.6) modulo r. If none exists, go to Step 5.
3. Let u_0, v_0 be the unique integers in $[0, r)$ congruent to u', v' respectively.
4. For each triple $(i_1, i_2, i_3) \in I \times I \times (I \cap \mathbb{Z}_{>0})$, do the following:
   (a) Set $u \leftarrow u_0 + i_1 r$, $v \leftarrow v_0 + i_2 r$, $w \leftarrow i_3 r$.
   (b) Compute q, s, and t by equations (4.11), (4.6), and (4.10), respectively.
   (c) If t and q are integers, q is prime, and $q \nmid t$, go to Step 6.
5. Set $v' \leftarrow v' + 1$. If $v' \equiv 0 \pmod{r}$ then output $\perp$; otherwise go to Step 2.
6. Output q and the polynomial $h(x) = x^4 - sx^3 + tx^2 - sqx + q^2$.

We can then use Algorithm 4.3 to construct the genus 2 curve C whose Jacobian has the desired properties. An example with a prime r of cryptographic size appears in Appendix B. Since equation (5.6) is quadratic in u, we expect to find a solution u' for half of all values v'; the remainder of the analysis proceeds as in Section 4.1.

We note that Proposition 2.6 requires that the prime r be unramified in the specified CM field K. Since r is large (currently at least 2^{160}) and current methods only allow us to work with CM fields with very small discriminant, this condition will always be satisfied in practice.

6 Extending the Algorithms

6.1 Composite-Order Subgroups

Recently, a number of protocols have been proposed that require a pairing-friendly abelian variety with a subgroup r whose order is a large composite number that is presumed to be infeasible to factor, such as an RSA modulus (see e.g. [3]). We observe that our algorithms extend readily to produce such varieties. If we choose the desired subgroup size $r = r_1 r_2$ and find appropriate $(q_i', s_i', t_i', u_i', v_i', w_i')$ modulo r_i for $i = 1, 2$, we can use the Chinese Remainder Theorem to compute the values of the parameters modulo r, which we then lift to the integers in the usual manner. An example where r is a product of 512-bit primes appears in Appendix A.

6.2 Improving the ρ-Values

Algorithms 4.2 and 5.1 produce pairing-friendly abelian varieties with ρ-values around 8. This ρ-value means that computations on the abelian variety A must be done over a field whose size (in bits) is four times the size of the prime-order subgroup. Since the fields of definition of abelian surfaces can have as little as half as many bits as group sizes, our large ρ-value implies that arithmetic in the order-r subgroup of A will be significantly less efficient than if r were the full order of $A(\mathbb{F}_q)$.

An important open problem is thus to produce genus 2 curves $C/\mathbb{F}_q$ whose Jacobians are ordinary and pairing-friendly with respect to subgroups of order $r \approx q$ (i.e. $\rho \approx 2$) or even $r \approx q^2$ (i.e. $\rho \approx 1$). Just as there are a multitude of techniques for producing elliptic curves with $\rho < 2$ [8], there may be many different ways to generate abelian surfaces with $\rho < 2$. The results presented in this paper suggest two possible approaches; these ideas are the basis of ongoing research.

The Brezing-Weng Extension. Our construction of pairing-friendly genus 2 curves is modeled on the Cocks-Pinch method for constructing pairing-friendly elliptic curves [5], which produces curves with $\rho \approx 2$. Brezing and Weng [4] generalized the Cocks-Pinch method to produce elliptic curves with $\rho < 2$ by working over a number field L instead of modulo the prime r. Our method invites a similar generalization.

A Brezing-Weng-like construction for abelian surfaces would look something like the following: choose an embedding degree k and a primitive quartic CM field K. Let $L = \mathbb{Q}[x]/(r(x))$ be an extension of K that contains the kth roots of unity. Consider equations (4.5) through (4.10) as having coefficients in L, and find a simultaneous solution (q', s', t', u', v', w') in L^6. Represent these solutions as polynomials modulo $r(x)$ and lift to $\mathbb{Q}[x]$ to compute polynomials $q(x)$, $s(x)$, and $t(x)$. Then for any x_0 for which $q(x_0)$, $r(x_0)$, $s(x_0)$, and $t(x_0)$ take on integer values and $q = q(x_0)$ is prime, we can use Algorithm 4.3 to produce a genus 2 curve C over $\mathbb{F}_q$ whose Jacobian has embedding degree k with respect to $r(x_0)$.

While the setup is straightforward, it is far from obvious how to choose a number field L so that the relevant equations have solutions in L. It is also unclear what ρ-values the construction would produce.

The MNT Extension. The first construction of pairing-friendly ordinary elliptic curves was given by Miyaji, Nakabayashi, and Takano [23], who constructed curves of prime order with embedding degree $k = 3$, 4, or 6. While the MNT method is fundamentally different from our approach in this paper, our hope is that the results established here – in particular the relationship between the CM field and the characteristic polynomial of Frobenius described by Proposition 3.3 – will lead to a generalization of the MNT equations that produces abelian surfaces of prime order with small embedding degree.

Acknowledgments. The problem of constructing pairing-friendly genus 2 curves was first suggested to me by Kristin Lauter. The results presented in this paper were inspired by discussions with Dan Boneh and Edward Schaefer in the winter of 2006-07. I thank Ken Ribet for useful discussions, and I thank Steven Galbraith, Laura Hitt, Tanja Lange, Kristin Lauter, Edward Schaefer, and Edlyn Teske for helpful feedback on earlier drafts. This research was supported by a National Defense Science and Engineering Graduate Fellowship.

References

1. Balasubramanian, R., Koblitz, N.: The improbability that an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm. Journal of Cryptology 11, 141–145 (1998)
2. Bernstein, D.: Elliptic vs. hyperelliptic, part 1. Talk at ECC 2006, Toronto, Canada (20 September 2006) Slides available at http://cr.yp.to/talks/2006.09.20/slides.pdf
3. 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)
4. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Designs, Codes and Cryptography 37, 133–141 (2005)
5. Cocks, C., Pinch, R.G.E.: Identity-based cryptosystems based on the Weil pairing (Unpublished manuscript 2001)

6. Eisenträger, K., Lauter, K.: A CRT algorithm for constructing genus 2 curves over finite fields. In: AGCT-11, 2007 (to appear), preprint available at http://arxiv.org/abs/math.NT/0405305
7. Freeman, D., Lauter, K.: Computing endomorphism rings of Jacobians of genus 2 curves over finite fields. In: Symposium on Algebraic Geometry and its Applications, Tahiti 2007 (to appear), preprint available at http://eprint.iacr.org
8. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Cryptology eprint,2006/371, available at http://eprint.iacr.org
9. Frey, G., Lange, T.: Fast bilinear maps from the Tate-Lichtenbaum pairing on hyperelliptic curves. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. LNCS, vol. 4076, pp. 466–479. Springer, Heidelberg (2006)
10. Galbraith, S.: Supersingular curves in cryptography. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 495–513. Springer, Heidelberg (2001)
11. Galbraith, S., McKee, J., Valença, P.: Ordinary abelian varieties having small embedding degree. Finite Fields and Their Applications (to appear), preprint available at http://eprint.iacr.org
12. Gaudry, P., Houtmann, T., Kohel, D., Ritzenthaler, C., Weng, A.: The 2-adic CM method for genus 2 curves with application to cryptography. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 114–129. Springer, Heidelberg (2006)
13. Goren, E., Lauter, K.: Class invariants for quartic CM fields. Annales Inst. Fourier, preprint (to appear), available at http://arxiv.org/abs/math/0404378
14. Hitt, L.: Families of genus 2 curves with small embedding degree. Cryptology eprint 2007/001, available at http://eprint.iacr.org
15. Hitt, L.: On the minimal embedding field. In: Pairing 2007 (to appear), preprint available at http://eprint.iacr.org
16. Howe, E.: Principally polarized ordinary abelian varieties over finite fields. Trans. Amer. Math. Soc. 347, 2361–2401 (1995)
17. Katz, N.: Serre-Tate local moduli. In: Surfaces algébriques (Sém. de géom. algébr. d'Orsay 1976-78), Springer Lect. Notes in Math., exposé V-bis, vol. 868, pp. 138–202 (1981)
18. Lange, T.: Elliptic vs. hyperelliptic, part 2, Talk at ECC 2006, Toronto, Canada (20 September, 2006), slides available at http://hyperelliptic.org/tanja/vortraege/ECC_06.ps
19. Luca, F., Mireles, D., Shparlinski, I.: MOV attack in various subgroups on elliptic curves. Illinois J. Math. 48, 1041–1052 (2004)
20. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory 39, 1639–1646 (1993)
21. Mestre, J.-F.: Construction de courbes de genre 2 à partir de leurs modules. In: Effective methods in algebraic geometry, Birkhäuser Progr. Math. vol. 94, pp. 313–334 (1991)
22. Milne, J.S.: Abelian varieties. In: Cornell, G., Silverman, J. (eds.) Arithmetic Geometry, pp. 103–150. Springer, Heidelberg (1986)
23. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Transactions on Fundamentals E84-A, 1234–1243 (2001)
24. Oort, F., Ueno, K.: Principally polarized abelian varieties of dimension two or three are Jacobian varieties. J. Fac. Sci. Univ. Tokyo Sect. IA Math. 20, 377–381 (1973)

25. Paterson, K.: Cryptography from pairings. In: Blake, I.F., Seroussi, G., Smart, N.P. (eds.) Advances in Elliptic Curve Cryptography, pp. 215–251. Cambridge University Press, Cambridge (2005)
26. Rubin, K., Silverberg, A.: Supersingular abelian varieties in cryptology. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 336–353. Springer, Heidelberg (2002)
27. Spallek, A.-M.: Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen. Ph.D. thesis, Institut für Experimentelle Mathematik, Universität GH Essen (1994)
28. Tate, J.: Classes d'isogénie des variétés abéliennes sur un corps fini (d'après T. Honda), Séminaire Bourbaki 1968/69, Springer Lect. Notes in Math. exposé 352, vol. 179, pp. 95–110 (1971)
29. van Wamelen, P.: Examples of genus two CM curves defined over the rationals. Math. Comp. 68, 307–320 (1999)
30. Waterhouse, W.C., Milne, J.S.: Abelian varieties over finite fields. Proc. Symp. Pure Math. 20, 53–64 (1971)
31. Weng, A.: Constructing hyperelliptic curves of genus 2 suitable for cryptography. Math. Comp. 72, 435–458 (2003)

A Appendix: Examples of Abelian Surfaces with Prescribed Embedding Degree

We implemented Algorithm 4.2 in MAGMA and ran the program on a Red Hat Linux system with a 2.38 GHz AMD Opteron processor and 4 GB of RAM. To speed up the computations we used probabilistic primality testing throughout.

Example 1. We used the CM field $K = \mathbb{Q}(\sqrt{-2+\sqrt{2}})$ to construct a curve whose Jacobian has embedding degree 2 with respect to $r = 2^{160} + 7$. The Igusa class polynomials for K can be found in [29]. Algorithm 4.2 ran in less than 1 second and output the following:

$q = 795006611640170109396940876005774396116863415419758542983000866861998633580771737$ \
$977185988060481042862469026090643969667638364464302415656507943863305115226587117$ \
$936072460021623269435928862304096161$ (651 bits)

$s = 241065221491947514428541310368448574139558370891656283357513064453386954760732987$ \
4103585110310902524

$t = 273597961733915219746412647988034912157244791593900712761392170842037137177036567$ \
$553396874299843349115422505366307475408332787349620259128899954674524332906359097$ \
$80374584172196269739884391025564649667.$

The equation of curve C is $y^2 = x^5 + a_3x^3 + a_2x^2 + a_1x + a_0$, with

$a_3 = 781556463828009280287360245134696723363334003262680019563373246663399403283036487$ \
$052339182967247341574259440091198071796180846063637213566649478498289206351995367$ \
$383165038349582750830436766970252305$

$a_2 = 758207154481521945617036644682392283114949510705878088132268608920626188933438797$ \
$483381881227773991187422405413699207810216146066184913867557690015136357020025837$ \
$900328984724486510381446751384612338$

$a_1 = 38180895173516496160634313784225571603708056687712617826249486612556118040311047 \backslash$
$28412516481837472216648106219404759483953015501490776665696628159270771565309 42 \backslash$
$39443567989044899842823923822 4064322$
$a_0 = 45177931133803554760365209853147386766975669710479527089607077734830321391815260 \backslash$
$19191380637561071045120827733864057302909106384439009009025246738012848274281139 \backslash$
$75208548420453372632684615214795613 0.$

The ρ-value of Jac(C) is 8.135.

Example 2. We used the CM field $K = \mathbb{Q}(\sqrt{-13+3\sqrt{13}})$ to construct a curve whose Jacobian has embedding degree 5 with respect to $r = 2^{256} + 1935$. The Igusa class polynomials for K can be found in [29]. Algorithm 4.2 ran in 3.0 seconds and output the following:

$q = 18705441730027288882908175810362262525180013527543429746883463492194609246911 30 \backslash$
$92072159086600150762181774510677124635515553318814158762548930085510167332573 21 \backslash$
$54412705897520775220066584322620164682082819849616622456096411918995649988546 47 \backslash$
$18489846985003356378154220307855272110401992016598515195942158384281100933489$
(1041 bits)
$s = 16219975780701742222835079271431305313350557798973620833166819306234378027012 75 \backslash$
$40335053439923145467000742770656935732638003623655485310513110055779056693512 36$
$t = 10291442194559070987163484703477341401646114776334728312660671825705004992614 94 \backslash$
$29943907533362944843972293978704122839900895094762253975101893873019512745957 94 \backslash$
$61083412822750684992156808871606294716943081223243697206228345159438753569052 73 \backslash$
$82882607370079126422730523333067568351208494587747330441954373352761499629373 4.$

The equation of curve C is $y^2 = x^5 + a_3x^3 + a_2x^2 + a_1x + a_0$, with

$a_3 = 17711932034826598918283689493771517557773876172909860411136075106337245263908578 \backslash$
$38130632676662412318317995526768766016156509218023316817809243032636104864542135 \backslash$
$90450974675151214252356394141967359391373046477845725484335245940593602161185522 \backslash$
$99407180726519279323555530065745247279169841864254113262083113263075718295$
$a_2 = 15986676510291987692234115367676532911850944091016679833071471047947234683585606 \backslash$
$97115034669874820450535884159056843728249225669815188503869445836591380801629 09 \backslash$
$05967274844595441965601950514813102729723644095411071587133554049011997599748427 \backslash$
$04612514672592478589233445454956670634552105995792263972845133993646857289$
$a_1 = 47201942823478063845886098041135725187242793436198889585030352165809645065572147 \backslash$
$21227820016294216571118639320025552488751294747094636654493682076253265791893965 \backslash$
$45479038178461819213500842438576057436074730626945979688825026746690602467861501 \backslash$
$22818186813590260409963846589232079707065644640692839018443390682626873 54$
$a_0 = 78304034375067256115734447740946659290071046170720112481596930213348431536496457 \backslash$
$55121110421815824365159812788162205652360709731948183507741581532695550495070587 \backslash$
$92454247668794075954692123006270938295625578774958097004925464585575953412442322 \backslash$
$12289080772802589930714322619719426885901604420229918101878858983346214 34.$

The ρ-value of Jac(C) is 8.130.

Example 3. We used the CM field $\mathbb{Q}(\zeta_5) \cong \mathbb{Q}(\sqrt{-5+2\sqrt{5}})$ to construct a curve whose Jacobian has embedding degree 2 with respect to a subgroup whose

order is the product of two randomly chosen 512-bit primes. The subgroup order is $r = r_1 r_2$, where

$$r_1 = 1180397868977793794363060648291663061077126236045103899805583954032652977066708 4 \backslash$$
$$0626954383484369579719868476824117153911720219574579837991644798160290425 51$$
$$r_2 = 1056214811242302041652440469475787736421437930439874230046281076619099420655483 7 \backslash$$
$$2078029786833382927371341816153271065505776589093000561750641434076122011 71.$$

The curve parameters output by the algorithm are

$$q = 1492575648439562058834082896167083825751347870666156569724682245250456655214057 4 \backslash$$
$$7427566870618418332723060848378371236647475854074621702866748764091196214098609 3 \backslash$$
$$5990378087560222470123272151775521362915575919809545431055802787421465282089778 0 \backslash$$
$$4833880605860664882186871812124523076254833497636649612455448398148337380182250 3 \backslash$$
$$1875872580636600862294107271176414159884670132313636166306092678456567866453825 0 \backslash$$
$$2387471811606710097747398591075467158303722446780859641674975847514971331906247 3 \backslash$$
$$7439910476426062081740575220503979494601931594617886637373038703700255950867989 1 \backslash$$
$$4435773073309574410347331834498300426876700767027024271973767911839362768322041 6 \backslash$$
$$7560500941974369069329083597086702358083871770413108550248587688844368340086636 8 \backslash$$
$$6434547994549615674931751552305085319169667752483361340513512981048962132966023 8 \backslash$$
$$8924453978024513608120925596409216323397648311343976954948506089625385935873433 1 \backslash$$
$$1993015139211462964161036467119410583324398159603028447384946960622522307709976 0 \backslash$$
$$8659471920286521205949551151597752968638719758638121844984366957688187107935204 2 \backslash$$
$$6946543159746899834639602301095209882606155611802028587711772509703238644061650 4 \backslash$$
$$2850259998938610782156353858678210524786864051502803963345888644673903158542867 8 \backslash$$
$$9262735572775970427343989598147871137491 \quad (4117 \text{ bits})$$
$$s = 1309874687896243644001443068389133668331718957149050591481327212081161430932444 4 \backslash$$
$$3033668713542937283000597953750481131930422787663654529415280088672148119481941 1 \backslash$$
$$4462413535325124807388410862800590178754296141968972469468748733633461996868279 0 \backslash$$
$$2103960783912350458055265202939110510975092005113145069782389653457523785799483 0 \backslash$$
$$5629869325071847516225158155761982641414750098401505143681328117709799445680176 1 \backslash$$
$$9717444275608172078749472606181869543550565428209008060059063173556446213346486 1 \backslash$$
$$7316281842768061133041911713805706898299076717788274615207566114434186640032015 1 \backslash$$
$$8414345765447763605149300871290591470423293221894274883969124$$
$$t = 7144787435191164904779058655809099399856326860349020002395737142280155516293768 5 \backslash$$
$$0729600261945916949895279824782323405035812478980954417856736951579004108911519 \backslash$$
$$1637490664834389911656043491104413002063355420970497939326080907921789352546461 7 \backslash$$
$$6975399490776753978123408657431186142327053633257047873978390520763369497116347 1 \backslash$$
$$1875787050985548173234719937091580295004771209568637110461794429344364328801553 0 \backslash$$
$$2722495832002038416645798872040150035192129637132500312620591009764319887383146 9 \backslash$$
$$6059084370667725537893545979164129137199593649188911399007565699575224418656294 0 \backslash$$
$$8357922063308304561090460928558473784246061491919591295824435090851185586027959 5 \backslash$$
$$5990660755981192038000549557471498433918339646781784527503596154897409684735772 6 \backslash$$
$$1186810526498627348121318287605849558734730732838428885168809232055332529526791 5 \backslash$$
$$6487993614335873937631278137023958484222652378109937457665790329004114004236784 2 \backslash$$
$$5530874894613633379544708621873764938067722197469475912611967969911573373251062 5 \backslash$$
$$1009118068517774538325440179740638841946741999971551345126799287983731744315849 8 \backslash$$

023383168900668022449948705091437624672716611253196204354197468514551516198067 00 \
93306648871023302649907691217825348504456208073558963082634045539270253362741192 \
27548353972566258529006993215737180564 06.

The equation of the curve C is $y^2 = x^5 + 1$. The ρ-value of $\mathrm{Jac}(C)$ is 8.044.

B Appendix: Examples of Abelian Surfaces with Prescribed Full Embedding Degree

We implemented Algorithms 5.1 and 5.3 in MAGMA and ran the programs on a Red Hat Linux system with a 2.38 GHz AMD Opteron processor and 4 GB of RAM. Again, we used probabilistic primality testing throughout.

Example 4. We used the CM field $K = \mathbb{Q}(\sqrt{-30 + 2\sqrt{5}})$ to construct a curve whose Jacobian has full embedding degree 4 with respect to $r = 2^{224} - 3047$. The field K is non-Galois and has class number 4. The Igusa class polynomials for K can be found in the preprint version of [31]. Algorithm 5.1 ran in 9.0 seconds and output the following:

$q =$ 25875665546464625747483263904141863215223855685631927398442175454186047689727181 \
935477877624034595615954645104592718893686922526984289911563166040265258306653 11 \
737526317236739759816581034931796803081221823586560157806990386975210310942567 55 \
40495783228843733030724545875768981 (912 bits)

$s =$ −49819073164368546886826258467120852760074966205236328690105833376230198398121 96 \
710646403289755093635995976110325660432316004402715132455 36

$t =$ 11315891237651345494483496588271771338331199214236780785739679696556702185995706 \
34372064002496085550624521210317464488708701543703195565346155063791365178640664 \
14606889659037650402335705765920362951147827106493530098874861489026762557535530 \
48773420611254748702256459338146056 6.

The equation of curve C is $y^2 = x^5 + a_3x^3 + a_2x^2 + a_1x + a_0$, with

$a_3 =$ 38557522515431718239138775060871205148933533072861261987872384836301846569735691 \
98659203885183971732172895101214552565189586386075463460413300321595163673236547 \
01858086829402659012929942804221720007434564827463546641733443182389934765013564 \
81279923766002824202682910288030 21

$a_2 =$ 24361005015196617163667303200244929565396754003199827807036659742558068205679538 \
42160098098260878264842948120639524037265934211880780488463539945820106947877300 \
19189022859405786742281608167622704227996495252854882617846133996598330042941638 \
79810835474597261493353081510971860

$a_1 =$ 66320501401668844356948887655622054099995573466241416241541350131153121137993090 \
85635251499283342087242563793741047433071997209515897943786177629734199317620050 \
78838031904755852619295711922546671672538134189634737065276752833417635898769501 \
63791039986027798808881283158482 57

$a_0 =$ 14382706785238411696604261079830832924182682159103069238604604235474798613117520 \
63172956299590491867116182929144152136870787179261898874121788589210503017604127 \
07396812946751792677163628751267190836563349333071193338619515991035776086979757 \
14869063111066962733459404280651443.

The ρ-value of $\mathrm{Jac}(C)$ is 8.139.

Example 5. We used the CM field $\mathbb{Q}(\zeta_5) \cong \mathbb{Q}(\sqrt{-5+2\sqrt{5}})$ to construct a curve whose Jacobian has full embedding degree 18 with respect to $r = 2^{512} - 21765$. It is well known that if q is a prime congruent to 1 modulo 5, then curves of the form $y^2 = x^5 + a$ over $\mathbb{F}_q$ have ordinary Jacobians with endomorphism ring equal to the ring of integers in $\mathbb{Q}(\zeta_5)$. Algorithm 5.1 ran in 118 seconds and output the following:

$$
\begin{aligned}
q = {} & 14555061271614284829374548810808668921344787153336861961038950978287550386155088 \\
& 27761408657449714149293817644592128041703058419247667367620955457503763764714368 \\
& 14187932866719028481645089094069906583996373002241963091289099300688688994901771 \\
& 83873675731615158570245932236872128937154479692085273245058632104897428914803561 \\
& 29605998354055071775377646204638906630542040592646748390702920765239663750378074 \\
& 83299238315928230714527086687127082551072561286442424934953309068386901100135192 \\
& 92807522477039053315542103935756755589377186962105535601164019023329205199251303 \\
& 42747200824596312881501890045322210553967105952772694133261 \quad (2061 \text{ bits}) \\
s = {} & 43752127389092211668913731818257083380184185029518072697970279699709038554595574 \\
& 67775441465528134480249170813639910902216349834259532905575148957826052783432254 \\
& 79119405645969200966206278121318902156015403475862583609211968197706368610425979 \\
& 74826227341593725640985162876416189413727067617925373652301554222041356 \\
t = {} & 76506345185413572894000615027118972245682316768376511436322271820887135834751430 \\
& 01408196970280322848312865810412722709808538881978637395656622842478438983838876 \\
& 26929204429568746779543344427174688447427513177017872136990404463200335646893378 \\
& 01074535914928884192396518493957853498563753295379600060198999275134256487324441 \\
& 89371513845650103152259540955370064472019241067489625256118717229482895166343889 \\
& 42663816081413289131028669600536087288481804865750637595013957630684845995843479 \\
& 83683439139891802705356974591743005414154458649393619533536954826980321148967937 \\
& 74765143135478758860646527654648060969149548725927379518330 86
\end{aligned}
$$

The equation of the curve C is $y^2 = x^5 + 32$. The ρ-value of $\mathrm{Jac}(C)$ is 8.047.

Example 6. We set $r = 2^{192} - 237$ and used the CM field $K = \mathbb{Q}(\sqrt{-13+2\sqrt{13}})$ to construct a curve $C/\mathbb{F}_q$ with $\mathrm{Jac}(C)[r] \subset \mathrm{Jac}(C)(\mathbb{F}_q)$. The Igusa class polynomials for K can be found in [29]. We used the probabilistic algorithm of Freeman and Lauter [7, Algorithm 4.3] to confirm that $\mathrm{Jac}(C)[r] \subset \mathrm{Jac}(C)(\mathbb{F}_q)$. Algorithm 5.3 ran in less than 1 second and output the following:

$$
\begin{aligned}
q = {} & 41917988492119141249022446188487568991935922462150808494416316158553598932697 73 \\
& 83272842381465331789525955141195441340428996707998134865330379302251402611349 72 \\
& 18485837183300638482574886142860696370694527884195065961476308415619112128 1 \\
& (773 \text{ bits}) \\
s = {} & 15760802478557791684911616040057445522031895708186178665984157191280664691164 54 \\
& 99383869668163280777796966990248899856 \\
t = {} & 93151085538042536108938769307750153315413161027001796654086997940653311817935 24 \\
& 78682802404960938177148354133899993271994641255739054712601215835633030023875 32 \\
& 19243748436256787827661246671324707727932944167299366404935886919307341774.
\end{aligned}
$$

The equation of the curve C is $y^2 = x^5 + a_3x^3 + a_2x^2 + a_1x + a_0$, with

$$\begin{aligned}
a_3 &= 28932282521819038280326666503307020611152548216399078321215512935546100541046 50 \backslash \\
&\quad 02364962806545504432997892720744589983358536928902319593105805009984556203183 82 \backslash \\
&\quad 52471270856385978986255881796236347205721596907069183349632688451888980843 0 \\
a_2 &= 18519126403250094965953802347785265408499724718736649019824988296072976579694 34 \backslash \\
&\quad 16034423386875812079107032339048677040568945619421747121612569346133509299587 34 \backslash \\
&\quad 16522531237666462484342201598945244051808496141043008227314651611993479002 1 \\
a_1 &= 51004522827135545868055349145766706573208071472631658403414584258588696715162155 \backslash \\
&\quad 27271154819467166690055569792351605027204302869720462405330228004753949945747 89 \backslash \\
&\quad 94684865345941988652933629587647405581943934960966405623978471953839825408 \\
a_0 &= 33277218609785779101614036830691708370515017230158427310490879303381054908563 00 \backslash \\
&\quad 21075933450284269249263947617171897597598045102922866075405839290140983089035 11 \backslash \\
&\quad 30759428669945350702323198089179219279729580597842288237226415502850187806 2.
\end{aligned}$$

The ρ-value of $\mathrm{Jac}(C)$ is 8.050.

Implementing Cryptographic Pairings

Michael Scott

School of Computing
Dublin City University
Ballymun, Dublin 9, Ireland
mike@computing.dcu.ie

Abstract. Here we review the state-of-the-art in cryptographic pairing implementation. Starting with a basic Miller algorithm for the Tate pairing we show how to successively apply a series of optimizations and tricks to improve performance. We will concentrate on the case of non-supersingular prime characteristic elliptic curves, although many of the optimizations equally apply to the cases of supersingular elliptic and hyperelliptic curves. We also discuss optimal implementation of extension field arithmetic.

Keywords: Tate pairing implementation, pairing-based cryptosystems.

1 Introduction

The Tate pairing, denoted $e(P,Q)$, where P and Q are linearly independent points on an elliptic curve $E(\mathbb{F}_{q^k})$ evaluates as an element of an extension field $\mathbb{F}_{q^k}$. If P is of prime order r, then the pairing evaluates as an element of order r. This feature can be used to transfer the discrete logarithm problem from the elliptic curve setting, to the easier finite field setting, an ability exploited by Menezes, Okamoto and Vanstone [32], and by Frey, Müller and Rück [16], to solve the elliptic curve discrete logarithm problem in certain instances. The embedding degree k is defined as the *smallest* value of k such that $r|q^k - 1$. For a random non-supersingular curve this is unlikely to be the case for a small value of k. However for supersingular curves and specially constructed pairing friendly non-supersingular curves, k will be small. Here we focus on the case of non-supersingular elliptic curves of prime characteristic, so from here on $q = p$. Some actual implementations of pairings are now available [31], [42].

The most useful property of the pairing is its *bilinearity*

$$e(aP, bQ) = e(P,Q)^{ab}$$

But how to calculate the pairing? In the beginning there was Miller's algorithm. Here we start with a generic implementation in the context of calculating the Tate pairing (which appears to be superior to the Weil pairing for all cases of interest). As is well known in this case a "final exponentiation" is required to obtain a unique result. See algorithm 1.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 177–196, 2007.

Algorithm 1. Computation of $e(P,Q)$ using basic Miller's algorithm

INPUT: $P \in E(\mathbb{F}_{p^k}), Q \in E(\mathbb{F}_{p^k})$, where P has order r
OUTPUT: $e(P,Q)$
1: $T \leftarrow P$, $f \leftarrow 1$
2: **for** $i \leftarrow \lfloor \lg(r) \rfloor - 1$ **downto** 0 **do**
3: $\quad f \leftarrow f^2.l_{T,T}(Q)/v_{2T}(Q)$
4: $\quad T \leftarrow 2T$
5: $\quad$ **if** $r_i = 1$ **then**
6: $\quad\quad f \leftarrow f.l_{T,P}(Q)/v_{T+P}(Q)$
7: $\quad\quad T \leftarrow T + P$
8: $\quad$ **end if**
9: **end for**
10: $f \leftarrow f^{(p^k-1)/r}$
11: **return** f

Observe that the point P is being implicitly multiplied by its group order r using a classic double-and-add line-and-tangent algorithm, until it finally ends up at the point-at-infinity. The values of the line and vertical functions $l_{A,B}(Q)$ and $v_{A+B}(Q)$ respectively, are distances calculated between the fixed point Q and the lines that arise when adding B to A on the elliptic curve in the standard way. If the point A has coordinates (x_j, y_j), the point $A + B$ has coordinates (x_{j+1}, y_{j+1}), the point Q has the coordinates (x_Q, y_Q), and the line through A and B has a slope of λ_j, then explicitly

$$l_{A,B}(Q) \leftarrow (y_Q - y_j) - \lambda_j(x_Q - x_j)$$
$$v_{A+B}(Q) \leftarrow (x_Q - x_{j+1})$$

Here we assume the use of affine coordinates, although in most cases projective coordinates are to be preferred, with minor modifications to these formulae.

The eagle-eyed may have spotted that the pairing seems to require another parameter, the order r of P. In fact knowledge of r is not strictly required, as the pairing can be calculated without it. Simply omit the final exponentiation, and replace the number of iterations of the Miller loop with $\lg(p^k - 1)$ instead of $\lg(r)$. In fact the work load can be shifted between the Miller loop and the final exponentiation by replacing the number of iterations in line 2 with $\lg(mr)$, and the final exponentiation by $(p^k - 1)/(mr)$, for any m for which $(p^k - 1)/(mr)$ is a whole number. However it is usual that r is known, in which case concentrating as much as the work as possible into the final exponentiation will be faster.

In fact the algorithm as described may fail catastrophically for random choices of P and Q, as it is quite possible that the tangent or addition line may pass directly through Q, or the vertical function may evaluate as zero. In the sequel this possibility will be side-stepped by choosing P and Q from particular disjoint groups.

Our first couple of optimizations are simple and rather obvious [2], [18]. First we will (to the extent that we can) choose r to have a low Hamming weight (or

indeed mr if that has a lower hamming weight for small m). This will clearly result in a faster algorithm. In some cases (e.g. for the MNT curves [34]) the choice of a low Hamming weight order may not be practical, in which case the optimal strategy might be to represent r in a NAF format, and use a standard windowed NAF double-add-and-subtract algorithm as used for standard elliptic curves. Alternatively the method of Eisentrager et al. [14] might be useful. This will lead to a more complex algorithm that would muddy the waters for us in what we are trying to do here. Therefore we will proceed on the assumption that a low hamming weight r is possible, and even if it is not the further optimizations that we will describe are still applicable.

Secondly we will choose P to be a point on the curve taken over the base field $E(\mathbb{F}_p)$ (Solinas's *Miller light*), with obvious performance advantages. A useful side effect is that the algorithm failure described above cannot now happen (for $k > 1$) if Q is a general point over the full extension field.

We will however resist the adoption of an optimization once suggested by Koblitz and Menezes that the modulus p should also be chosen as being of low Hamming weight as this can introduce plausible security concerns [36].

2 A Restriction

Before proceeding we will introduce a useful restriction. From here we assume that k is even, and that $k = 2d$. As we will see this brings many advantages.

The final exponentiation can now be written as $f^{(p^d-1)(p^d+1)/r}$. (And we know that r must divide the $p^d + 1$ part, as otherwise the definition of k above would be violated). As we will see this simple observation facilitates many of our optimizations.

3 Extension Field Arithmetic

Before proceeding to optimize the pairing algorithm itself, we pause to consider the implementation of extension field arithmetic. This has arisen before in the context of cryptography, notably in the XTR scheme [28], and for elliptic curves implemented over Optimal Extension Fields [1]. However in the former case only quadratic extensions are considered, and in the latter the context is rather specialised by the preference for a small word-sized modulus p of simple form. A much wider range of possible extension fields need to be considered in the context of pairings.

First we need a suitable *irreducible polynomial*, which has a degree the same as our chosen extension field, but which has no factors over the base field.

As an example consider the case where $k = 2$. In this case if $p = 3 \bmod 4$, then we might choose as our irreducible polynomial $x^2 + 1$, because -1 is a quadratic non-residue modulo p, and so the polynomial does not factor over the base field. Now elements in $\mathbb{F}_{p^2}$ can be represented as polynomials like $a+xb$ with $a, b \in \mathbb{F}_p$. Such elements can be multiplied as normal, and then reduced modulo $x^2 + 1$. This means that the occurrence of x^2 in the product may be replaced

by -1. Note that x can be considered as the imaginary root of the irreducible polynomial, in which case elements of the extension field can be represented as $a+ib$, where i is the imaginary square root of -1, and the analogy with complex numbers is exact (and quite comforting!).

The addition and subtraction algorithms are quite obvious and cheap. Division (which turns out not to be time-critical) can be implemented only ever requiring one base field inversion [30]. It is however worth taking a little care over multiplication and squaring. We will assume that a base field squaring (called a modsqr) costs about 0.9 of a base field multiplication (a modmul), a figure typical of practical finite field implementations.

3.1 Multiplication and Squaring

The naive way to do multiplication is

$$(a+ib)(c+id) = ac - bd + i(bc+ad)$$

at a cost of 4 base field modmuls. But of course Karatsuba will be our friend in this setting, and so

$$(a+ib)(c+id) = ac - bd + i[(a+b)(c+d) - ac - bd)]$$

which only requires three modmuls. However a modmul can be divided into separate multiplication and reduction steps. For a prime characteristic field, using Montgomery's method [35] (and recalling that we have eschewed the possibility of a special form for the modulus p on the grounds that it may weaken our security), reduction will cost about the same as a multiplication. In this case the method of "lazy reduction" [30] can be applied, whereby we calculate the full values of the real and imaginary parts of our product before reducing them separately modulo p. In this way only two reductions are required in the above formula, and the overall cost falls to about that of 2.5 base field modmuls.

In the case of squaring we could also apply Karatsuba for a cost of two modsqrs and one modmul. However a better idea would be to use this identity well known to implementors of complex arithmetic

$$(a+ib)(a+ib) = (a+b)(a-b) + i.2ab$$

which requires just two modmuls. Note that for an alternative irreducible polynomial the same basic costs apply, perhaps with more modular additions and subtractions.

Turning next to the case of the cubic extension, multiplication can be carried out using Karatsuba and 6 modmuls. Using the method of Toom-Cook this can be reduced to 5 modmuls, with some tricky multiplications and divisions by small constants (and divisions are particularly difficult). The idea of lazy reduction can again be used here to advantage. For squaring the best algorithm was only very recently discovered by Chung and Hasan [9]. Assume an irreducible polynomial

of the simple form $x^3 + n$, and consider the calculation of $(a + bx + cx^2)^2$. First precalculate $A = a^2$, $B = 2bc$, $C = c^2$, $D = (a - b + c)^2$ and $E = (a + b + c)^2$, then

$$(a+bx+cx^2)^2 = (A-Bn)+((E-D)/2-B-nC)x+(E-A-C-(E-D)/2)x^2$$

which requires only 4 modsqrs and 1 modmul (and a division by 2 and a lot of adds/subtracts).

What about those annoying divisions by constants that occur for Toom-Cook and for the Chung-Hasan formula above? Recall that the final exponentiation in the Tate pairing involves exponentiation by a power of $p^d - 1$. This always contains as a factor $p-1$. Recall next Fermat's little theorem ($y^{p-1} \bmod p = 1$). This implies that we are free to include a constant factor into any extension field multiplication or squaring, on the basis that the contribution of this constant will be "wiped-out" in the final exponentiation. Therefore divisions by small constants can always be replaced by multiplications by small constants (in our context), which in turn can be replaced by additions.

3.2 Inversion

For completeness we include formulae for inversion in the quadratic and cubic extension fields. These are quite easy to derive [30].

For the quadratic case, and assuming an irreducible polynomial of $x^2 + n$

$$1/(a + bx) = (a - xb)/(a^2 + nb^2)$$

For the cubic case, and assuming an irreducible polynomial $x^3 + n$. First precalculate $A = a^2 + nbc$, $B = -nc^2 - ab$, and $C = b^2 - ac$. Then $F = -nbC + aA - ncB$, and

$$1/(a + bx + cx^2) = (A + Bx + Cx^2)/F$$

3.3 Square Roots

In many early protocols there was a requirement to hash values to curves points, e.g. Boneh and Franklin IBE [7]. This practise now seems to be deprecated in favour of hashing to point multipliers, which is much easier to do. However hashing to a curve point is perhaps not quite as complex as is thought. The issue of a "large cofactor" can be dealt with by exploiting the feature of the Tate pairing that it does not require the parameter Q to be of order r for bilinearity to hold (it is sufficient that Q be a representative of a coset – in effect any point on the curve) [38].

If hashing to a curve point there may be a requirement to hash to a point on the curve over an extension field. This usually requires the ability to extract square roots over the extension, although for some curves cube rooting may

be more useful [5], [7]. In the quadratic extension case (again borrowing from standard methods for complex arithmetic [17]) we have the nice identity

$$\sqrt{a+xb} = \pm(\sqrt{(a \pm \sqrt{a^2+nb^2})/2} + xb/(2\sqrt{(a \pm \sqrt{a^2+nb^2})/2}))$$

which is a lot simpler than the method suggested by Wang et al. [46]. Only two square roots are required over the base field. Whether or not $a+xb$ is a quadratic residue is determined solely by whether or not a^2+nb^2 is a quadratic residue (the proof left as an exercise to the reader), so the quadratic residuosity test is in this case particularly efficient.

For the cubic extension case no simple closed formula for the square root seems to be possible. Over the field $\mathbb{F}_{p^3}$ for $p = 3 \bmod 4$ we can offer the formula

$$\sqrt{u} = \pm(u^{p^2}(u^p)^3 u)^{(p-3)/4} u(u^p)^2$$

assuming of course that u is a quadratic residue. A similar formula requiring primarily the work of one simple exponentiation can also be derived for the $p = 5 \bmod 8$ case.

$$v = ((2u)^{p^2}((2u)^p)^5 2u)^{(p-5)/8}((2u)^p)^3$$
$$\sqrt{u} = \pm(uv(2uv^2-1))$$

For the remaining $p = 1 \bmod 8$ case a variant of the standard Tonelli-Shanks algorithm can be used [33]. The Frobenius (see below) is very helpful in deriving these formulae.

For a discussion of cube roots in quadratic extensions, see Appendix B of [5].

3.4 The Frobenius Action

Of particular importance in extension field arithmetic is the "Frobenius action". Raising an extension field element to the power of the modulus is always very cheap. Going back for a moment to our simple example above, we have the relationship

$$(a+ib)^p = (a^p + i^p.b^p) = (a - ib) \bmod p$$

This is quite easy to prove, the "other" terms is the expansion of $(a+ib)^p$ are all zero modulo p, and recall that $i^{(p-1)/2} = -1$ as i is a quadratic non-residue.

3.5 A Tower of Extensions

For higher extension fields, a "tower of extensions" can be used, and these methods for multiplication, squaring, inversion and square rooting can be used recursively. For example consider the sextic extension with the irreducible polynomial x^6+n. Using the Chung-Hasan idea and building a cubic extension on top

of a quadratic extension, a squaring in this field will only require 11 base field modmuls.

For most cases of interest we can further restrict $k = 2^i 3^j$ for $i \geq 1$ and $j \geq 0$ [27]. In these cases we only need to consider quadratic or cubic extensions of the field below.

However more work is needed to determine which is the best method to use for higher extension degrees. Which is better Karatsuba or Toom-Cook? How best to organise the towering? It seems that for MNT $k = 6$ curves that a quadratic layered on top of a cubic works best. For $k = 12$ BN curves (see below) a quadratic over a cubic over a quadratic works well in practise. Of course it is possible to switch from one representation to another, but in the interests of keeping the code elegant and minimizing code-size it would be nice to avoid it if possible. Extensive testing of the various alternatives has been carried out and reported in [11].

4 Another Restriction

Koblitz and Menezes [27] have proposed the use of "pairing-friendly" fields, where $p = 1 \bmod 12$ and the irreducible polynomial is of the form $x^k + \beta$, where $\beta \in \mathbb{F}_p$. In our view the $p = 1 \bmod 12$ condition is quite restrictive, and the value of β for a particular p can be quite large, leading to more additions at the bottom of the tower, and hence less efficient implementations.

Instead we recommend a type of construction like that proposed by Barreto and Naehrig [5] for use with the BN family of pairing friendly curves (see below). We will continue to insist that the irreducible polynomials at each level in the tower are binomial, and furthermore that the imaginary roots of these polynomials should contain the same roots as those used at the lower level. For example for $k = 12$ we could have an irreducible polynomial of the form $X^6 + (\alpha + \sqrt{-\beta})$ as a sextic extension built on top of a quadratic extension which has as an irreducible polynomial $x^2 + \beta$. This particular construction is ideal for the BN curves, allows the use of $p = 3 \bmod 4$ curves, and often permits the use of very small $|\alpha|, |\beta| \leq 2$.

5 Types of Curves

Pairing-friendly curves have two parameters of relevance to us here. The first ρ is the rounded-to-nearest-simple-fraction ratio of the size in bits of the modulus p to the size in bits of the group order r. Recall that the number of points on the elliptic curve over the base field is given by $\#E = p + 1 - t$, where t is the trace of the Frobenius and $|t| \leq 2\sqrt{p}$, and $r|\#E$. The second parameter ω is the rounded-to-nearest-simple-fraction ratio of the size in bits of the order r to the size in bits of the trace t. In general we would prefer a small ρ, with $\rho = 1$ being regarded as ideal, and a large ω.

For example consider the BN family of curves [5], a family with an embedding degree of $k = 12$, and with a very simple generation function

$$p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$$

$$\#E(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$$

$$t(x) = 6x^2 + 1$$

To find a pairing-friendly curve, simply choose an x of the appropriate size and check that $p(x)$ generates a prime. If $r(x) = \#E(x)$ is also prime, then in this case $\rho = 1$ and $\omega = 2$. The actual parameters of the curve can then be found using the method of Complex Multiplication [10]. Many other families of curves have also been discovered – see [15].

As a general truism it can be said that the closer ρ is to 1, the more difficult it becomes to force a low Hamming weight for r. On the other hand using a Cocks-Pinch pairing-friendly curve [6] for which $\rho = 2$, there is complete freedom in the choice of r. A value of $\rho = 1$ may however be mandatory in certain applications, for example short-signature schemes [8]. However it is our opinion that curves with $\rho = 2$ are still very usable (and there are a lot more of them), and their supposed inefficiency is often overstated (at higher levels of security in particular it is the extension field arithmetic which predominates in the calculation of the pairing, and it is the bit length of this extension field that matters, not the size of the base field from which it is constructed).

See below for the relevance of ω.

6 Optimizing the Miller Loop

The algorithm as it currently stands looks like this – see algorithm 2.

Algorithm 2. Computation of $e(P, Q)$ with some optimizations

INPUT: $P \in E(\mathbb{F}_p), Q \in E(\mathbb{F}_{p^k})$, where P has order r
OUTPUT: $e(P, Q)$

1: $T \leftarrow P$, $f \leftarrow 1$
2: **for** $i \leftarrow \lfloor \lg(r) \rfloor - 1$ **downto** 0 **do**
3: $\quad f \leftarrow f^2.l_{T,T}(Q)/v_{2T}(Q)$
4: $\quad T \leftarrow 2T$
5: $\quad$ **if** $r_i = 1$ **then**
6: $\quad\quad f \leftarrow f.l_{T,P}(Q)/v_{T+P}(Q)$
7: $\quad\quad T \leftarrow T + P$
8: $\quad$ **end if**
9: **end for**
10: $f \leftarrow f^{p^d-1}$
11: $f \leftarrow f^{(p^d+1)/r}$
12: **return** f

The fact that k is always even allows us to assume that the extension field $\mathbb{F}_{p^k}$ is built as a quadratic extension on top of an implementation of $\mathbb{F}_{p^d}$. An element in $\mathbb{F}_{p^k}$ can then be represented as $w = a + ib$ where $a, b \in \mathbb{F}_{p^d}$. Then the conjugate of w is $\bar{w} = a - ib$. Now it is well known (Frobenius) that

$$(a + ib)^{p^d} = (a - ib)$$

The next optimization we suggest [23], [26] is to further exploit the fact that the output of the Miller loop is to be raised to the power of $p^d - 1$. From the above it follows immediately that

$$(1/(a + ib))^{p^d-1} = (a - ib)^{p^d-1}$$

In other words after raising to the power of $p^d - 1$ inversion and conjugation cannot be distinguished. So now "push" the effect of the final exponentiation back into the main Miller loop, and replace inversions by much cheaper conjugations.

Algorithm 3. Computation of $e(P, Q)$ with further optimization

INPUT: $P \in E(\mathbb{F}_p), Q \in (\mathbb{F}_{p^k})$, where P has order r
OUTPUT: $e(P, Q)$

$T \leftarrow P$, $f \leftarrow 1$
for $i \leftarrow \lfloor \lg(r) \rfloor - 1$ **downto** 0 **do**
  $f \leftarrow f^2.l_{T,T}(Q).\bar{v}_{2T}(Q)$
  $T \leftarrow 2T$
  if $r_i = 1$ **then**
    $f \leftarrow f.l_{T,P}(Q).\bar{v}_{T+P}(Q)$
    $T \leftarrow T + P$
  end if
end for
$f \leftarrow f^{p^d-1}$
$f \leftarrow f^{(p^d+1)/r}$
return f

Now at a stroke we have gotten rid of the potentially very expensive extension field divisions in the main Miller loop (algorithm 3).

7 What About Q?

We have already chosen P to suit our needs, but what about Q? We would like to choose Q from a group disjoint from P, and ideally with some other advantages.

As things stand the point Q is a general point (x_Q, y_Q) on the elliptic curve over the field $\mathbb{F}_{p^k}$, where $x_Q = a + ib$ and $y_Q = c + id$, and $a, b, c, d \in \mathbb{F}_{p^d}$. Let us now restrict Q to be of a form where $b = c = 0$. This has an immediately useful effect as $\bar{v}_{2T}(Q)$ and $\bar{v}_{T+P}(Q)$ are now elements in the field $\mathbb{F}_{p^d}$, which

means that they will be wiped out by the final exponentiation. This is the famous denominator-elimination optimization [2]. And as an extra bonus it is not difficult to establish that if $Q(a, id)$ is a point on $E(\mathbb{F}_{p^k})$, then it can be mapped to a point in an isomorphic group on the quadratic twist of this curve $E'(\mathbb{F}_{p^d})$. So now Q, prior to its use in the pairing, can be manipulated as a point over the smaller extension field $\mathbb{F}_{p^d}$ [4] (requiring a small modification to the line function). See algorithm 4.

Algorithm 4. Computation of $e(P, Q)$ with denominator elimination

INPUT: $P \in E(\mathbb{F}_p), Q \in E'(\mathbb{F}_{p^d})$, where P has order r
OUTPUT: $e(P, Q)$
1: $T \leftarrow P$, $f \leftarrow 1$
2: **for** $i \leftarrow \lfloor \lg(r) \rfloor - 1$ **downto** 0 **do**
3: $\quad f \leftarrow f^2.l_{T,T}(Q)$
4: $\quad T \leftarrow 2T$
5: $\quad$ **if** $r_i = 1$ **then**
6: $\quad\quad f \leftarrow f.l_{T,P}(T, Q)$
7: $\quad\quad T \leftarrow T + P$
8: $\quad$ **end if**
9: **end for**
10: $f \leftarrow f^{p^d-1}$
11: $f \leftarrow f^{(p^d+1)/r}$
12: **return** f

In some situations and for certain types of pairings it may appear at first glance that such a favourable choice of Q may not be possible. However using a trick from [37] and [4], in fact it is always possible.

8 Some Further Optimizations

The group order r will always be odd, therefore the last iteration of the Miller loop will always invoke the $r_i = 1$ condition. However this last point addition, which takes T to the point at infinity, always results in a line value which will be again wiped out by the final exponentiation, and so this last step can be omitted [13].

Looking again at the final exponentiation we observe that in many cases the exponent $p^d + 1$ can be further factored. For example $p^3 + 1 = (p+1)(p^2 - p + 1)$. From the definition of k, the group order must divide $p^2 - p + 1$, which is the sixth cyclotomic polynomial $\Phi_6(p)$. So in general the final exponentiation can be broken down into three parts, the easy exponentiation to the power of $p^d - 1$, the equally easy exponentiation to the power of $(p^d+1)/\Phi_k(p)$ (easy because the Frobenius can be used), and the "hard" exponentiation to the power of $\Phi_k(p)/r$. These modifications lead us to algorithm 5.

Algorithm 5. Computation of $e(P,Q)$ with yet more optimization

INPUT: $P \in E(\mathbb{F}_p), Q \in E'(\mathbb{F}_{p^d})$, where P has order r
OUTPUT: $e(P,Q)$

$T \leftarrow P$, $f \leftarrow 1$
$s \leftarrow r-1$
for $i \leftarrow \lfloor \lg(s) \rfloor - 1$ **downto** 0 **do**
  $f \leftarrow f^2.l_{T,T}(Q)$
  $T \leftarrow 2T$
  if $s_i = 1$ **then**
    $f \leftarrow f.l_{T,P}(Q)$
    $T \leftarrow T+P$
  end if
end for
$f \leftarrow f^{p^d-1}$
$f \leftarrow f^{(p^d+1)/\Phi_k(p)}$
$f \leftarrow f^{\Phi_k(p)/r}$
return f

8.1 Calculating the Hard Part of the Final Exponentiation

When exponentiating in $\mathbb{F}_{p^k}$ there is never any need to use an exponent greater than p. Recall that exponentiation to the power of p is cheap, using the Frobenius. Therefore an exponent e can be represented to the base p as $e_0 + e_1.p + e_2.p^2 ....$ Now f^e can be written as

$$f^e = f^{e_0+e_1.p+e_2.p^2\cdots} = f^{e_0}.(f^p)^{e_1}.(f^{p^2})^{e_2}...$$

which can be quickly calculated using the Frobenius and the fast method of simultaneous exponentiation [22], [20], [33]. Standard windowing methods can be used, and the fact that inverses can be treated as conjugates can be further exploited to allow a NAF representation of the exponent.

For smaller values of $k \leq 8$, it may be faster to use Lucas or XTR style exponentiation [28], which uses the larger exponent, but only requires arithmetic over the smaller fields $\mathbb{F}_{p^{k/2}}$ or $\mathbb{F}_{p^{k/3}}$ respectively [43], [20]. As a nice side effect this has the result of "compressing" the value of the pairing to the smaller field size. Even if the multi-exponentiation method is used, it may still be useful to finally compress the pairing output.

9 Application-Dependent Optimizations

It may be that in a particular application the first parameter P is a fixed constant, like perhaps a fixed private key. In this case it clearly makes sense to precompute the values of T which are just fixed multiples of P. In this situation it is always preferable to use affine coordinates for all the points (although in the general case where $P \in E(\mathbb{F}_p)$ projective coordinates are to be preferred).

Some protocols require the pairing value, an element of order r, to be subsequently further exponentiated to a value $v < r$. If using the multi-exponentiation method for the hard part of the final exponentiation, then we observe that the further exponentiation to the power of v can be "folded into" the multi-exponentiation at virtually no extra cost. In many protocols these pairing exponentiations are included in the estimate of the protocols cost. Using this idea they can be obtained "for free". This trick works best for larger k.

If an entirely inversion-free pairing is desirable, then use projective coordinates throughout, and at little extra cost the full final exponentiation to the power of $(p^k - 1)/r$ can be carried out using one large multi-exponentiation.

10 Curve-Dependent Optimizations

10.1 The Ate Pairing

Some useful optimizations are possible, but they will depend on the particular type of pairing-friendly curve that is used. For example there are families of curves developed by Barreto et al. [3] and Duan et al. [12], for which the ω parameter is always greater than one. The same is true for the MNT curves where $\omega = 2$. In these cases a truncated loop variant of the Tate pairing is possible called the Ate pairing [24]. Here instead of putting the first parameter P on the curve over $\mathbb{F}_p$, and Q as a point on the twist over $\mathbb{F}_{p^d}$, we do it the other way around. Then, as it turns out, by simply substituting the line $s = t-1$ for the line $s = r - 1$ in algorithm 5, we still get a viable bilinear pairing (which has a simple relationship with the Tate pairing) with a Miller loop truncated by a factor of ω. For example for the BN curves mentioned above, the loop will be half the length of that required for the Tate pairing. Of course the fact that P is now taken over an extension field introduces an extra cost to the manipulation of T. This will not matter at all if P is fixed, as the precomputation optimization applies. In fact as we will see there are other ways to offset this extra cost.

10.2 Low Discriminant Complex Multiplication Curves

The parameters of all non-supersingular pairing-friendly elliptic curves must be found using the method of Complex Multiplication (CM method) [10]. Many families of pairing-friendly curves are found to have a CM discriminant of -1 or -3 [3], [12], [5]. In these cases the curves exhibit quartic and sextic twists respectively, as well as quadratic twists. Now it may be possible to select Q in the Tate pairing (and P in the Ate pairing) from a higher order twist, and hence over a lower order extension field. For example for the BN curves for which $k = 12$ it is possible to place P on $\mathbb{F}_p$ and Q on the sextic twist over $\mathbb{F}_{p^2}$, or vice versa for the Ate pairing. In fact for a $k = 6$ pairing friendly curve with a CM discriminant of -3, it is perfectly possible to have both P and Q as points on

Algorithm 6. Computation of $e(P,Q)$ with Ate pairing

INPUT: $P \in E'(\mathbb{F}_{p^d}), Q \in E(\mathbb{F}_p)$, P has order r
OUTPUT: $e(P,Q)$
1: $T \leftarrow P$, $f \leftarrow 1$
2: $s \leftarrow t-1$
3: **for** $i \leftarrow \lfloor \lg(s) \rfloor - 1$ **downto** 0 **do**
4: $\quad f \leftarrow f^2.l_{T,T}(Q)$
5: $\quad T \leftarrow 2T$
6: $\quad$ **if** $s_i = 1$ **then**
7: $\quad\quad f \leftarrow f.l_{T,P}(Q)$
8: $\quad\quad T \leftarrow T+P$
9: $\quad$ **end if**
10: **end for**
11: $f \leftarrow f^{p^d-1}$
12: $f \leftarrow f^{(p^d+1)/\Phi_k(p)}$
13: $f \leftarrow f^{\Phi_k(p)/r}$
14: **return** f

curves over $\mathbb{F}_p$. However we are only able to find such a pairing friendly curve with $\rho = 2$. For example consider this family of curves [15].

$$\begin{aligned} p(x) &= 27x^4 + 9x^3 + 3x^2 + 3x + 1 \\ r(x) &= 9x^2 + 3x + 1 \\ t(x) &= 3x + 2 \end{aligned}$$

Observe that $\omega = 2$, so the Ate pairing will be efficient in this case. And don't be put off by $\rho = 2$ – this is a useful curve!

In all these cases the line functions $l_{A,B}(Q)$ will be of a sparse form which will further speed up calculations. An alternative idea which exploits low discriminant curves, which is particularly effective for the case $k = 2$, is described in [40].

10.3 Truncated Final Exponentiation

In some cases the method of generation of the curves allows us to dramatically shorten the hard part of the final exponentiation. Consider for example a $k = 6$ MNT curve. In this case the exponent will be $(p^2 - p + 1)/r$. Assume that $r = p + 1 - t$, in other words the number of points on the base field curve is a prime. Then this exponent will be $(p^2 - p + 1)/(p + 1 - t) = p \pm \delta$, where $\delta \approx t$. Therefore the hard part of the final exponentiation will be $f^{p \pm \delta} = f^p.f^{\pm\delta}$, which can be calculated using a Frobenius and a half-length exponentiation (and avoiding multi-exponentiation). This same idea applies to a varying extent to other families of curves – for example for the BN curves the hard part of the final exponentiation requires only a three-quarters length exponent.

10.4 Avoiding Multi-exponentiation

Using multi-exponentiation for the final exponentiation is fast, but requires a lot of memory for precomputed values. As it happens, in many cases it can be avoided. Considering again the final exponentiation for the BN curves, it is not hard to work out that the "hard part" of the final exponentiation in algorithms 5 and 6 can be calculated using multi-exponentiation as

$$f \leftarrow f^{p^3}.(f^{p^2})^{6x^2+1}.(f^p)^{36x^3-18x^2+12x+1}.f^{36x^3-30x^2+18x-2}$$

But with a little work this can also be equivalently calculated as

$$\begin{aligned} a &\leftarrow f^{6x-5} \\ b &\leftarrow a^p \\ c &\leftarrow ab \\ f &\leftarrow f^{p^3}.[c.(f^p)^2.f^{p^2}]^{6x^2+1}.c.(f^p.f)^9.a.f^4 \end{aligned}$$

requiring only simple exponentiation and some multiplications and squarings. For x negative, the first step can be calculated instead as $a = 1/f^{5-6*x}$. Note that the overall length of exponent in bits is not any greater than for the multi-exponentiation. Note also that if x has a low Hamming weight, windowing methods of exponentiation will not be much better than a simple square-and-multiply algorithm, again saving on memory for precomputed values. This could be useful in a constrained environment. In some experiments this method was found to be about 20% faster than using multi-exponentiation.

In this case it was beneficial to work out the exponents as explicit polynomials based on the polynomial description of the pairing-friendly family of curves. However we will not always be so lucky. For example attempting to do the same for the $k = 16$ family of curves discovered by Kachisa [25] results in very complex exponents that cannot be exploited in this way. However if this optimization does not apply, the optimization of section 9 can still be used instead in many applications.

10.5 Super Pairing-Friendly Curves

Using the method of Barreto, Lynn and Scott [3], we can find this pairing-friendly curve:-

$$\begin{aligned} D &= -1 \\ p(x) &= (1 + 2x + x^2 + x^6 - 2x^7 + x^8)/4 \\ t(x) &= x + 1 \\ \#E(x) &= ((x-1)^2)(x^2+1)(x^4 - x^2 + 1)/4 \\ &= ((x-1)^2.\Phi_4(x).\Phi_{12}(x))/4 \end{aligned}$$

An actual curve from this family can be easily found using the CM method.

We define a curve whose number of points is divisible by more than one cyclotomic polynomial, as a "super pairing-friendly" curve. In fact it is perfectly possible (but often overlooked) that one elliptic curve might support multiple embedding degrees.

By choosing points of an order which divides $\Phi_4(x)$ we can do pairing-based cryptography with an embedding degree of 4, albeit with a rather high ρ value of 4. Using points of an order which divides $\Phi_{12}(x)$ we can do pairing-based cryptography *on the same curve* with an embedding degree of 12 and $\rho = 2$.

One might for example choose x of about 128 bits (taking care that both $\Phi_4(x)$ and $\Phi_{12}(x)$ evaluate as primes or near-primes), to achieve a security equivalent to 128-bit AES, with a prime modulus of 1024 bits, using the embedding degree of 4. The discrete logarithm difficulty would be equivalent to about $4.1024 = 4096$ bits, which is about right. Later one could switch to an embedding degree of 12 on the same curve, to obtain security roughly equivalent to 256-bit AES, and with a discrete logarithm difficulty of $12.1024 = 12288$ bits, which again is about right [29].

11 The Wider Context

A pairing is not calculated in isolation, it is calculated as part of a wider context, to implement some useful cryptographic protocol.

11.1 Which Embedding Degree?

One question that arises immediately is which embedding degree should be used for a particular level of security. The answer is, regrettably, that it depends. Our view on it is summarised in Table 1.

Table 1. Key size security in bits

Symmetric key size	Group size	Extension field size	Embedding degree
80	160	960 – 1280	2–8
128	256	3000 – 5000	12–18
256	512	12000 – 18000	24–36

For example at the 80-bit level of security, one could use a $k = 2$ Cocks-Pinch curve with a 512-bit prime p, for an extension field size of 1024-bits, and using a group size r of 160 bits. Alternatively one could use a $k = 6$ MNT curve again with a 160-bit prime p and group size of 160 bits, and an extension field size of 960 bits. On the face of it these choices represent similar levels of security. The protocol may require, as well as the pairing, point multiplications, and in this case the smaller size of p (the field over which the pairing-friendly elliptic curve will be defined) will be a big advantage. On the other hand these point multiplications may be of fixed points, in which case, using precomputation, the

cost of point multiplication may in fact be insignificant, as in the case of Boneh and Franklin IBE [7]. Note that a point multiplication of a pairing parameter can always be changed to an exponentiation of the pairing itself, using the property of bilinearity, so point multiplications may not be needed at all. For a $k = 2$ curve, precomputation based on a fixed point P will be a big advantage, much less so on a $k = 6$ curve. For a $k = 2$ curve the parameter Q will be on the twist also over $\mathbb{F}_p$, whereas for a $k = 6$ MNT curve the twist is over $\mathbb{F}_{p^3}$, which is more complicated to implement. If one was implementing a short-signature scheme [8], then a $k = 2$ curve cannot even be considered. Clearly a very careful analysis of the protocol is required before a choice can be made.

The issue of how best to scale security in pairings has been much debated, see [27], [20], [41]. One point perhaps not considered is that the code for higher extension fields becomes a lot "fussier", in that rather than spending most of its time for example in the inner loops of a modular multiplication, it spends more time hopping around in and out of functions, and accumulating costs from function overheads which can be quite significant. We have already observed how in a constrained cache environment such code can be punished by an increased rate of instruction cache misses [44].

11.2 Products of Pairings

In the case of a protocol which requires the products of pairings, three ideas can be used to speed up the calculation [38]. For example consider the calculation of $e(P, Q) \cdot e(R, S)$. These can be combined into a single algorithm.

- Since the implicit multiplications of P and R occur in lock-step with one another, it makes sense to use affine coordinates in conjunction with Montgomery's trick. This means that just one modular inversion will be required instead of two. Montgomery's trick is based on the simple observation that $1/x = y/xy$ and $1/y = x/xy$.
- Both pairings can share the same *Miller variable* f. This means only a single squaring of f in the combined pairing algorithm will be required.
- Both pairings can share the final exponentiation (as pointed out by Solinas [45]).

These ideas are further evaluated in [21].

12 Timings

Here we present some timings, which, at the 80-bit level of security compare four quite different pairing implementations with one another, and with a standard 1024-bit RSA decryption on the same platform, a standard 32-bit PC. The code is written in a combination of C++, C and assembly language.

First, we implement the Tate pairing using a $k = 2$ Cocks-Pinch curve, a group size of 160 bits and a prime modulus of 512 bits, with an extension field size

of $2.512 = 1024$ bits. Second, we implement the Tate pairing on a $k = 2$ curve which supports an efficient endomorphism [39], indicated by EE in the table below. Next, an Ate pairing implementation using a $k = 4$ pairing-friendly curve [15] with $\omega = 2$, a group size of 160-bits and a prime modulus of 256 bits, with an extension field size of $4.256 = 1024$ bits. The extension field is implemented as a 2-over-2 tower of extensions. Both the $k = 2$ and $k = 4$ pairings use a Lucas sequence for the hard part of the final exponentiation. Finally we implement a $k = 6$ Tate pairing, using an MNT curve, a group size and prime modulus size both of 160 bits, with an extension field size of $6.160 = 960$ bits. In this case the extension field is implemented as a 2-over-3 tower of extensions. For the final exponentiation the trick of section 10.3 is used. The slightly smaller extension field size in this case is motivated by the desire to make the modulus p a multiple of the computer word size of 32-bits.

The point multiplication timings are for the multiplication of a variable point on the elliptic curve over the base field by a random 160-bit value using projective coordinates. Note that the Ate pairing without precomputation would probably be more competitive if the $E(\mathbb{F}_{p^2})$ arithmetic that is required was implemented using projective rather than affine coordinates, as field inversions as required when using affine coordinates are resistant to aggressive optimization.

Table 2. Timings in milliseconds on 3GHz Pentium IV (using SSE2 instructions)

	$k = 2$ Tate	$k = 2$ Tate (EE)	$k = 4$ Ate	$k = 6$ Tate
Pairing w/o precomputation	6.7	5.1	9.1	6.2
Pairing with precomp.	3.0	3.0	3.1	4.5
Point Multiplication	2.9	1.9	1.1	0.6
1024-bit RSA decryption	1.92			

Note that these timings could be improved a little by converting the C++ programs to C.

Of interest is the fact that an MNT $k = 6$ implementation is competitive with the $k = 2$ Cocks-Pinch curve, when precomputation does not apply. This is largely thanks to the implementation of an efficient towering extension. However when precomputation does apply, the $k = 2$ curves are still just about the fastest. The idea of using a curve with an efficient endomorphism seems to be competitive at this level of security. Using a smaller base field with a larger k does of course dramatically improve the base field point multiplication timings. The extent to which this is relevant depends on the protocol being implemented. When an efficient endomorphism is available, this can be exploited to speed up point multiplication times significantly using the method of Gallant, Lambert and Vanstone [19].

13 Conclusions

One of the big questions to be asked about pairing-based cryptography is this: Does it take so long to calculate the pairing that the viability of pairing-based cryptography must be called into question? The answer is No.

References

1. Bailey, D., Paar, C.: Optimal extension field for fast arithmetic in public key algorithms. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 472–485. Springer, Heidelberg (1998)
2. 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)
3. 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. 263–273. Springer, Heidelberg (2002)
4. Paulo, S.L.M., Barreto, B., 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 (2003)
5. 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)
6. Blake, I.F., Seroussi, G., Smart, N.P. (eds.): Advances in Elliptic Curve Cryptography, vol. 2. Cambridge University Press, Cambridge (2005)
7. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)
8. 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 (2002)
9. Chung, J., Hasan, M.A.: Asymmetric squaring formulae (2006) http://www.cacr.math.uwaterloo.ca/
10. Crandall, R., Pomerance, C.: Prime Numbers: a Computational Perspective. Springer, Heidelberg (2001)
11. Dahab, R., Devegili, A.J., OhEigeartaigh, C., Scott, M.: Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report, 2006/471 (2006) http://eprint.iacr.org/2006/471
12. Duan, P., Cui, S., Wah Chan, C.: Special polynomial families for generating more suitable elliptic curves for pairing-based cryptosystems. Cryptology ePrint Archive, Report, 2005/342 (2006) http://eprint.iacr.org/2005/342
13. Duursma, I., Lee, H.-S.: Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
14. Eisentrager, K., Lauter, K., Montgomery, P.L.: Fast elliptic curve arithmetic and improved weil pairing evaluation. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 343–354. Springer, Heidelberg (2003)
15. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report, 2006/372 (2006) http://eprint.iacr.org/2006/372

16. 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)
17. Friedland, P.: Absolute value and square root of a complex number. Communications of the ACM 10, 665 (1967)
18. Galbraith, S., Harrison, K., Soldera, D.: Implementing the Tate pairing. In: Fieker, C., Kohel, D.R. (eds.) Algorithmic Number Theory. LNCS, vol. 2369, pp. 324–337. Springer, Heidelberg (2002)
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. Granger, R., Page, D., Smart, N.P.: High security pairing-based cryptography revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. LNCS, vol. 4076, pp. 480–494. Springer, Heidelberg (2006)
21. Granger, R., Smart, N.P.: On computing products of pairings. Cryptology ePrint Archive, Report, 2006/172 (2006) http://eprint.iacr.org/2006/172
22. Hei, L., Dong, J., Pei, D.: Implementation of cryptosystems based on Tate pairing. J. Comput. Sci & Technolgy 20(2), 264–269 (2005)
23. ÓhÉigeartaigh, C.: Speeding up pairing computation (2005) http://eprint.iacr.org/2005/293
24. Hess, F., Smart, N., Vercauteren, F.: The Eta pairing revisited. Cryptology ePrint Archive, Report, 2006/110, (2006) http://eprint.iacr.org/2006/110
25. Kachisa, E.: Constructing brezing-weng pairing friendly elliptic curves using elements in the cyclotomic field. M.Sc. dissertation, Mzuzu University (2007)
26. Kobayashi, T., Aoki, K., Imai, H.: Efficient algorithms for Tate pairing. IEICE Trans. Fundamentals, E89-A (2006)
27. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)
28. Lenstra, A., Verheul, E.: An overview of the XTR public key system. In: Warsaw Conference on Public-Key cryptography and computational number theory (2001)
29. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. Journal of Cryptology 14(4), 255–293 (2001)
30. 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)
31. Lynn, B.: PBC library (2007) http://crypto.stanford.edu/pbc/download.html
32. Menezes, A.: Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publishers, Dordrecht (1993)
33. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton, Florida (1996) URL: http://cacr.math.uwaterloo.ca/hac
34. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Transactions on Fundamentals E84-A(5), 1234–1243 (2001)
35. Montgomery, P.: Modular multiplication without division. Mathematics of Computation 44(170), 519–521 (1985)
36. Schirokauer, O.: The number field sieve for integers of low weight. Cryptography ePrint Archive, Report, 2006/107 (2006) http://eprint.iacr.org/2006/107
37. Scott, M.: Faster identity based encryption. Electronics Letters 40(14), 861 (2004)

38. Scott, M.: Computing the Tate pairing. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
39. Scott, M.: Faster pairings using an elliptic curves with an efficient endomorphism. In: Maitra, S., Madhavan, C.E.V., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 258–269. Springer, Heidelberg (2005)
40. Scott, M.: A note on Boneh and Franklin IBE (2005) ftp://ftp.computing.dcu.ie/pub/resources/crypto/note.pdf
41. Scott, M.: Scaling security in pairing-based protocols. Cryptology ePrint Archive, Report, 2005/139 (2005) http://eprint.iacr.org/139
42. Scott, M.: (2007) http://ftp.computing.dcu.ie/pub/crypto/miracl.zip
43. Scott, M., Barreto, P.: Compressed pairings. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 140–156. Springer, Heidelberg (2004), Also available from http://eprint.iacr.org/2004/032/
44. Scott, M., Costigan, N., Abdulwahab, W.: Implementing cryptographic pairings on smartcards. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 134–147. Springer, Heidelberg (2006)
45. Solinas, J.: ID-based digital signature algorithms (2003) http://www.cacr.math.uwaterloo.ca/conferences/2003/ecc2003/solinas.pdf
46. Wang, F., Nogami, Y., Morikawa, Y.: An efficient square root computation in finite fields GF(p^{2^d}). IEICE Trans. Fundamentals E88-A(10), 2792–2799 (2005)

Implementing Cryptographic Pairings over Barreto-Naehrig Curves

Augusto Jun Devegili[1,⋆], Michael Scott[2], and Ricardo Dahab[1]

[1] Instituto de Computação, Universidade Estadual de Campinas
Caixa Postal 6176, CEP 13084-971 Campinas, SP, Brazil
augusto@devegili.org, rdahab@ic.unicamp.br
[2] School of Computing, Dublin City University
Dublin 9, Ireland
mscott@computing.dcu.ie

Abstract. In this paper we describe an efficient implementation of the Tate and Ate pairings using Barreto-Naehrig pairing-friendly curves, on both a standard PC and on a 32-bit smartcard. First we introduce a subfamily of such curves with a particularly simple representation. Next we consider the issues that arise in the efficient implemention of field arithmetic in $\mathbb{F}_{p^{12}}$, which is crucial to good performance. Various optimisations are suggested, including a novel approach to the 'final exponentiation', which is faster and requires less memory than the methods previously recommended.

1 Introduction

Pairing-based cryptography requires pairing-friendly curves. These are parameterised by their embedding degree k. The embedding degree dictates to an extent the security level efficiently achievable on the curve.

While it is well known that super-singular curves are viable and useful candidates, they are limited in terms of the possible values of the embedding degree. Furthermore the highest embedding degree possible for super-singular elliptic curves ($k = 6$) requires us to use curves of characteristic 3, which is rather awkward from an implementation point of view (we refer the reader to a recent paper on arithmetic in GF(3^m) by Ahmadi, Hankerson and Menezes [1]). Attention has therefore switched to consideration of non-supersingular curves of prime characteristic, for which there is no such limitation. Nonetheless finding suitable curves, or ideally whole families of suitable curves, has proven to be non-trivial [2].

In this context the security of pairing-based cryptography depends on finding curves whose order n is divisible by a large prime r such that generic attacks on small group orders (Pohlig-Hellman attacks) can be resisted. It is also important that $k \lg(p)$, where p is the modulus, is large enough to resist index-calculus attacks.

⋆ Funded by the Brazilian Government/Coordination for the Improvement of Higher Education Personnel (CAPES).

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 197–207, 2007.

Given this scenario, Barreto-Naehrig (BN) curves with their embedding degree of 12 are particularly significant, as before their discovery only MNT curves [3] of embedding degree 3, 4 and 6 were known to support rare curves of prime order ($r = n$). The prime order requirement is significant in terms of efficiency of implementation, and is crucial for certain applications [4]. Furthermore since BN curves define a whole family of curves, there are plenty to choose from. One drawback of using BN curves is Schirokauer's method [5], which can be applied to BN primes because of their special format. In this case, it might be necessary to rescale parameters accordingly.

2 Bilinear Pairings

A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ where $\mathbb{G}_1, \mathbb{G}_2$ are additive groups, $\mathbb{G}_3$ is a multiplicative group, and the map is linear in each component:

$$e(P + Q, R) = e(P, R) \cdot e(Q, R)$$
$$e(P, Q + R) = e(P, Q) \cdot e(P, R)\,.$$

Let $\mathbb{F}_p$ be the prime field with characteristic p and let $E(\mathbb{F}_p)$ be an elliptic curve defined over $\mathbb{F}_p$. Let n be the order of $E(\mathbb{F}_p)$, let r be a large prime dividing n, and let k be the least positive integer such that $r \mid p^k - 1$ and $r^2 \nmid p^k - 1$. We call such an integer the embedding degree of r with regard to $\mathbb{F}_p$, and we have that the r-torsion group of the curve is contained in $E(\mathbb{F}_{p^k})$ and the r-th roots of unity are contained in $\mathbb{F}_{p^k}$.

Let $[a]P$ denote the multiplication of a point $P \in E$ by a scalar $a \in \mathbb{Z}$, and let $\infty \in E$ denote the point at infinity. A Miller [6] function $f_{r,P}(\cdot)$ is a rational function on E with r zeroes at P, one pole at $[r]P$ and $r - 1$ poles at ∞:

$$(f_{r,P}) = r(P) - ([r]P) - (r - 1)\infty\,.$$

The Tate pairing [7] is a well-defined, non-degenerate bilinear pairing with $\mathbb{G}_1 = E[r]$, $\mathbb{G}_2 = E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k})$, and $\mathbb{G}_3 = \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r$. Let $P \in E[r]$ and $Q \in E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k})$. Then the Tate pairing of P, Q is computed as

$$e(P, Q) = f_{r,P}(Q)^{(p^k-1)/r}\,.$$

The Ate pairing [8] is a well-defined, non-degenerate bilinear pairing with $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_q - [1])$, $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi_q - [q])$, and $\mathbb{G}_3 = \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r$, where π_q is the Frobenius endomorphism. Let $P \in E[r] \cap \mathrm{Ker}(\pi_q - [1])$, $Q \in E[r] \cap \mathrm{Ker}(\pi_q - [q])$, and t be the trace of Frobenius of the curve. Then the Ate pairing of Q, P is computed as

$$e(Q, P) = f_{t-1,Q}(P)^{(p^k-1)/r}\,.$$

The most efficient pairing computation algorithms are derived from Miller's algorithm [6], which computes a Miller function $f_{r,P}$ evaluated at a point Q.

Algorithm 1 describes the BKLS algorithm [7] commonly used to compute the Tate pairing, including the denominator elimination optimisation that may be applied for even embedding degree. The algorithm needs $\mathbb{F}_{p^k}$ arithmetic (squaring and multiplication), elliptic curve arithmetic (point addition and doubling), computation of the line function $l_{A,B}(C)$ that intercepts points A, B and its evaluation at a point C, and a final exponentiation in $\mathbb{F}_{p^k}$.

Algorithm 1. BKLS algorithm to compute the Tate pairing $e(P, Q)$

Input: $P, Q \in E$ and $n, r \in \mathbb{Z}$
Output: $f_{r,P}(Q)^{(p^k-1)/n}$
1: $T \leftarrow P$
2: $f \leftarrow 1$
3: **for** $i \leftarrow \lfloor \lg(r) \rfloor - 2$ **downto** 0 **do**
4: $\quad f \leftarrow f^2 \cdot l_{T,T}(Q)$
5: $\quad T \leftarrow [2]T$
6: $\quad$ **if** $r_i = 1$ **then**
7: $\quad\quad f \leftarrow f \cdot l_{T,P}(Q)$
8: $\quad\quad T \leftarrow T + P$
9: $\quad$ **end if**
10: **end for**
11: $f \leftarrow f^{(p^k-1)/n}$

3 Barreto-Naehrig Curves

Barreto and Naehrig [9] devised a method to generate pairing-friendly elliptic curves over a prime field, with prime order and embedding degree $k = 12$. The equation of the curve is $E : y^2 = x^3 + b$, with $b \neq 0$. The trace (of Frobenius) of the curve, the curve order and the characteristic of $\mathbb{F}_p$ are parameterised as:

$$\begin{aligned} t(x) &= 6x^2 + 1 \\ n(x) &= 36x^4 - 36x^3 + 18x^2 - 6x + 1 \\ p(x) &= 36x^4 - 36x^3 + 24x^2 - 6x + 1\,. \end{aligned}$$

Such a curve is called a Barreto-Naehrig or BN curve. Throughout this text, we drop the parameter x and write t, n, p instead of $t(x)$, $n(x)$, $p(x)$ respectively. Because a BN curve has prime order, every point in the curve has order n, so the r value defined in Section 2 (a large prime dividing the curve order) is the same as n.

Since t, n, p are parameterised, the space needed to store or transmit information about a BN curve is small: The length of x is roughly one sixth of the concatenation of both p and t. This parameterisation also allows for a faster final exponentiation method described in Section 7.

Because BN curves have embedding degree $k = 12$, pairings are computed over points in $E(\mathbb{F}_{p^{12}})$. A common optimisation [7] is to take one of the points in $E(\mathbb{F}_p) \subset E(\mathbb{F}_{p^{12}})$ and the other in $E(\mathbb{F}_{p^{12}})$. However, it is possible to do better than this: By using an appropriate map, we can compress certain points in $E(\mathbb{F}_{p^{12}})$ to points in a sextic twist $E'(\mathbb{F}_{p^2})$. Let $\xi \in \mathbb{F}_{p^2}$ be such that $W^6 - \xi$ is irreducible over $\mathbb{F}_{p^2}[W]$ whenever $p \equiv 1 \pmod 6$. Then there exists a curve $E'/\mathbb{F}_{p^2} : y'^2 = x'^3 + b/\xi$ that is a sextic twist of $E/\mathbb{F}_p$ and whose order is divisible by n. In fact, the order of the sextic twist $E'/\mathbb{F}_{p^2}$ is $n(2p - n)$.

Let $z \in \mathbb{F}_{p^{12}}$ be a root of $W^6 - \xi$. We can build an injective group homomorphism $\psi : E'(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^{12}})$ as $(x', y') \mapsto (x'z^2, y'z^3)$. This allows us to map points in the sextic twist $E'(\mathbb{F}_{p^2})$ to points in $E(\mathbb{F}_{p^{12}})$, and if ξ is used to construct the extension field $\mathbb{F}_{p^{12}}$ then multiplication by powers of z does not incur any multiplication overhead.

We chose x such that p is a 256-bit prime subject to the following congruences: $p \equiv 7 \pmod 8$ (so that we can use -2 as a quadratic non-residue, optimising arithmetic operations on $\mathbb{F}_{p^2}$), $p \equiv 4 \pmod 9$ (as suggested in [9] to compute cube roots efficiently), and $p \equiv 1 \pmod 6$ (so that we can find $\xi \in \mathbb{F}_{p^2}$ such that $W^6 - \xi$ is irreducible over $\mathbb{F}_{p^2}[W]$). The elliptic curve equation is given by $E/\mathbb{F}_p : y^2 = x^3 + 3$, and $G = (1, 2)$ is a generator of $E(\mathbb{F}_p)$. For a 256-bit modulus, the x which defines the curve is just a 64-bit number.

Observe from Algorithm 1 that the Ate pairing benefits from r having low Hamming weight. In the context of BN curves, this is facilitated by preferring an x value of low Hamming weight. For example, we find that r has a Hamming weight of 90 and t has a Hamming weight of 28 if we choose x = -6000000000001F2D (hex).

4 Finite Field Arithmetic

We construct the finite extension field $\mathbb{F}_{p^{12}}$ as a tower of finite extensions: Quadratic on top of a cubic on top of a quadratic. The quadratic/cubic non-residues and reduction polynomials are detailed in Table 1. The multiplication and squaring algorithms chosen to implement field arithmetic are listed in Table 2. We made the choice of multiplication and squaring algorithms based on the exhaustive testing described in [10].

Table 1. Extension fields

Extension	Non-Residue	Construction	Representation
$\mathbb{F}_{p^2}$	$\beta = -2$	$\mathbb{F}_p[X]/(X^2 - \beta)$	$a = a_0 + a_1 X$
$\mathbb{F}_{p^6}$	$\xi = -1 - \sqrt{\beta}$	$\mathbb{F}_{p^2}[Y]/(Y^3 - \xi)$	$a = a_0 + a_1 Y + a_2 Y^2$
$\mathbb{F}_{p^{12}}$	$\xi' = \sqrt[3]{\xi}$	$\mathbb{F}_{p^6}[Z]/(Z^2 - \xi')$	$a = a_0 + a_1 Z$

Table 2. Multiplication and squaring algorithms for finite extension fields

Extension	Multiplication	Squaring
$\mathbb{F}_p$	Comba	Comba
$\mathbb{F}_{p^2}$	Karatsuba	Complex
$\mathbb{F}_{p^6}$	Karatsuba	Chung-Hasan SQR2
$\mathbb{F}_{p^{12}}$	Karatsuba	Complex

Throughout this text we also use an alternative equivalent representation of elements in $\mathbb{F}_{p^{12}}$ based on z, a root of $(W^6 - \xi) \in \mathbb{F}_{p^2}[W]$:

$$\begin{aligned} a \in \mathbb{F}_{p^{12}} &= (a_{0,0} + a_{0,1}Y + a_{0,2}Y^2) + (a_{1,0} + a_{1,1}Y + a_{1,2}Y^2)Z \\ &= a_{0,0} + a_{1,0}z + a_{0,1}z^2 + a_{1,1}z^3 + a_{0,2}z^4 + a_{1,2}z^5\,, \end{aligned}$$

where $a_{i,j} \in \mathbb{F}_{p^2}$.

5 The Tate Pairing

The Tate pairing $e(P, Q)$ takes a point $P = (x_P, y_P) \in E(\mathbb{F}_p)$ and a point $Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^{12}})$. Recall from Section 3 that BN curves have sextic twists defined over $\mathbb{F}_{p^2}$, so we can use a point $Q' = (x'_Q, y'_Q) \in E'(\mathbb{F}_{p^2})$ instead of the full point $Q \in E(\mathbb{F}_{p^{12}})$, and that there exists a group homomorphism $\psi : E'(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^{12}})$ defined as $\psi((x', y')) \mapsto (x'z^2, y'z^3)$. The coordinates of a point Q in the codomain of ψ have a compact representation, needing at most $1/6$ of the number of $\mathbb{F}_p$ elements needed to represent a full $\mathbb{F}_{p^{12}}$ element.

In Algorithm 1, the point P is repeatedly doubled or added during the Miller loop, and therefore we need elliptic curve arithmetic on $E(\mathbb{F}_p)$.

We use $\lambda_{A,B}$ in order to evaluate $l_{A,B}(Q)$. Because $A, B \in E(\mathbb{F}_p)$, every coordinate is an element of $\mathbb{F}_p$ and thus $\lambda_{A,B} \in \mathbb{F}_p$. Let $C = A + B$. Using projective coordinates,

$$\begin{aligned} l_{A,B}(Q) &= (y_Q \cdot z_A^3 - y_A)z_C - (x_Q \cdot z_A^3 - x_A \cdot z_A)\lambda_{A,B} \\ &= y'_Q \cdot z^3 \cdot z_A^3 \cdot z_C - y_A \cdot z_C - x'_Q \cdot z^2 \cdot z_A^3 \cdot \lambda_{A,B} - x_A \cdot z_A \cdot \lambda_{A,B} \\ &= (x_A \cdot z_A \cdot \lambda_{A,B} - y_A \cdot z_C) - (x'_Q \cdot z_A^3 \cdot \lambda_{A,B})z^2 + (y'_Q \cdot z_A^3 \cdot z_C)z^3\,. \end{aligned}$$

Since $x_A, y_A, \lambda_{A,B} \in \mathbb{F}_p$ and $x'_Q, y'_Q \in \mathbb{F}_{p^2}$, we can avoid $\mathbb{F}_{p^{12}}$ arithmetic entirely when computing $l_{A,B}(Q)$.

6 The Ate Pairing

The Ate pairing $e(Q, P)$ takes a point $Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^{12}})$ and a point $P = (x_P, y_P) \in E(\mathbb{F}_p)$. As in Section 5, we can use a point $Q' = (x'_Q, y'_Q) \in E'(\mathbb{F}_{p^2})$ instead of the full point $Q \in E(\mathbb{F}_{p^{12}})$.

The Miller loop needs two elliptic curve arithmetic operations: Point doubling and point addition. As these operations are computed over Q', we need elliptic curve arithmetic over $E'(\mathbb{F}_{p^2})$. Instead of using r to control the Miller loop, the Ate pairing uses $s = t - 1$, which in the case of BN curves provides for roughly half the number of iterations needed to compute the Tate pairing. Algorithm 2 describes the BKLS algorithm adapted to compute the Ate pairing.

Algorithm 2. BKLS-like algorithm to compute the Ate pairing $e(Q, P)$

Input: $P, Q \in E$ and $n, s = t - 1 \in \mathbb{Z}$
Output: $f_{t-1,P}(Q)^{(p^k-1)/n}$
1: $T \leftarrow P$
2: $f \leftarrow 1$
3: **for** $i \leftarrow \lfloor \lg(s) \rfloor - 2$ **downto** 0 **do**
4: $\quad f \leftarrow f^2 \cdot l_{T,T}(Q)$
5: $\quad T \leftarrow [2]T$
6: $\quad$ **if** $s_i = 1$ **then**
7: $\quad\quad f \leftarrow f \cdot l_{T,P}(Q)$
8: $\quad\quad T \leftarrow T + P$
9: $\quad$ **end if**
10: **end for**
11: $f \leftarrow f^{(p^k-1)/n}$

The formula for the slope of the line function $l_{A,B}$ that intercepts two points $A = (x_A, y_A)$ and $B = (x_B, y_B)$ is

$$\lambda_{A,B} = \frac{y_B - y_A}{x_B - x_A}.$$

If $A, B \in E(\mathbb{F}_{p^{12}})$, the slope is an element of $\mathbb{F}_{p^{12}}$. We can easily derive this from the slope of the line function $l_{A',B'}$ that intercepts $A', B' \in E'(\mathbb{F}_{p^2})$, where A', B' are the images of A, B under ψ:

$$\lambda_{A,B} = \frac{(y'_B - y'_A)z^3}{(x'_B - x'_A)z^2} = (\lambda_{A',B'})z\,,$$

so computing $\lambda_{A,B} \in \mathbb{F}_{p^{12}}$ amounts to computing $\lambda_{A',B'} \in \mathbb{F}_{p^2}$. We use this slope to evaluate $l_{A,B}(P)$ as follows:

$$\begin{aligned} l_{A,B}(P) &= (x_P - x_A)\lambda_{A,B} - (y_P - y_A) \\ &= (x_P - x'_A \cdot z^2)(\lambda_{A',B'})z - (y_P - y'_A \cdot z^3) \\ &= (-y_P) + (x_P \cdot \lambda_{A',B'})z + (y'_A - x'_A \cdot \lambda_{A',B'})z^3\,. \end{aligned}$$

We prefer affine coordinates because the savings due to the use of projective coordinates are diluted when compared to the overall $\mathbb{F}_{p^{12}}$ operations.

As $x_P, y_P \in \mathbb{F}_p$ and $x'_A, y'_A, \lambda_{A',B'} \in \mathbb{F}_{p^2}$, we can avoid $\mathbb{F}_{p^{12}}$ arithmetic entirely when computing $l_{A,B}(P)$. The resulting value has a sparse representation in $\mathbb{F}_{p^{12}}$,

needing only five $\mathbb{F}_p$ elements instead of twelve. However, the Miller variable is a full $\mathbb{F}_{p^{12}}$ element. For a given point P, the value $-y_P$ is constant, so it needs to be computed only once during the pairing computation.

7 Final Exponentiation

Both the Ate and Tate pairing algorithms compute a final exponentiation after the Miller loop. A common optimisation of this computation is to factor $(p^k - 1)/n$ into three parts: An easy exponentiation to the power of $(p^{k/2} - 1)$, an equally easy exponentiation to the power of $(p^{k/2} + 1)/\Phi_k(p)$ (easy because of the Frobenius), and a 'hard' exponentiation to the power of $\Phi_k(p)/n$, where Φ_k is the k-th cyclotomic polynomial [11]. In the context of BN curves, these exponents translate to $(p^6 - 1)$, $(p^2 + 1)$, and $(p^4 - p^2 + 1)/n$.

Recall that p and n have a special form: Both are polynomials on x. Therefore this hard part of the final exponentiation can be computed explicitly as a large polynomial in x. This can in turn be expressed to the base p as

$$p^3 + (6x^2 + 1)p^2 + (36x^3 - 18x^2 + 12x + 1)p + (36x^3 - 30x^2 + 18x - 1)\,.$$

Now the standard continuation is to use the method of multi-exponentiation combined with the Frobenius [12], so that the final exponentiation is the calculation of

$$(f^{p^3}) \cdot (f^{p^2})^{6x^2+1} \cdot (f^p)^{36x^3-18x^2+12x+1} \cdot f^{36x^3-30x^2+18x-1}\,.$$

However, multi-exponentiation is expensive in terms of memory as it requires extensive precomputation. So instead we exploit the specific and fixed form of the final exponent to obtain Algorithm 3 which can easily be verified to produce the equivalent result, but only requires simple exponentiation.

Algorithm 3. 'Hard' exponentiation

Input: f, x, p
Output: $f^{(p^4-p^2+1)/n}$
1: $a \leftarrow f^{6x-5}$
2: $b \leftarrow a^p$ using Frobenius
3: $b \leftarrow ab$
4: Compute f^p, f^{p^2}, and f^{p^3} using Frobenius
5: $f \leftarrow f^{p^3} \cdot \left[b \cdot (f^p)^2 \cdot f^{p^2}\right]^{6x^2+1} \cdot b \cdot (f^p \cdot f)^9 \cdot a \cdot f^4$

Note that exponentiations to powers of p are efficiently computed using Frobenius; other exponentiations may be computed using the square-and-multiply method. Because x is chosen so as to have low Hamming weight, there is no benefit in using window methods to compute the exponentiations to the powers of $6x - 5$ and $6x^2 + 1$, which saves on memory. Experiments show this method to be 20% faster than using multi-exponentiation.

8 The Philips HiPerSmart™ Smartcard

As in [13], we used the Philips HiPerSmart™ smartcard, which is an instantiation of the MIPS32®-based SmartMIPS® architecture with various instruction set enhancements to facilitate the implementation of popular cryptographic algorithms. Fortunately these enhancements are also relevant to our efforts here.

Specifically the architecture supports a `ACX|HI|LO` triple of registers that can be used to accumulate the partial products that arise when employing the popular Comba/Montgomery technique for multi-precision multiplication [14]. This is supported by a modified `MADDU` instruction which carries out an unsigned integer multiplication and addition to the triple register.

One architectural feature of particular significance is the 2 KB instruction cache. This is appropriate for an algorithm which spends most of its time in a small inner loop, as say for a modular exponentiation as required by the RSA algorithm. However it is problematical for the more complex algorithms considered here. Therefore we found that it was hard to avoid frequent cache misses, which are particularly expensive at the higher clock speeds. In particular it was self-defeating to try and use the popular optimisation of loop-unrolling, as the reduction in instruction count was more than offset by an increase in clock cycle count.

By using stack allocation for the multiprecision variables it was possible to keep the RAM requirement within the 16 KB allocation. However this could have been a problem if we were to use methods which are more memory-hungry (like multi-exponentiation).

9 Results

The performance of the Ate and Tate pairings was measured on two platforms: The Philips HiPerSmart™ platform described in Section 8, and an Intel Pentium IV 3.4 GHz running GNU/Linux 2.6.18.

Our programs used the MIRACL library[1], which implements multi-precision number arithmetic, and supports a number of powerful optional optimizations. In particular it supports completely unrolled assembly language support for fixed-size big number multiplication and modular reduction. Internally, prime field elements are in Montgomery representation [15], which allows for fast reduction without divisions. The memory for big numbers can be allocated from the heap, or more efficiently from the stack, which is the case for our experiments. When required to multiply big numbers by a small integer, multiplications by numbers less than or equal to 6 are instead carried out by up to 3 modular additions.

Table 3 lists the number of instructions required to compute the pairings on the smartcard, as well as the number of cache misses. The actual count of clock cycles and the time needed to compute the pairings is listed in Tables 4 and 5. The timings for the 3.4 GHz Pentium IV are listed in Table 6.

[1] `http://www.shamus.ie`

Table 3. Instructions required (% icache misses) — Philips HiPerSmart™

	Ate pairing	Tate pairing
Miller loop	47,987,089 (17.4%)	65,163,248 (19.3%)
Final exponentiation	58,948,105 (19.4%)	58,925,260 (19.3%)
Total	106,935,194 (18.5%)	124,088,508 (19.3%)

Table 4. Clock cycles required/CPI/time in seconds @ 9 MHz

	Ate pairing	Tate pairing
Miller loop	67,538,579/1.41/7.50	93,524,664/1.44/10.39
Final exponentiation	84,373,180/1.43/9.37	84,387,321/1.43/9.38
Total	151,911,759/1.42/16.88	177,911,985/1.43/19.77

Table 5. Clock cycles required/CPI/time in seconds @ 20.57 MHz

	Ate pairing	Tate pairing
Miller loop	77,125,478/1.61/3.75	107,704,196/1.65/5.24
Final exponentiation	97,161,864/1.65/4.72	97,144,880/1.65/4.72
Total	174,287,342/1.63/8.47	204,849,076/1.65/9.96

Table 6. Timings in miliseconds on 3.4 GHz Intel Pentium VI

	Ate pairing	Tate pairing
Miller loop	17.4	24.3
Final exponentiation	21.9	21.8
Total	39.3	46.1

Our hardware emulator is only cycle-accurate up to 20.57 MHz, but the maximum supported speed for the smartcard is 36 MHz. At this clock frequency the Ate pairing over BN curves should take approximately 5 seconds, which is clearly not adequate for use at the moment. Nevertheless it is anticipated that, with improvements in technology, BN curves might be practical even on such resource constrained devices in the near future.

10 Conclusions

We have described the first implementation of both the Ate and Tate pairings over Barreto-Naehrig curves.

Because of the restrictions imposed on the selection of primes for BN curves, we have not used primes congruent to 1 (mod 12), and therefore our finite fields are not strictly pairing-friendly in the sense of Koblitz and Menezes [16]. We have used the technique of towering extensions to construct the extension field

$\mathbb{F}_{p^{12}}$, and our choice of non-residues for the reduction polynomials that define the extensions still allows for efficient finite field arithmetic.

In order to avoid full $\mathbb{F}_{p^{12}}$ arithmetic throughout pairing computation, we have provided explicit $\mathbb{F}_{p^2}$-formulae for the evaluation of the line function $l_{A,B}(C)$ required by the Miller algorithm, thus alleviating part of the burden imposed by using the $k = 12$ embedding degree. The value of $l_{A,B}(C)$ computed by the Ate (resp. Tate) pairing uses five (resp. six) $\mathbb{F}_p$ components instead of twelve.

Our method for the final exponentiation is both faster and less memory-intensive than previous methods in the literature which are based on multi-exponentiation. We also observe that the analysis of Hess, Smart and Vercauteren [8] for the settings of BN curves ($k = 12$, $\lg(p) = \lg(r) = 256$) is more optimistic with regard to the Ate pairing than our experimental results.

Acknowledgements. We would like to thank the anonymous reviewers for their valuable comments, and Décio Gazzoni for insightful information on the number field sieve.

References

1. Ahmadi, O., Hankerson, D., Menezes, A.: Software implementation of arithmetic in GF(3^m). In: WAIFI 2007 (to be published)
2. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report, 2006/372 (2006) http://eprint.iacr.org/
3. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Trans. Fundamentals E84-A(5), 1234–1243 (2001)
4. Boneh, D., Lynn, B., Schacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17(4), 297–319 (2004)
5. Schirokauer, O.: The number field sieve for integers of low weight. Cryptology ePrint Archive, Report, 2006/107 (2006) http://eprint.iacr.org/
6. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17(4), 235–261 (2004)
7. 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)
8. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
9. 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)
10. Devegili, A.J., Ó hÉigeartaigh, C., Scott, M., Dahab, R.: Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report, 2006/471 (2006) http://eprint.iacr.org/
11. Granger, R., Page, D., Smart, N.P.: High security pairing-based cryptography revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. LNCS, vol. 4076, pp. 480–494. Springer, Heidelberg (2006)
12. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)

13. Scott, M., Costigan, N., Abdulwahab, W.: Implementing cryptographic pairings on smartcards. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 134–147. Springer, Heidelberg (2006)
14. Großschädl, J., Savas, E.: Instruction set extensions for fast arithmetic in finite fields GF(p) and GF(2^m). In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, Springer, Heidelberg (2004)
15. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44(170), 519–521 (1985)
16. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)

Instruction Set Extensions for Pairing-Based Cryptography*

Tobias Vejda[1], Dan Page[2], and Johann Großschädl[2]

[1] Institute for Applied Information Processing and Communications, Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria
Tobias.Vejda@iaik.tugraz.at
[2] University of Bristol, Department of Computer Science, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, U.K.
{page,johann}@cs.bris.ac.uk

Abstract. A series of recent algorithmic advances has delivered highly effective methods for pairing evaluation and parameter generation. However, the resulting multitude of options means many different variations of base field must ideally be supported on the target platform. Typical hardware accelerators in the form of co-processors possess neither the flexibility nor the scalability to support fields of different characteristic and order. On the other hand, extending the instruction set of a general-purpose processor by custom instructions for field arithmetic allows to combine the performance of hardware with the flexibility of software. To this end, we investigate the integration of a tri-field multiply-accumulate (MAC) unit into a SPARC V8 processor core to support arithmetic in $\mathbb{F}_p$, $\mathbb{F}_{2^n}$ and $\mathbb{F}_{3^n}$. Besides integer multiplication, the MAC unit can also execute dedicated multiply and MAC instructions for binary and ternary polynomials. Our results show that the tri-field MAC unit adds only a small size overhead while significantly accelerating arithmetic in $\mathbb{F}_{2^n}$ and $\mathbb{F}_{3^n}$, which sheds new light on the relative performance of $\mathbb{F}_p$, $\mathbb{F}_{2^n}$ and $\mathbb{F}_{3^n}$ in the context of pairing-based cryptography.

1 Introduction

Although pairings, or bilinear maps, on elliptic curves were initially only useful as a destructive tool for cryptanalysis, a slew of constructive applications [12] has motivated research into efficient pairing evaluation. Clearly the dominant form of optimisation for pairing evaluation lies at the algorithmic level; for a good overview of the evolution of optimisations, see the description of Scott [37]. In short, improvement of seminal but unpublished work by Miller [31] resulted in the first practical algorithms for evaluation of the Tate pairing [7,16]. These

* The work described in this paper has been supported by the European Commission through the IST Programme under contract no. IST-2002-507932 ECRYPT. The information in this paper reflects only the authors' views, is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 208–224, 2007.

results were further optimised by Duursma and Lee [13] who developed an inexpensive, closed form for specific parameterisations, later improved by Kwon [30]. Their techniques were generalised and extended to produce the Eta [6] and Ate [26] pairings, currently considered the fastest means of evaluating cryptographic pairings.

Much like the situation with vanilla elliptic curve cryptography (ECC), there is a broad range of choices to consider when actually implementing these algorithms; this range is amplified by the larger number of parameterisation options [37]. There are three major regions within the hardware/software design space for pairing evaluation, each offering a different cost versus performance trade-off. At one end of the design space, the composition of arithmetic in $\mathbb{F}_{q^k}$ has motivated numerous designs for hardware accelerators [42,10,27,28,8], which follow high-performance ECC accelerator design by utilising dedicated, parallel execution units. Coupled with the potential for pipelining, this approach trades area in favour of low latency. One can make the opposite trade-off by following ECC co-processor design and utilising a single execution unit in a more iterative manner [35,17]. With the co-processor coupled to a general-purpose processor core, this approach moves toward a more flexible solution in that it can, for example, be used to accelerate curve and field operations that are both vital within pairing-based protocols. Both these approaches have lent to some extent on efficient hardware implementation [34,9,19] of arithmetic in $\mathbb{F}_q$. Finally, and at the other end of the design spectrum, one can consider implementation entirely in software. Utilising niche techniques for representation and arithmetic in $\mathbb{F}_{3^m}$ [25,3] and better known techniques drawn from experience with ECC for $\mathbb{F}_{2^n}$ and $\mathbb{F}_p$, one can efficiently evaluate pairings on general-purpose desktop [37] and embedded [38] processors.

Somewhere between purely software and co-processor-assisted implementation lies the technique of *instruction set extension* (ISE) [22]. The premise here is that after careful workload characterisation, it is possible to identify a small set of operations that dominate performance in a software implementation. By supporting these specific operations using additional or modified hardware and exposing their behaviour to the programmer via the instruction set architecture (ISA), performance can be significantly improved. This is possible with only minor penalties in terms of datapath disruption and logic overhead. In the context of ECC, the use of ISEs has focused on acceleration of arithmetic in $\mathbb{F}_q$; see for example [29,21]. Since the efficacy of $\mathbb{F}_q$ underpins the performance of pairing evaluation, one can clearly reuse this work to gain an advantage. Moreover, as pairings can be parameterised by several different types of base field, one can leverage the advantages of unified multiplier circuits [36,20,2].

To summarise, the implementation of a means for pairing evaluation depends on arithmetic in $\mathbb{F}_q$; if the performance of said arithmetic can be improved, one can expect incremental but non-trivial improvements in the cost of pairing evaluation. Our goal in this paper is the construction of an ISE for the SPARC V8 compliant LEON-2 processor that enables acceleration of pairings parameterised over $\mathbb{F}_{2^n}$, $\mathbb{F}_{3^n}$ and $\mathbb{F}_p$ for large p. To this end, we developed an efficient tri-field

arithmetic unit based on *redundant signed digit* (RSD) encoding [4]. Algorithms and hardware architectures for long integer modular multiplication using RSD encoding were first proposed by Takagi et al. [39,40]. Other work in this area includes that of Öztürk et al. [33], who designed a tri-field Montgomery multiplier; we are careful to compare and contrast our results with their work. The major advantage of hardware acceleration through instruction set extensions is the ability to use optimised algorithms for fast squaring (resp. fast cubing) in $\mathbb{F}_{2^n}$ (resp. $\mathbb{F}_{3^n}$). This is significant since squaring (resp. cubing) is fundamental to the overall performance of pairing evaluation.

The rest of this paper is organised as follows. In Section 2 we present an overview of cryptographic pairings. Our aim is to construct a flexible cost model to use in comparisons of performance; this is presented in Appendix A. Section 3 details algorithms for arithmetic in the three fields $\mathbb{F}_{2^n}$, $\mathbb{F}_{3^n}$ and $\mathbb{F}_p$ and Section 4 introduces our tri-field multiplier design. Section 5 describes the LEON-2 processor and the modifications required to accommodate the tri-field multiplier before a set of experimental results are analysed in Section 6. Finally, we present some conclusions in Section 7 that demonstrate a clear improvement in arithmetic and pairing performance for all three fields.

2 Cryptographic Pairings

Let E be an elliptic curve over a finite field $\mathbb{F}_q$, and let $\mathcal{O}$ denote the identity element of the associated group of rational points $E(\mathbb{F}_q)$. For a positive integer $l|\#E(\mathbb{F}_q)$ co-prime to q, let k be the minimal positive integer such that $l|(q^k-1)$; k is often called the embedding degree or security multiplier. Let $E(\mathbb{F}_q)[l]$ denote the subgroup of $E(\mathbb{F}_q)$ of all points whose order is divisible by l, and similarly for the degree k extension of $\mathbb{F}_q$. Thus, the Tate pairing of order l is a map from elements of two source groups to a target group

$$e_l : E(\mathbb{F}_q)[l] \times E(\mathbb{F}_{q^k})[l] \rightarrow \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^l$$

which satisfies the following properties

- For each $P \neq \mathcal{O}$ there exists $Q \in E(\mathbb{F}_{q^k})[l]$ such that $e_l(P,Q) \neq 1 \in \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^l$.
- For any integer n, $e_l([n]P,Q) = e_l(P,[n]Q) = e_l(P,Q)^n$ for all $P \in E(\mathbb{F}_q)[l]$ and $Q \in E(\mathbb{F}_{q^k})[l]$.
- Let $L = hl$. Then $e_l(P,Q)^{(q^k-1)/l} = e_L(P,Q)^{(q^k-1)/L}$.
- It is efficiently computable.

Since e_l is defined as taking $P \in E(\mathbb{F}_q)[l]$ and $Q \in E(\mathbb{F}_{q^k})[l]$ as input, it is common to define a distortion map Ψ to lift elements of $E(\mathbb{F}_q)[l]$ into elements of $E(\mathbb{F}_{q^k})[l]$; careful selection of the map permits specific optimisation within the algorithm to evaluate the pairing. Selection of parameters involves many subtle trade-offs between security and performance in source and target groups; it is far from clear which is the single best parameterisation. Ignoring issues of curve

generating and concentrating on performance of the pairing (instead of curve operations), one might opt to compare different selections for q by balancing the target group size. For example

$$\begin{array}{lcll} q = 2^m & \rightarrow & k = 4, m = 233 & \rightarrow \quad \log_2(q^k) \sim 932 \\ q = 3^m & \rightarrow & k = 6, m = 97 & \rightarrow \quad \log_2(q^k) \sim 922 \\ q = p & \rightarrow & k = 6, \log_2(p) \sim 160 & \rightarrow \quad \log_2(q^k) \sim 960. \end{array}$$

In reality, these parameters probably provide slightly less than the level of security typically required. However, they allow us to make somewhat reasoned comparisons later on; we derive a rough cost estimate for pairing evaluation in Appendix A.

3 Field Arithmetic

In this section we briefly review the basic algorithms for arithmetic operations in $\mathbb{F}_p$, $\mathbb{F}_{2^m}$, and $\mathbb{F}_{3^m}$. The elements of a prime field $\mathbb{F}_p$ can be represented by the integers from 0 to $p-1$. On the other hand, the elements of extension fields of characteristic two and three are commonly represented by binary and ternary polynomials, respectively. Addition and multiplication in $\mathbb{F}_p$ is performed modulo the prime p, while arithmetic in $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$ is carried out modulo an irreducible polynomial of degree exactly m with coefficients from the respective base field ($\mathbb{F}_2$ or $\mathbb{F}_3$).

Due to the large field orders used in pairing-based cryptography, the field elements can not be directly processed on a processor with a 32 or 64-bit datapath. Software implementations usually solve the mismatch between the operand length and the size of the processor datapath by storing the field elements in arrays of single-precision words (e.g. arrays of 32-bit unsigned integers) and using efficient arithmetic algorithms that manipulate these arrays with help of the instructions provided by the processor, e.g. (32×32)-bit multiply instructions [11]. In the following, we will denote the wordsize of the processor by w and the number of w-bit words required to store an n-bit field element by t, i.e. we have $t = \lceil n/w \rceil$. For example, an n-bit integer A can be stored in an array of t words, each consisting of w bits: $A = (A_{t-1}, \ldots, A_1, A_0)$. The words A_{t-1} and A_0 are the most and least significant word of A, respectively. Equation (1) specifies the relation between the integer A and the w-bit words A_i.

$$A = \sum_{i=0}^{t-1} A_i \cdot 2^{i \cdot w} = A_{t-1} \cdot 2^{(t-1) \cdot w} + \cdots + A_1 \cdot 2^w + A_0 \tag{1}$$

The elements of $\mathbb{F}_{2^m}$ are polynomials of degree up to $m-1$ with coefficients from $\mathbb{F}_2 = \{0, 1\}$. A binary polynomial of degree $m-1$ can be represented by a bitstring of length m in which each bit corresponds to a coefficient. This bitstring can be stored in an array of w-bit words, similar to a long integer. The same holds for elements of $\mathbb{F}_{3^m}$, but it must be considered that two bits are necessary for each coefficient of a ternary polynomial. Therefore, the number of w-bit words required for storing a ternary polynomial of degree $m-1$ is $t = \lceil 2m/w \rceil$.

3.1 Addition and Subtraction

Both addition and subtraction of two elements of a prime field is straightforward to implement. The long integers representing the field elements are added/subtracted, followed by a reduction modulo the prime p if the result is not within the interval $[0, p-1]$. This reduction is simply done by subtracting or adding p. Many processor architectures, including SPARC, feature an add-with-carry and subtract-with-borrow instruction to facilitate long integer arithmetic.

Addition of elements of $\mathbb{F}_{2^m}$ is simply a logical XOR operation and does not require a reduction. Addition and subtraction in ternary extension fields is more complex than addition in the two other field types [25]. An addition of ternary polynomials requires to add the coefficients modulo 3, which is relatively costly since this operation is not supported by general-purpose processors.

3.2 Multiplication and Squaring/Cubing

Multiplication in $\mathbb{F}_p$ is done by multiplying the two integers representing the field elements and then reducing the product modulo the prime p. Long integer multiplication can be performed through operand scanning or product scanning [24]. In the following we will discuss one method in more detail, namely product scanning, which is based on Comba's multiplication technique [11].

Algorithm 1. Comba multiplication [11].

Input: Two t-word integers $A = (A_{t-1}, \ldots, A_1, A_0)$ and $B = (B_{t-1}, \ldots, B_1, B_0)$.
Output: The $2t$-word product $C = A \cdot B = (C_{2t-1}, \ldots, C_1, C_0)$.

```
1:  S ← 0
2:  for i from 0 to t − 1 do
3:      for j from 0 to i do
4:          S ← S + A_j · B_{i−j}
5:      end for
6:      C_i ← S mod 2^w
7:      S ← S/2^w
8:  end for
9:  for i from t to 2t − 2 do
10:     for j from i − t + 1 to t − 1 do
11:         S ← S + A_j · B_{i−j}
12:     end for
13:     C_i ← S mod 2^w
14:     S ← S/2^w
15: end for
16: C_{2t−1} ← S mod 2^w
```

Comba's method [11] for multiple-precision multiplication is shown in Algorithm 1. It has a nested loop structure and accumulates the inner-product terms $A_j \cdot B_{i-j}$ on a column-by-column basis. The operation carried out in the

inner loop of this algorithm is a *multiply-and-accumulate* operation. That is, two w-bit words A_i, B_j are multiplied and the $2w$-bit product is then added to a running sum S. As this sum can become $2w + \lceil \log_2(t) \rceil$ bits long, three word-size registers are needed to hold it during the computation [24]. Algorithm 1 performs a total of t^2 single-precision multiplications when A and B consist of t words.

Long integer squaring is a special case of long integer multiplication and allows for optimisation to reduce the execution time. Algorithm 1 can be rewritten such that only $(t^2 + t)/2$ single-precision multiplications to need to be carried out when $A = B$, which corresponds to a reduction of almost 50% compared to the t^2 single-precision multiplications when multiplying two distinct integers. In practice, however, squaring is only slightly faster than multiplication.

Multiplication in $\mathbb{F}_{2^m}$ requires to multiply two binary polynomials, followed by a reduction modulo an irreducible polynomial. A well-known algorithm for polynomial multiplication over $\mathbb{F}_2$ is the so-called *shift-and-xor* method [24]. The multiplicand is scanned coefficient-wise and the multiplier added to a running sum, depending on the value of the coefficient. More advanced methods, like the *right-to-left comb* method [24], use look-up tables to reduce the number of shifts and xor operations. However, if the target processor provides an instruction for word-level multiplication of binary polynomials, then two elements of $\mathbb{F}_{2^m}$ can be multiplied in a similar way as shown in Algorithm 1. Squaring of a binary polynomial is a linear operation and hence much faster than multiplication.

Multiplication in $\mathbb{F}_{3^m}$ can be performed in a similar way as in $\mathbb{F}_{2^m}$, namely through polynomial multiplication and reduction modulo an irreducible polynomial. All algorithms for the multiplication of binary polynomials, ranging from the shift-and-xor method to the left-to-right comb method, can be adapted to work for the multiplication of ternary polynomials as well. Moreover, Comba's multiplication technique could also be used for multiplying ternary polynomials if the processor provides a suitable word-level multiply instruction. Unfortunately, this is not the case for today's general-purpose processors, but a custom instruction for word-level multiplication of ternary polynomials can be easily integrated into any standard RISC architecture, as will be demonstrated in the following sections. Cubing a ternary polynomial is a linear operation, similar to squaring of a binary polynomial, and thus much faster than multiplication.

3.3 Modular Reduction

A widely-used algorithm for modular reduction of integers is due to Montgomery [32]. Montgomery's algorithm is a generic modular reduction method that works for any odd prime. However, certain primes, like pseudo-Mersenne primes or generalised-Mersenne primes, facilitate faster reduction methods. A reduction modulo such special primes can be performed efficiently with additions and shift operations [24].

Polynomial modular reduction is also very fast if the irreducible polynomial has few non-zero coefficients. Of particular importance are irreducible trinomials and pentanomials since they allow to accomplish a reduction with simple shift operations and polynomial additions (see [24] for further details).

4 Tri-field Multiplier

Recent research [36,20,2] has provided a number of so-called unified multipliers that reuse elements of the datapath for different field types. They rely on multi-field adder cells (e.g. dual-field adders) as basic building block. Most previous work focussed on the unification of integers and binary polynomials. The first unified multiplier for integers, binary and ternary polynomials was presented by Öztürk et al. [33]. Multiplier designs making use of unified components are efficient in terms of silicon area as the adder cells are reused for different types of operands. We exploit the advantages of this approach and introduce a unified multiply-accumulate (MAC) unit able to support our custom instruction set.

4.1 Redundant Signed Digit (RSD) Representation

The design of fast parallel multipliers relies on efficient arithmetic that restricts the carry propagation in addition to a few positions. Previous work examined the use of (3:2) counters or (4:2) compressors and *carry-save* representation. A different approach is to use a signed representation. Instead of splitting the sum of two numbers into carry and sum vectors, each number is split into a positive and a negative part. This representation is called *borrow-save* [5] and follows the relation

$$\mathbf{X} = \sum_{i=0}^{n-1} x_i \cdot 2^i = \sum_{i=0}^{n-1} (x_i^+ - x_i^-) \cdot 2^i \tag{2}$$

where $\mathbf{X}$ denotes an n-digit number with digit set $\{-1, 0, 1\}$. Such digit systems were first studied by Avizienis [4] and are called *redundant signed digit* (RSD) systems. This digit set gives an advantage over the two's complement form as it naturally handles signed numbers, which simplifies the design of a multiplier for signed and unsigned integers. Furthermore, the RSD system allows a straightforward representation of the coefficients of ternary polynomials.

RSD addition of radix-2 numbers is performed in two steps whereby in each step a carry can occur. Hence, the maximum carry propagation distance is restricted to two digits, which facilitates efficient hardware implementation. Our design of the RSD adder performs the addition of integers in the conventional way following the rules of RSD arithmetic. The addition of binary polynomials uses a different recoding to prevent carry propagation. Ternary polynomials, on the other hand, are recoding after each step to ensure that the coefficients of the result remain in $\mathbb{F}_3$.

4.2 Architecture of the Multiply-Accumulate Unit

Figure 1 shows the main components of our unified MAC unit. It consists of a unified multiplier followed by a unified adder acting as accumulator. The MAC unit is pipelined, and hence we need to store the result of the multiplier for one cycle. Our design is scalable to any operand length and allows for different trade-offs between silicon area and critical path delay. We explored two specific

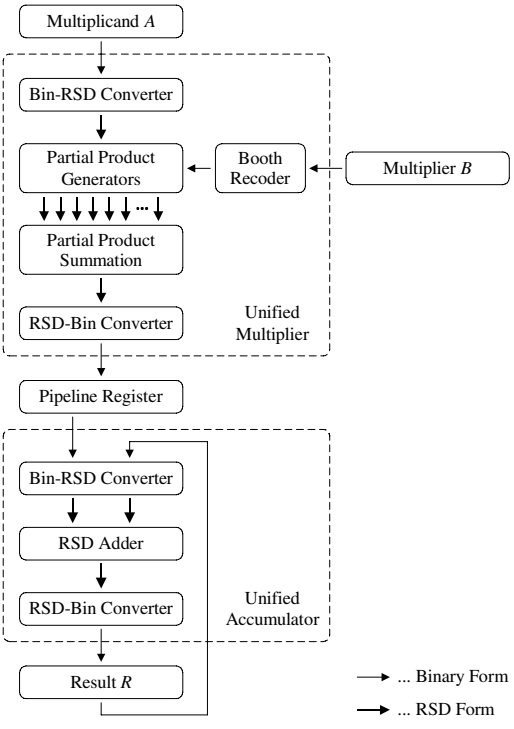


Fig. 1. Main components of the unified multiply-accumulate unit

implementations of the MAC unit; the first contains a (32×16)-bit multiplier and the second a smaller (32×8)-bit multiplier. These multipliers perform a full (32×32)-bit multiplication in two and four clock cycles, respectively.

Parallel multipliers can be implemented in from of an array structure or a tree structure. Array multipliers feature high regularity and short interconnect wires, while tree multipliers have a smaller critical path delay. When using RSD representation, this trade-off can be crucial since a single RSD adder cell is more complex than a conventional full adder. Due to negatively weighted inputs to adders, common techniques developed for the summation of partial products are difficult to apply in this setting. We implemented both the array and the tree architecture to assess the resulting area and delay complexities.

As shown in Figure 1, the result of the multiplication is converted from RSD to binary representation before it is buffered in the pipeline register, and then converted back into RSD form. These conversions increase the critical path, but reduce the silicon area since the binary representation requires only half of the storage that would be needed for RSD representation. A typical standard-cell implementation of the MAC unit has a delay between 12 and 22 ns, depending on the architecture and size of the multiplier (see Section 6 for details). These delays correspond to a maximum clock frequency between 45 and 83 MHz, which

is reasonable for embedded devices like smart cards. Higher clock frequencies are possible through the integration of additional pipeline registers.

5 Extended LEON-2 Processor

SPARC V8 [43] is a general-purpose RISC architecture with a 32-bit datapath and a "windowed" register file containing an implementation-dependent number of general-purpose registers (GPRs), of which 32 are visible to the programmer at a time. Besides the GPRs, the SPARC V8 architecture also includes several special-purpose registers like the multiply-divide register (`%y`) and a total of 31 ancillary state registers (`%asr1` to `%asr31`).

The SPARC instruction set contains Delayed Control Transfer Instructions (DCTI). In particular, branches and calls have an architectural delay slot of one instruction, which means that the instruction immediately following a DCTI is executed (unless the DCTI annuls it) before the control transfer to the target address is completed. Arithmetic and logical instructions have a conventional three-operand format with two source registers and a single destination register [43]. Multiply instructions like `smul` and `umul` write the 32 least significant bits of the product to a destination register and the 32 most significant bits to the multiply-divide register (`%y`). The `rdy` instruction allows to transfer the content of register `%y` to a GPR.

5.1 Main Characteristics of the LEON-2 Core

The LEON-2 processor [15] is a configurable and synthesizable VHDL implementation of the SPARC V8 instruction set. Originally developed by the European Space Agency, the LEON-2 softcore is now maintained by Gaisler Research and has found widespread use in system-on-chip (SOC) designs in recent years. The LEON-2 VHDL model is highly configurable; various options like the number of register windows, the size and organisation of caches, and performance/area trade-offs for the integer multiplier can be defined through a single configuration file. In addition, the LEON-2 core is extensible since the full VHDL source code is available under the GNU LGPL license.

The LEON-2 pipeline can be configured to have either one or two load delay cycles. We used a LEON-2 core with one load delay cycle as this configuration allows to achieve better performance in FPGAs. The LEON-2 processor also contains a hardware multiplier that can be configured to perform a (32×32)-bit integer multiplication in either 35, 4, 2, or 1 clock cycles.

5.2 Custom Instructions

The extensions for pairing-based cryptography we propose in this paper include a total of five custom instructions to accelerate arithmetic operations in prime fields, binary fields, and ternary fields. Table 1 gives an overview of the instructions and summarises the operations they perform. The first two instructions

Table 1. Custom instructions for pairing-based cryptography

Format	Description	Operation
`umac rs1, rs2`	Unsigned Mul. and Acc.	$accu \leftarrow accu + rs1 \times rs2$
`umac2 rs1, rs2`	Unsigned Mul. and Acc. Twice	$accu \leftarrow accu + 2(rs1 \times rs2)$
`shacr rd`	Shift Accu Registers Right	$rd \leftarrow accu[31:0]\,;\ accu \leftarrow accu \gg 32$
`gf2mac rs1, rs2`	Binary Poly. Mul. and Acc.	$accu \leftarrow accu \oplus rs1 \otimes rs2$
`gf3mac rs1, rs2`	Ternary Poly. Mul. and Acc.	$accu \leftarrow accu + rs1 \times rs2$

(`umac` and `umac2`) allow to speed up the inner loop operations of long integer multiplication and squaring according to Comba's algorithm. The instructions `gf2mac` and `gf3mac` can be used to implement the multiplication of binary and ternary polynomials, respectively.

The `umac` instruction performs a MAC operation on unsigned 32-bit integers. More precisely, `umac` multiplies the content of two GPRs, treating both operands as unsigned integers, and adds the 64-bit product to a cumulative sum stored in the three registers `%asr20`, `%y`, and `%asr18`, subsequently called *accu registers*. The cumulative sum is, in general, exceeding 64 bits in precision when several 64-bit products are summed up. Therefore, three 32-bit registers are needed to accommodate the cumulative sum, whereby the 32 least significant bits are stored in `%asr18`, the bits 32 through 63 in register `%y`, and the most significant bits in `%asr20`, respectively. After adding the 64-bit product to the cumulative sum, the result is written back to the accu registers (see Figure 2). The custom instruction `shacr` allows to shift the cumulative sum held in the accu registers 32 bits to the right, whereby the least significant 32-bit word of the cumulative sum (i.e. the content of `%asr18`) is written to a destination register `rd`.

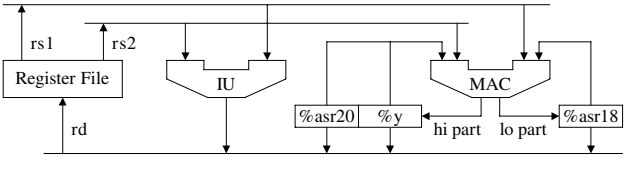


Fig. 2. Datapath consisting of integer unit (IU) and MAC unit with accu registers

We implemented a "pairing-friendly" MAC unit for the LEON-2 core consisting of a (32×16)-bit tree multiplier and a 72-bit accumulator. The 72-bit accumulator guarantees that up to 256 double-precision (i.e. 64-bit) products can be summed up without overflow or loss of precision, which is sufficient for cryptographic applications. Besides the custom instructions shown in Table 1, the MAC unit is also capable to execute the "native" SPARC multiply instructions like `umul` and `smul` [23]. Therefore, the proposed extensions for pairing-based cryptography can be easily integrated into the LEON-2 core by simply replacing the integer multiplier with a MAC unit that provides the extra functionality. In

addition to modifications of the LEON-2 core, we also adapted the tool-chain, in particular the GNU assembler `gas`, to support the custom instructions.

A LEON-2 core equipped with a (32×16+72)-bit MAC unit executes the "native" SPARC V8 multiply instructions `smul/umul` in two clock cycles, whereby higher part of the product is written to the `%y` register, while the lower part is directed to a GPR in the register file. The custom instruction `umac` also has a latency of two cycles, but places its result in the accu registers (and not in a GPR), and therefore an independent instruction can be executed in the integer unit during the second cycle of a `umac` instruction [23]. This parallel execution is possible since the buses connecting the register file and the functional units are not occupied during the second cycle of a `umac` instruction, similar to the execution of the `madd` instruction in MIPS32 processors.

6 Experimental Results

We prototyped the extended LEON-2 processor on a Xess XSV800 board which houses a Xilinx Virtex FPGA providing about 800k gates. The LEON-2 source code contains scripts and configuration files for several FPGA boards, including the XSV800. We used Xilinx XST 8.3 to perform the synthesis. As mentioned in Section 4, we implemented several versions of the MAC unit with different multiplier dimensions (32×16 bit, 32×8 bit) and structures (array, tree). In order to assess area and delay of the different implementations, we synthesised the MAC unit not only as part of the LEON-2, but also as stand-alone circuit using a 0.35 μm standard cell library. The results of these synthesis runs are summarised in Table 2. The gate equivalents were calculated taking a 2-NAND gate from the same library as reference. This gate has an area of 55 μm^2.

Table 2. Synthesis results of different implementations of the MAC unit

MAC Type	Silicon Area	Delay	Max. Frequency	Latency
(32×16)-array	16,400 GE	22 ns	45 MHz	2 cycles
(32×16)-tree	16,200 GE	16 ns	62 MHz	2 cycles
(32×8)-array	11,900 GE	14 ns	71 MHz	4 cycles
(32×8)-tree	12,700 GE	12 ns	83 MHz	4 cycles

Besides area and delay, we also evaluated the performance gain due to the integration of the MAC unit and the custom instructions. Table 3 and 4 summarise the execution times of arithmetic operations in $\mathbb{F}_p$, $\mathbb{F}_{2^m}$, and $\mathbb{F}_{3^m}$ when using native SPARC instructions and the extended instruction set, respectively. Our reference implementation (Table 3) is based on a ANSI C library for arithmetic in $\mathbb{F}_p$, $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$ that was developed at the University of Bristol. It uses Montgomery multiplication in $\mathbb{F}_p$ and features a number of Assembler macros for performance-critical operations. The multiplication of binary polynomials is performed via a recursive Karatsuba technique, while the polynomial squaring

Table 3. Arithmetic performance (in clock cycles) of the SPARC V8 instruction set

Field Arithmetic in $\mathbb{F}_{2^n}$			
Field	Addition	Multiplication	Squaring
$\mathbb{F}_{2^{163}}$	162	4,978	1,468
$\mathbb{F}_{2^{233}}$	197	7,010	1,463
$\mathbb{F}_{2^{283}}$	215	14,531	1,861
$\mathbb{F}_{2^{353}}$	275	14,154	2,359
$\mathbb{F}_{2^{457}}$	323	21,496	2,840
$\mathbb{F}_{2^{557}}$	413	42,863	3,264
Field Arithmetic in $\mathbb{F}_{3^m}$			
Field	Addition	Multiplication	Cubing
$\mathbb{F}_{3^{79}}$	96	8,992	1,360
$\mathbb{F}_{3^{97}}$	116	13,460	1,448
$\mathbb{F}_{3^{163}}$	156	26,779	1,996
$\mathbb{F}_{3^{193}}$	176	35,690	2,251
$\mathbb{F}_{3^{239}}$	196	39,855	2,448
$\mathbb{F}_{3^{353}}$	276	79,195	3,650
Field Arithmetic in $\mathbb{F}_p$			
Field	Addition	Multiplication	Squaring
$\mathbb{F}_p, \log_2(p) = 160$	69	1,183	1,183
$\mathbb{F}_p, \log_2(p) = 192$	80	1,334	1,334
$\mathbb{F}_p, \log_2(p) = 224$	92	1,618	1,618
$\mathbb{F}_p, \log_2(p) = 256$	104	1,907	1,907
$\mathbb{F}_p, \log_2(p) = 384$	152	3,866	3,866
$\mathbb{F}_p, \log_2(p) = 512$	200	6,254	6,089

is done via table look-up. Ternary polynomials use a bit-sliced representation where high and low parts of the coefficients are stored in separate vectors. Multiplication of ternary polynomials is also based on the Karatsuba approach and cubing on the table look-up method. The addition of ternary polynomials is realised in a straightforward way. We compiled the library using a GCC cross compiler for the SPARC V8 architecture with optimisations enabled. All timings shown in Table 3 were measured under warm cache conditions with help of the built-in cycle counter (register `%cycnt`) of the modified LEON-2 core.

Table 4 shows the cycle counts of the arithmetic operations when using our custom instructions described in Section 5. In short, the custom instructions accelerate multiplication in $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$ by a factor of (at least) 10 and 20, respectively, while multiplication in $\mathbb{F}_p$ achieves a two-fold performance gain.

7 Conclusions

Considering the results from the previous section in the context of approximate costs for pairing evaluation given in Appendix A, the value of ISE and our tri-field MAC unit is clear. Specifically, for similar sized base fields outlined in

Table 4. Arithmetic performance (in clock cycles) of the extended instruction set

Field Arithmetic in $\mathbb{F}_{2^n}$			
Field	Addition	Multiplication	Squaring
$\mathbb{F}_{2^{163}}$	65	415	166
$\mathbb{F}_{2^{233}}$	81	634	207
$\mathbb{F}_{2^{283}}$	89	778	244
$\mathbb{F}_{2^{353}}$	113	1,238	311
$\mathbb{F}_{2^{457}}$	137	1,806	378
$\mathbb{F}_{2^{557}}$	161	2,545	508
Field Arithmetic in $\mathbb{F}_{3^m}$			
Field	Addition	Multiplication	Cubing
$\mathbb{F}_{3^{79}}$	65	422	475
$\mathbb{F}_{3^{97}}$	85	637	593
$\mathbb{F}_{3^{163}}$	125	1,257	927
$\mathbb{F}_{3^{193}}$	125	1,295	984
$\mathbb{F}_{3^{239}}$	165	2,069	1,261
$\mathbb{F}_{3^{353}}$	245	4,247	1,896
Field Arithmetic in $\mathbb{F}_p$			
Field	Addition	Multiplication	Squaring
$\mathbb{F}_p, \log_2(p) = 160$	68	498	498
$\mathbb{F}_p, \log_2(p) = 192$	80	574	574
$\mathbb{F}_p, \log_2(p) = 224$	92	759	759
$\mathbb{F}_p, \log_2(p) = 256$	104	913	913
$\mathbb{F}_p, \log_2(p) = 384$	152	1,883	1,883
$\mathbb{F}_p, \log_2(p) = 512$	200	3,094	3,094

Section 2 (and ignoring issues of curve generation) we find that the number of clock cycles required to evaluate a pairing in $\mathbb{F}_{2^n}$ is reduced from $6,762,354$ to $646,554$, in $\mathbb{F}_{3^n}$ from $12,061,117$ to $981,984$, and in $\mathbb{F}_p$ from $10,788,960$ to $4,541,760$. Clearly these are only estimates, but they equate to between a two and twelve-fold saving depending on the field choice, a significant improvement for such little overhead. Further, utilising the tri-field MAC greatly reduces the cost ratio between $\mathbb{F}_{3^n}$ and other choices. Previously this parameterisation was at least twice as slow as the selection of $\mathbb{F}_{2^n}$, for example, with this gap widening for larger n. However, with custom instructions and the integration of our tri-field MAC unit, the two choices are roughly comparable in performance.

The advantages of this result are two-fold. Firstly, with only minor modification to the processor datapath and ISA one can significantly improve the performance of pairing evaluation. Unlike some dedicated hardware accelerators, this improvement costs only a moderate size overhead and is useful in accelerating operations within the pairing, at the protocol level and within vanilla ECC. Secondly, the acceleration of pairings over $\mathbb{F}_{3^n}$, which were previously prohibitively slow, allows a more free choice of parameterisation: if this is a good choice for the application, it need not be restrictive in terms of performance.

References

1. Ahmadi, O., Hankerson, D., Menezes, A.: Formulas for Cube Roots in $\mathbb{F}_{3^m}$. Available at: http://www.cacr.math.uwaterloo.ca/ajmeneze/publications/cuberoots.pdf
2. Au, L.-S., Burgess, N.: Unified Radix-4 Multiplier for $GF(p)$ and $GF(2^n)$. In: Application-Specific Systems, Architectures and Processors (ASAP), pp. 226–236. IEEE Press, Los Alamitos (2003)
3. Austrin, P.: Efficient Arithmetic in Finite Fields of Small, Odd Characteristic. MSc Thesis, Royal Institute of Technology, Stockholm (2004)
4. Avizienis, A.: Signed-Digit Number Representations for Fast Parallel Arithmetic. IRE Transactions on Electronic Computers 10(9), 389–400 (1961)
5. Bajard, J.-C., Duprat, J., Kla, S., Muller, J.-M.: Some Operators for On-Line Radix-2 Computations. Journal of Parallel and Distributed Computing 22(2), 336–345 (1994)
6. Barreto, P.S.L.M., Galbraith, S., ÓhÉigeartaigh, C., Scott, M.: Efficient Pairing Computation on Supersingular Abelian Varieties. In: Cryptology ePrint Archive, Report 2004/375 (2004)
7. Barreto, P.S.L.M., Kim, H., 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)
8. Bertoni, G., Breveglieri, L., Fragneto, P., Pelosi, G.: Parallel Hardware Architectures for the Cryptographic Tate Pairing. In: Information Technology: New Generations (ITNG), pp. 186–191. IEEE Press, Los Alamitos (2006)
9. Bertoni, G., Guajardo, J., Kumar, S., Orlando, G., Paar, C., Wollinger, T.: Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 158–175. Springer, Heidelberg (2003)
10. Beuchat, J.-L., Shirase, M., Takagi, T., Okamoto, E.: An Algorithm for the η_T Pairing Calculation in Characteristic Three and its Hardware Implementation. In: Cryptology ePrint Archive, Report 2006/327 (2006)
11. Comba, P.G.: Exponentiation cryptosystems on the IBM PC. IBM Systems Journal 29(4), 526–538 (1990)
12. Dutta, R., Barua, R., Sarkar, P.: Pairing-Based Cryptographic Protocols: A Survey. In: Cryptology ePrint Archive, Report 2004/064 (2004)
13. Duursma, I., Lee, H.: Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
14. Fong, K., Hankerson, D., López, J., Menezes, A.: Field Inversion and Point Halving Revisited. Technical Report CORR 2003-18, University of Waterloo (2003)
15. Gaisler, J.: The LEON-2 Processor User's Manual (Version 1.0.30) (July 2005), Available for download at http://www.gaisler.com
16. Galbraith, S., Harrison, K., Soldera, D.: Implementing the Tate pairing. In: Fieker, C., Kohel, D.R. (eds.) Algorithmic Number Theory (ANTS-V). LNCS, vol. 2369, pp. 324–337. Springer, Heidelberg (2002)
17. Grabher, P., Page, D.: Hardware Acceleration of the Tate Pairing in Characteristic Three. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 398–411. Springer, Heidelberg (2005)
18. Granger, R., Page, D., Smart, N.P.: High Security Pairing-Based Cryptography Revisited. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory (ANTS-VII). LNCS, vol. 4076, pp. 480–494. Springer, Heidelberg (2006)

19. Granger, R., Page, D., Stam, M.: Hardware and Software Normal Basis Arithmetic for Pairing Based Cryptography in Characteristic Three. IEEE Transactions on Computers 54(7), 852–860 (2005)
20. Großschädl, J.: A Bit-Serial Unified Multiplier Architecture for Finite Fields $GF(p)$ and $GF(2^m)$. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 202–218. Springer, Heidelberg (2001)
21. Großschädl, J., Kumar, S., Paar, C.: Architectural Support for Arithmetic in Optimal Extension Fields. In: Application-Specific Systems, Architectures and Processors (ASAP), pp. 111–124. IEEE Press, Los Alamitos (2004)
22. Großschädl, J., Savaş, E.: Instruction Set Extensions for Fast Arithmetic in Finite Fields $GF(p)$ and $GF(2^m)$. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 133–147. Springer, Heidelberg (2004)
23. Großschädl, J., Tillich, S., Szekely, A.: Cryptography Instruction Set Extensions to the SPARC V8 Architecture. Preprint (submitted for publication, 2007)
24. Hankerson, D.R., Menezes, A.J., Vanstone, S.A.: Guide to Elliptic Curve Cryptography. Springer, Heidelberg (2004)
25. Harrison, K., Page, D., Smart, N.P.: Software Implementation of Finite Fields of Characteristic Three, for use in Pairing Based Cryptosystems. LMS Journal of Computation and Mathematics 5(1), 181–193 (2002)
26. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. Transactions on Information Theory 52, 4595–4602 (2006)
27. Kerins, T., Marnane, W.P., Popovici, E.M., Barreto, P.S.L.M.: Efficient Hardware for the Tate Pairing Calculation in Characteristic Three. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 412–426. Springer, Heidelberg (2005)
28. Kerins, T., Popovici, E., Marnane, W.P.: Algorithms and Architectures for Use in FPGA Implementations of Identity Based Encryption Schemes. In: Becker, J., Platzner, M., Vernalde, S. (eds.) FPL 2004. LNCS, vol. 3203, pp. 74–83. Springer, Heidelberg (2004)
29. Kumar, S., Paar, C.: Reconfigurable Instruction Set Extension for Enabling ECC on an 8-Bit Processor. In: Becker, J., Platzner, M., Vernalde, S. (eds.) FPL 2004. LNCS, vol. 3203, pp. 586–595. Springer, Heidelberg (2004)
30. Kwon, S.: Efficient Tate Pairing Computation for Elliptic Curves over Binary Fields. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 134–145. Springer, Heidelberg (2005)
31. Miller, V.: Short programs for functions on curves. Available at: http://crypto.stanford.edu/miller/miller.pdf
32. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44(170), 519–521 (1985)
33. Öztürk, E., Savas, E., Sunar, B.: A Versatile Montgomery Multiplier Architecture with Characteristic Three Support. Available at: http://ece.wpi.edu/ sunar/preprints/versatile.pdf
34. Page, D., Smart, N.P.: Hardware Implementation of Finite Fields of Characteristic Three. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 529–539. Springer, Heidelberg (2003)
35. Ronan, R., ÓhÉigeartaigh, C., Murphy, C., Scott, M., Kerins, T., Marnane, W.P.: An Embedded Processor for a Pairing-Based Cryptosystem. In: Information Technology: New Generations (ITNG), pp. 192–197. IEEE Press, Los Alamitos (2006)
36. Savas, E., Tenca, A.F., Koç, Ç.K.: A Scalable and Unified Multiplier Architecture for Finite Fields $GF(p)$ and $GF(2^m)$. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 277–295. Springer, Heidelberg (2000)

37. Scott, M.: Implementing Cryptographic Pairings, Available at: ftp://ftp.computing.dcu.ie/pub/resources/crypto/pairings.pdf
38. Scott, M., Costigan, N., Abdulwahab, W.: Implementing Cryptographic Pairings on Smartcards. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 134–147. Springer, Heidelberg (2006)
39. Takagi, N., Yajima, S.: Modular Multiplication Hardware Algorithms with a Redundant Representation and their Application to RSA Cryptosystem. IEEE Transactions on Computers 41(7), 887–891 (1992)
40. Takagi, N.: A Radix-4 Modular Multiplication Hardware Algorithm for Modular Exponentiation. IEEE Transactions on Computers 41(8), 949–956 (1992)
41. Shirase, M., Takagi, T., Okamoto, E.: Some Efficient Algorithms for the Final Exponentiation of η_T Pairing. In: Cryptology ePrint Archive, Report 2006/431 (2006)
42. Shu, C., Kwon, S., Gaj, K.: FPGA Accelerated Tate Pairing Based Cryptosystems over Binary Fields. In: Cryptology ePrint Archive, Report 2006/179 (2006)
43. SPARC International, Inc. The SPARC Architecture Manual Version 8 (August 1993) Available for download at http://www.sparc.org/standards/V8.pdf

A The Cost of Pairing Evaluation

We are generally interested in algorithms for pairing evaluation from the perspective of operation count rather than rigorous mathematical definition; this stems from our focus on implementation of arithmetic in $\mathbb{F}_q$. As such, we lean on other work to provide the most current, most efficient cost model. Let $\mathcal{A}_f$, $\mathcal{S}_f$, $\mathcal{C}_f$, $\mathcal{M}_f$ and $\mathcal{I}_f$ denote the cost of computing addition, squaring, cubing, multiplication and inversion in some finite field f. We quote a given cost model in terms of dominant operations; this typically means neglecting the cost of $\mathcal{A}_f$ for example, and concentrating on $\mathcal{M}_f$ and $\mathcal{I}_f$. Thanks to efficient methods for computing square roots [14] and cube roots [1] in $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$, respectively, we estimate their cost to be equivalent to squaring/cubing in the same fields; this makes sense given the usual opportunity to pre-compute square roots (resp. cube roots) via repeated squaring (resp. cubing).

We do not allow for pre-computation based on fixed values of either input to the pairing. Further, we are not concerned with compatibility, specifically we assume that if one utilises the Eta or Ate pairings their output need not be further powered to provide the same result as a comparable invocation of the Tate pairing.

Eta Pairing in Characteristic 2. Following Barreto et al. [6], consider the Eta pairing with source groups instantiated using the supersingular curve

$$E : y^2 + y = x^3 + x + b$$

over $\mathbb{F}_{2^m}$ where $b \in \{0, 1\}$ and the embedding degree $k = 4$. One can construct a tower of fields to form the required extension via $\mathbb{F}_{2^{2m}} = \mathbb{F}_{2^m}[\alpha]/(\alpha^2 - \alpha + 1)$ and $\mathbb{F}_{2^{4m}} = \mathbb{F}_{2^{2m}}[\beta]/(\beta^2 - \beta - \alpha)$. The distortion map $\Psi(x, y) = (\alpha^2 + x, \beta + \alpha x + y)$. The cost of evaluating the Eta pairing can be approximated by

$$(7(m + 1)/2)\mathcal{M}_{\mathbb{F}_{2^m}} + (4(m + 1)/2)\mathcal{S}_{\mathbb{F}_{2^m}}$$

while the final powering costs roughly

$$36\mathcal{M}_{\mathbb{F}_{2^m}} + \mathcal{I}_{\mathbb{F}_{2^{4m}}}.$$

Estimating $\mathcal{I}_{\mathbb{F}_{2^{4m}}} \sim 12\mathcal{M}_{\mathbb{F}_{3^m}}$, we get an overall cost of roughly

$$(3.5m + 51.5)\mathcal{M}_{\mathbb{F}_{2^m}} + (2m + 2)\mathcal{S}_{\mathbb{F}_{2^m}}.$$

For $m = 233$ this yields

$$867\mathcal{M}_{\mathbb{F}_{2^m}} + 468\mathcal{S}_{\mathbb{F}_{2^m}}.$$

Eta Pairing in Characteristic 3. Following Shirase et al. [41], consider the Eta pairing with source groups instantiated using the supersingular curve

$$E : y^2 = x^3 - x + b$$

over $\mathbb{F}_{3^m}$ where $b \in \{-1, +1\}$ and the embedding degree $k = 6$. One can construct a tower of fields to form the required extension via $\mathbb{F}_{3^{3m}} = \mathbb{F}_{3^m}[\rho]/(\rho^3 - \rho - b)$ and $\mathbb{F}_{3^{6m}} = \mathbb{F}_{3^{3m}}[\sigma]/(\sigma^2 + 1)$. The distortion map $\Psi(x, y) = (\rho - x, \sigma y)$. Shirase et al. [41][Table 2] show that evaluation of the Eta pairing with this parameterisation can cost as little as

$$(7.5m + 68.5)\mathcal{M}_{\mathbb{F}_{3^m}} + (8m + 8)\mathcal{C}_{\mathbb{F}_{3^m}} + \mathcal{I}_{\mathbb{F}_{3^{3m}}}$$

where they estimate $\mathcal{I}_{\mathbb{F}_{3^{3m}}} \sim 15.73\mathcal{M}_{\mathbb{F}_{3^m}}$ to get an overall cost of roughly

$$(7.5m + 84.23)\mathcal{M}_{\mathbb{F}_{3^m}} + (8m + 8)\mathcal{C}_{\mathbb{F}_{3^m}}.$$

For $m = 97$ this yields

$$811.73\mathcal{M}_{\mathbb{F}_{3^m}} + 784\mathcal{C}_{\mathbb{F}_{3^m}}.$$

Ate Pairing in Characteristic *p*. Using curve-specific optimisations, the Ate [26] pairing redefines the bilinear map as

$$\hat{e_l} : E(\mathbb{F}_p) \times \overline{E}(\mathbb{F}_{p^{k/2}}) \rightarrow \mathbb{F}^*_{p^k}$$

where $\overline{E}$ is the quadratic twist of an elliptic curve E defined over $\mathbb{F}_{p^{k/2}}$. As such, efficient arithmetic in $\mathbb{F}_{p^{k/2}}$ and $\mathbb{F}_{p^k}$ is required, both underpinned by arithmetic in $\mathbb{F}_p$. Granger et al. [18] describe a range of options for arithmetic in different instantiations of these fields; selection of $log_2(p) \sim 256$ and $k = 6$ implies Case A of [18][Section 5] which estimates the cost of pairing evaluation to be

$$9120\mathcal{M}_{\mathbb{F}_p}.$$

The Importance of the Final Exponentiation in Pairings When Considering Fault Attacks

Claire Whelan and Michael Scott

School of Computing, Dublin City University
Ballymun, Dublin 9. Ireland
{cwhelan*,mike}@computing.dcu.ie

Abstract. We investigate the possibilities for injecting faults on pairings and assess their consequences. We assess the effect of faults that seek to corrupt the data being operated on and show that pairings with either no or a straightforward final exponentiation are less secure than pairings with a more complex final exponentiation when considering such fault attacks. As evidence, we describe two types of fault attacks on the Weil and η pairing that recover the secret point, which cannot be applied to the Tate pairing. This can be accredited to its more complex final exponentiation.

1 Introduction

In most applications based on pairing based cryptography, the secret key is one of the elliptic curve points input to the pairing. For example, in Boneh and Franklin's identity based encryption (IBE) [5] the critical operation involving the secret key in a pairing is the decryption operation. Therefore, it is necessary for the security of pairings to be investigated in both settings, contemporary security and side channel security, given that one of the input parameters to the pairing may be the secret key. Here, we investigate the security of pairings in the context of fault attacks.

Fault attacks are invasive attacks that actively seek to alter the normal execution of an algorithm. Fault attacks can be categorised as having three main effects on an algorithm. The first effect seeks to knock out a step in the computation, i.e. a no-op replaces another working instruction. This allows selective execution of instructions in an algorithm. The second effect seeks to cause a loop to either end prematurely or run over. The third effect seeks to cause the data being operated on to be corrupted in some way.

The first (and only to date) description of a fault attack on pairings was by Page and Vercauteren [18] when they demonstrated an attack on the Duursma-Lee [7] and Kwon-BGOS algorithm [2,14] for the Tate and η pairing. The fault type they focused on was one which caused the Miller loop to run over. By

* Supported by the Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 225–246, 2007.

inducing extra iterations they are able to isolate a single contribution to the Miller loop, which could be "picked apart" to find the secret.

In this paper, another fault type is focused on. We describe the effect of faults which seek to corrupt data in a range of pairing algorithms. We provide evidence that the final exponentiation is a vital operation in deterring such fault attacks by demonstrating that pairings having no or a simple final exponentiations are susceptible to data corruption fault attacks, whereas pairings with more complex final exponentiations are resistant to such attacks. The fault attack of Page and Vercauteren [18] did successfully attack a pairing with a complex final exponentiation. However, they describe a special case where in order to reverse the final exponentiation, they must exploit the fact that the additional contribution to the Miller loop which is induced by the fault, has a special redundant form, in that it is not a general element in the sextic extension field. Therefore, their method to reverse the final exponentiation is not applicable in many situations.

For demonstrative purposes we describe a successful sign change fault attack on the Weil pairing of [20] (with a simple final exponentiation) which reduces to solving a cubic equation, and a successful fault attack on the Galbraith et al. variant of the η pairing [8] (with no final exponentiation) which in one case reduces to solving a system of linear equations and in another to a simple congruence. We also provide evidence that these attacks cannot be applied to the Tate pairing because the final exponentiation makes it impossible.

The paper is structured as follows. Section 2 presents necessary background on pairings. Section 3 describes the theory behind our fault attack. We give concrete examples of this fault attack in section 4 when we present a fault attack on the η pairing and Weil pairing. We argue that pairings with a more complex final exponentiation are more secure than the former pairings by showing how the fault attacks described cannot be applied to the Tate pairing. Finally we discuss countermeasures and conclude in sections 5 and 6. Numerical examples of some of the attacks described are given in Appendices A and B.

2 Background

Let E be an elliptic curve over a finite field $\mathbb{F}_{q^k}$, and let r be an integer not divisible by the characteristic of $\mathbb{F}_{q^k}$. A pairing is function which maps a pair of elliptic curve points, $P, Q \in E(\mathbb{F}_{q^k})$, to an element in the underlying multiplicative finite field $\mathbb{F}^*_{q^k}$. Two founding pairings exist, namely the Weil [5] and the Tate [3] pairing. Specifically, the Weil pairing is defined as

$$\omega : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})[r] \rightarrow \mu_r$$

and the Tate pairing is defined as

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

where the embedding degree k is the smallest value of k such that $r|q^k - 1$ and $\mu_r = \{x \in \mathbb{F}^*_{q^k} | x^r = 1\}$ is the group of r-th roots of unity in $\mathbb{F}_{q^k}$. The output

of the Tate pairing is not unique and is a member of the coset $(\mathbb{F}_{q^k}^*)/(\mathbb{F}_{q^k}^*)^r$, which is defined up to r-th powers. To produce a unique value, the reduced Tate pairing is defined as

$$e(P,Q) = \langle \cdot, \cdot \rangle_r^{(q^k-1)/r} \tag{1}$$

where this additional operation of raising the output of $\langle \cdot, \cdot \rangle_r$ to the power of $(q^k - 1)/r$ is known as the final exponentiation.

Miller's algorithm is the most widely used technique to compute a bilinear pairing [16]. Miller's algorithm consists of a double and add algorithm for elliptic curve point multiplication with additional functionality. The input point P to Miller's algorithm is chosen such that it has order r. During the point scalar multiplication of the point $[r]P$, which upon completion should result in the point at infinity $\mathcal{O}$, a distance relationship between the lines produced from the point addition and a static point Q is calculated. If $l_{A,B}$ and v_{A+B} are the lines which arise in the addition rule for adding A to B to produce $C = A + B$, and the point A has coordinates (x_i, y_i), the point C has coordinates (x_{i+1}, y_{i+1}), the point Q has coordinates (x_Q, y_Q), and the line through A and B has a slope of λ_i, then explicitly

$$\begin{aligned} l_{A,B}(Q) &= (y_Q - y_i) - \lambda_i(x_Q - x_i) \\ v_{A+B}(Q) &= (x_Q - x_{i+1}). \end{aligned}$$

In each round of the Miller loop, $l_{A,B}$ is divided by v_{A+B} to produce an element in the extension field $m \in \mathbb{F}_{q^k}$, referred to as the Miller variable. This value is multiplicatively accumulated and eventually outputted from Miller's algorithm. Miller's algorithm is given in algorithm 1. The Weil pairing requires two applications of Miller's algorithm to produce a bilinear map, whereas the Tate pairing requires only one application followed by the final exponentiation.

Algorithm 1. Miller's Algorithm

INPUT: $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(\mathbb{F}_{q^k})$ where P is a point of order r.
OUTPUT: $m \in \mathbb{F}_{q^k}$

1: $T \leftarrow P$, $m \leftarrow 1$
2: **for** $i \leftarrow \lfloor \lg(r) \rfloor - 1$ **to** 0 **do**
3: $\quad m \leftarrow m^2 \cdot l_{T,T}(Q)/v_{2T}(Q)$
4: $\quad T \leftarrow 2T$
5: $\quad$ **if** $r_i = 1$ **then**
6: $\quad\quad m \leftarrow m \cdot l_{T,P}(Q)/v_{T+P}(Q)$
7: $\quad\quad T \leftarrow T + P$
8: $\quad$ **end if**
9: **end for**
10: **return** m

Since the definition of Miller's algorithm and the founding pairings, great efforts have been made and are currently ongoing, to produce efficiently

computable pairings. Some of the main optimisations which have lead to faster pairings include the following. (i) The special choice of the input parameters, in particular choosing the elliptic curve points from specific subgroups to exploit faster finite field arithmetic and avail of the optimisation known as denominator elimination [3]. (ii) The special choice of the order r such that the number of additions performed in the point scalar multiplication are minimised. (iii) The special choice of r such that it is actually a multiple of the group order, thus reducing the complexity of the final exponentiation. As a result of such efforts, a number of efficient variants of the Tate pairing have been developed, for example the η [2], η_T [2], and Ate [9] pairing.

We will describe our fault attacks in the context of the Weil, η and Tate pairing because of the varying complexity of their final exponentiation.

3 The General Idea

Various mechanisms have been researched and found to cause transient faults [1], from the most straightforward attacks which purposely cause variations in the supply voltage or clock signal, to more invasive techniques which target specific areas of the chip with white light and laser beams. Each attack seeks to disrupt the normal execution of an algorithm in a way advantageous to the adversary.

The type of fault that we aim to induce is one which causes the data being operated on to be corrupted in some way. Fault attacks will target memory locations or registers used for computation. Therefore, the way in which data values in the pairing are stored and operated on, will influence the focus of a fault attack.

We assume that the adversary will be able to locate the target point in time in which to inject the fault using Simple Power Analysis (SPA) [13], that is since each round in the Miller loop should be identifiable in the power consumption trace the adversary can identify the target round.

3.1 Overview of the Proposed Attacks

Bilinear pairings consist of two main parts, the Miller loop and the final exponentiation. Therefore, either part of the algorithm can be targeted. To attack the Miller loop, there are a number of locations that a data corruption fault can target. It can affect the Miller variable, the point P (or an intermediate point calculated during the computation of $[r]P$), the point Q (or an intermediate point calculated during computation of $[r]Q$, specifically in the case of the Weil and η pairing), or the order of the Miller loop. An attack that alters the order of the Miller loop was examined by [18]. All other possibilities for attack will be discussed in this paper. To attack the final exponentiation, two data values can be targeted for a data corruption fault, either the Miller variable resulting from the Miller loop or the exponent can be affected. However, the consequences of tampering with either component in the final exponentiation will be difficult to exploit, as will be discussed below.

Let $\rho(P,Q)$ denote any of the candidate bilinear pairing algorithms, i.e. $\rho(P,Q) = \eta(P,Q)$, $\omega(P,Q)$ or $e(P,Q)$ where $\eta(P,Q)$ denotes the η pairing, $\omega(P,Q)$ denotes the Weil pairing and $e(P,Q)$ denotes the Tate pairing. Let $\rho(P,Q)'$ denote a pairing execution in which a fault has been injected. The type of fault that produces $\rho(P,Q)'$ is discussed in the respective attack descriptions.

If we adopt the approach of Page and Vercauteren and consider the Miller loop alone, the pairing $\rho(P,Q)$ can be represented as the product

$$m = \prod_{i=1}^{r} g_i^{2^{r-i}} \tag{2}$$

where g_i accounts for all line function contributions. If a fault is injected into one of the line functions, then the following relationship can be exploited,

$$\frac{\rho(P,Q)}{\rho(P,Q)'} = \frac{m}{m'} = \left(\frac{g_i}{g_i'}\right)^{2^{r-i}} \tag{3}$$

where g_i is the correct line contribution to the i-th iteration, g_i' is the corrupted line contribution to the i-th iteration, i is the round in the Miller loop where the fault was injected and r denotes the number of rounds in the Miller loop. For example, if a correct pairing[1] is calculate as

$$(((((g_1)^2 \cdot g_2)^2 \cdot g_3)^2 \ldots)^2 \cdot g_r) \tag{4}$$

and a faulty pairing is calculated as

$$(((((g_1)^2 \cdot g_2)^2 \cdot g_3')^2 \ldots)^2 \cdot g_r) \tag{5}$$

where the fault corrupts the line contribution in the third round of the algorithm, then dividing a correct pairing by a faulty pairing, where input parameters are identical for both pairings, will nullify any parts of the pairing computation common to both executions, and leave the parts of the pairing computations where the two executions diverge, i.e.

$$\frac{\rho(P,Q)}{\rho(P,Q)'} = \left(\frac{g_3}{g_3'}\right)^{2^{r-3}} \tag{6}$$

the parts of the computation affected by the fault. From equation (6), it can be seen that depending on the round in which the fault is injected, this may be a single contribution to the Miller loop and so a single factor. An obvious objective of the attack will be to inject a fault so that it is as localised as possible, minimising the difference between the valid and faulty pairing. This is achieved by targeting local values and later rounds in the Miller loop, as will

[1] When the pairing involves double and additions, this may also be represented as $(((g_1)^2 \cdot g_2 \cdot a_2)^2 \ldots)^2 \cdot g_r)$ where in some rounds, depending on the binary representation of the loop order, both an addition and double is performed.

be demonstrated below. Generally, once the fault targets the optimal location a single factor from the Miller loop can be accessed. Accessing this single factor can greatly facilitate extraction of the secret, as was the case for [18]. However, in certain bilinear pairing algorithms extracting the secret is not this straightforward, since the output of the Miller loop undergoes another step in the pairing computation, the final exponentiation.

If the bilinear pairing requires a final exponentiation, then relationship in (6) becomes

$$\frac{\rho(P,Q)}{\rho(P,Q)'} = \left(\frac{g_i}{g_i'}\right)^{2^{(r-i)}f} \tag{7}$$

where f is the final exponent. To reclaim the straightforward relationship in (6), reversal of the final exponentiation is necessary. However, depending on f this may not be possible.

The complexity of the final exponentiation depends on the bilinear pairing. In the situation where a simple final exponentiation exists, as is the case for the variant of the Weil pairing considered here [20], access to the output of the Miller loop is possible. This is due to the Frobenius action. For example, let $a + ib$ be an element in the field $\mathbb{F}_{p^2}$, with $a, b \in \mathbb{F}_p$, the Frobenius action is calculated as

$$(a + ib)^{p-1} = \frac{(a - ib)}{(a + ib)} = c + id,$$

thus providing a straightforward relationship between $c+id$ and $a+ib$. This will be demonstrated further in section 4.2.

In the situation where a more complex final exponentiation exists, as in the case for the Tate pairing and most other bilinear pairings, the output of the Miller loop is difficult to access and equivalent to solving a n-th root problem [10]. What makes the final exponentiation difficult to reverse is that it is a one-to-many relationship [8]. In general, the final exponentiation involves raising the output of the Miller loop to the power of $(q^k - 1)/r$. This can be thought of as $(q^d - 1)((q^d + 1)/r)$ or $(q^d - 1)((q^d + 1)/\Phi_k(q))(\Phi_k(q)/r)$ where $\Phi_k(q)$ is the cyclotomic polynomial and $k = 2d$. Raising to the power of $(q^d - 1)$ and $(q^d + 1/\Phi_k(q))$ is easy using the Frobenius action. Raising to the power of $(\Phi_k(q)/r)$ is not as straightforward, requiring usage of an algorithm for fast exponentiation [17,11,12].

Page and Vercauteren describe an attack of the Duursma-Lee algorithm for the Tate pairing, which has a final exponent of $q^3 - 1$. In their attack they are able to reverse this final exponentiation and access the single factor from the Miller loop. This is on account of the fact that the single factor that they are accessing has a special form and so is identifiable from all other roots. In the next section, it will be shown that the successful attacks on the η and Weil pairing, do not work on the Tate pairing. This is because the factor that the Tate pairing produces from the Miller loop is not of special form and so cannot be pinpointed from all other roots.

As mentioned above, the final exponentiation can itself also be the target of a fault attack. However, considering even the most powerful scenario, where the fault nullifies the final exponentiation, this mean that the output of the bilinear pairing algorithm will be the output of the Miller loop, a system of multivariate equations similar to those described in [8], which is difficult to solve. Therefore, corrupting the final exponentiation, will not aid the adversary in finding the secret. If the adversary can inject multiple faults such that the final exponentiation is nullified in one execution, and in another execution the final exponentiation is nullified and a fault is injected into the Miller loop to isolate a single factor, then an attack could be launched. This however, is an unrealistic attack scenario.

4 Specific Examples of Attack

In this section, concrete instances on a data corruption fault attack on the η and Weil pairing are presented. It is then shown how such attacks are not applicable to the Tate pairing.

4.1 Corrupting the η Pairing

The η pairing $\eta(P, Q)$, specialises in pairings over supersingular curves of small characteristic. The main distinction between the η pairing and its siblings is that it chooses the order of the Miller loop r as a multiple of the group order such that it divides $q^k - 1$ nicely to give a small factor, resulting in a simple final exponentiation. For example, consider the η pairing on a supersingular curve of characteristic two, if $q^k - 1 = 2^{4m} - 1$ and $r = 2^{2m} + 1$, which is a multiple of the order $2^m \pm 2^{(m+1)/2} + 1$, then the final exponentiation basically involves a conjugation and division, i.e. $(2^{2m} - 1)$.

Recently Galbraith et al. [8] described a variant of the η pairing which required no final exponentiation, i.e. the result of the pairing is a unique element and a bilinear map without exponentiation. This is enabled by the additional evaluation of vertical line functions (which the original η pairing [2] does not have). This version of the η pairing considered here will be referred to as the η_G pairing to distinguish it from the original η pairing. Algorithms 2 and 3 describe the η_G pairing.

When [8] presented the η_G pairing with no final exponentiation, they addressed possible security implications. Mathematical attacks such as a multivariate attack and a straight line program (SLP) were considered. However, they conclude that the pairing's security (in the non fault attack sense) is still strong, and breaking such a pairing requires solving the pairing inversion problem, which both [23] and [19] show is difficult. In this section, it is shown how pairings with no final exponentiation can succumb to a data corruption fault attack.

The implementation assessed is over the quartic extension field, i.e. $k = 4$. Field elements in $\mathbb{F}_{2^{4m}}$ will be represented as four elements $a_i \in \mathbb{F}_{2^m}$ and so stored in four different memory locations. We define notation $[a_0][a_1][a_2][a_3]$ to represent the storage of each of these elements. Each component of this representation is

Algorithm 2. Adds elliptic curve points and calculated the most recent contribution to the Miller variable g

INPUT: $A = (x_A, y_A), C = (x_C, y_C), Q = (x_Q, y_Q)$
OUTPUT: $g \in \mathbb{F}_{2^{4m}}$ and updates A
1: $(A', \lambda_1) \leftarrow A + C$
2: $l \leftarrow [y_Q + y_1 + \lambda_1(x_Q + x_1 + 1)][\lambda_1 + x_Q + 1][\lambda_1 + x_Q][0]$
3: $v \leftarrow [x_Q + x_A + 1][1][1][0]$
4: $A \leftarrow A'$
5: **return** $g \leftarrow \frac{l}{v}$

Algorithm 3. Computation of $\eta_G(P, Q)$ on $E(\mathbb{F}_p) : y^2 + y = x^3 + x + b$, where P is a point of prime order r on $E(\mathbb{F}_{2^m})$ and Q is a point on the $E(\mathbb{F}_{2^m})$

INPUT: $P = (x_P, y_P), Q = (x_Q, y_Q)$
OUTPUT: $m \in \mathbb{F}_{2^{4m}}$
1: $m \leftarrow 1$
2: $A \leftarrow P$
3: $r \leftarrow m$
4: **for** r **to** 0 **do**
5: $\quad m \leftarrow m^2 * g(A, A, Q)$
6: **end for**
7: **return** m

referred to as a cell. The cells of l and v will be denoted by $[l_0]$, $[l_1]$, $[l_2]$, $[l_3]$ and $[v_0]$, $[v_1]$, $[v_2]$, $[v_3]$ respectively.

There are a number of possible locations in which the fault can be injected, and a number of different effects that this fault can have. A fault can be injected into any of the cells of l or v, or any of the coordinates x_A, y_A, x_C, x_Q or y_Q and the fault injected can corrupt a bit or byte or multiple bits or bytes of the target. If a fault is injected into any of the cells of l or v, then the effect will be local, only corrupting the cell in question. If the fault is injected into one of the coordinates, then the resulting erroneous coordinate will have consequences for all locations and all subsequent operations in which that coordinate is used. The effects of the faults in these locations, will be each addressed in turn.

Let $\eta_G(P, Q)'$ denote a corrupted pairing, where the fault is injected into the cell l_0 in the last round (round r)of the Miller loop. Division of the faulty pairing by the valid pairing will isolate the round in which the fault was injected to yield

$$\frac{\eta_G(P, Q)'}{\eta_G(P, Q)} = \frac{g'_r}{g_r} = \frac{(l'/v)}{(l/v)} = \frac{l'}{l} = \frac{[l_0]'[l_1][l_2][l_3]}{[l_0][l_1][l_2][l_3]}.$$

This division will produce an element in $\mathbb{F}_{2^{4m}}$, and can be thought of as four different cell values N_0, N_1, N_2 and N_3, where $N_i \in \mathbb{F}_{2^m}$. Therefore,

$$\frac{[l_0]'[l_1][l_2][l_3]}{[l_0][l_1][l_2][l_3]} = [N_0][N_1][N_2][N_3]. \tag{8}$$

Given $\eta_G(P, Q)$ and $\eta_G(P, Q)'$, the adversary can compute N_0, N_1, N_2 and N_3.

Using knowledge of how multiplication in $\mathbb{F}_{2^{4m}}$ is performed, the following three equations can be derived.

$$l'_0 = N_0 l_0 + N_1 l_1 + N_2 l_2 + (N_1 + N_3) l_1 \tag{9}$$

$$l_1 = N_0 l_0 + 2N_1 l_1 + 2N_2 l_2 + (N_0 + N_1)(l_0 + l_1) + (N_1 + N_3) l_1 + (N_2 + N_3) l_2 \tag{10}$$

$$l_2 = N_0 l_0 + N_1 l_1 + 2N_2 l_2 + (N_2 + N_3) l_2 + (N_0 + N_2)(l_0 + l_2) \tag{11}$$

The adversary will obviously choose the optimal equation to solve. Equation (9), contains l'_0, whereas equations (10) and (11) do not. Hence either equations (10) or (11) contain less unknown information. In the scenario where P is private, (11) can be simplified to

$$A_0 x_A + B_0 y_A + C_0 \lambda + D_0 = 0 \tag{12}$$

where

$$\begin{aligned} A_0 &= N_2 \lambda \\ B_0 &= N_2 \\ C_0 &= N_0 + N_1 + N_2 + N_3 + N_2 x_Q + 1 \\ D_0 &= (N_0 + N_1 + N_3 + 1) x_Q + N_2 y_Q + N_1 \end{aligned}$$

In the scenario where Q is private, (11) can be simplified to

$$A_0 x_Q + B_0 y_Q + C_0 = 0 \tag{13}$$

where

$$\begin{aligned} A_0 &= N_0 + N_1 + N_2 \lambda + N_3 + 1 \\ B_0 &= N_2 \\ C_0 &= N_0 \lambda + N_1 + N_1 \lambda + N_2 (x_A \lambda + y_A + \lambda) + N_3 \lambda + \lambda \end{aligned}$$

Note that (10) could just as validly have been used. When P is secret, notice that equation (12) is in fact a non-linear equation. However, when Q is secret a linear equation is obtained. To solve for Q, equation (13) has two unknown variables and so two equations are required to solve the system. First however, the required equations must be produced. The main requirement is that the equations are produced with the same unknown variables. This can be performed by repeatedly executing the pairing with the objective of injecting various faults into l_0. For example, l''_0/l_0 will produce N_4, N_5, N_6, and N_7, and so an entirely different set of equations in the same variable. Similarly, the pairing could be executed with other parameters, $\eta_G(P_i, Q)$ for various elliptic curve points P_i, where the secret Q remains static. Given two equations, modular Gaussian elimination [15] or simply substitution can be used. Alternatively, since x and y can be expressed in terms of each other using the elliptic curve equation, one fault alone will be sufficient to extract Q. A specific example of this will be given in the next section.

If a fault is injected into any of the other cells of l, then a similar process can be applied to derive similar equations. Hence, fault injection into any of the cells of l, is equivalent to solving a system of linear equations mod p.

In the scenario where the fault is injected into v, the η_G pairing succumbs to a more straightforward attack. Let $\eta_G(P,Q)'$ denote a corrupted pairing, where the fault is injected into the cell v_0 in the last round of the Miller loop. Division of the valid pairing by the faulty pairing will isolate the round in which the fault was injected to yield

$$\frac{\eta_G(P,Q)}{\eta_G(P,Q)'} = \frac{g_r}{g_r'} = \frac{(l/v)}{(l/v')} = \frac{v'}{v} = \frac{[v_0]'[v_1][v_2][v_3]}{[v_0][v_1][v_2][v_3]}.$$

This division will again produce elements N_0, N_1, N_2, N_3. Equations similar to (9), (10) and (11) can be generated, in which v_0, v_0', v_1 and v_2 replace l_0, l_0', l_1 and l_2. Since $v_1 = 1$ and $v_2 = 1$, only one unknown piece of data remains,

$$v_0 = \frac{N_0 + 4N_1 + 3N_2 + 2N_3 + 1}{2N_0 + N_1} \tag{14}$$

or

$$v_0 = \frac{N_0 + N_1 + 4N_2 + N_3 + 1}{2N_0 + N_2}, \tag{15}$$

allowing the coordinate x_Q to be retrieved when Q is secret and x_C to be extracted when P is secret. A numerical example of this attack is given in appendix A.

If the target for the fault attack is instead a coordinate used in the calculation of the line function, various consequences can be witnessed. The main difference between corruption of a coordinate and corruption of a cell as was described above, is that a coordinate may be influential in a number of cells. The coordinate x_C is only resident in the cell v_0 and so if corrupted will have similar consequences to the corruption of v_0 as was described above. Similarly, if the coordinates x_A, y_A or y_Q are corrupted, then this is equivalent to the corruption of the cell l_0.

If however, the fault attack corrupts either the coordinate x_Q or the slope λ, then the fault will affect numerous cells. If the fault affects λ, the cells l_0, l_1 and l_2 will also be affected, and result in introducing an extra unknown variable into the system of linear equations. This results in an increase in the number of unknowns to four variables for P secret and three for Q secret. In addition, this extra variable is very specific in that the solution of the system of equations requires the ability to recreate identical faults. For example, the same λ' must be created in at least three pairings in order to be able to extract Q. This is a difficult task to initiate and detect. If the fault affects x_Q, then the cells l_0, l_1, l_2 and v_0 will be corrupted. Therefore, the division of the valid and faulty pairing will not cancel one of the line functions, l or v, leaving a relationship of the form

$$\frac{\eta_G(P,Q)}{\eta_G(P,Q)'} = \frac{(l/v)}{(l'/v')} = \frac{\dfrac{[l_0][l_1][l_2][l_3]}{[v_0][v_1][v_2][v_3]}}{\dfrac{[l_0]'[l_1]'[l_2]'[l_3]}{[v_0]'[v_1][v_2][v_3]}} = [N_0][N_1][N_2][N_3].$$

When expanded, a difficult modular non-linear equation is obtained.

Another consequence of corrupting coordinates, is that the effect of the fault will not be local, as is the case for cell corruption. This is particularly problematic if the fault is injected in a round prior to the final round. For example, if the coordinates x_A or y_A are corrupted in a round preceding the final round, then the subsequent point addition in which x_A and y_A will be involved, will be affected. A common practice to optimise computation time is to pre-compute the intermediate points required in the computation of $[r]P$ and store them on the device, so only a look up operation is required as opposed to elliptic curve point scalar multiplication [22]. In such implementations, the fault can target the memory cell in which x_A or y_A are stored, and so the effect will remain local. In scenarios where these types of attacks are a threat, this would be a valid argument against pre-calculation.

4.2 Corrupting the Weil Pairing

The Weil pairing $\omega(P, Q)$ over a prime characteristic elliptic curve with embedding degree $k = 2$, was originally defined with no final exponentiation. However by eliminating the vertical function evaluations (an optimisation known as denominator elimination), a Weil pairing with a simple final exponent can be more efficient [20]. To distinguish the version of the Weil pairing considered here from the original Weil pairing, it will be denoted by ω_D.

The particular implementation assessed [20] considers ordinary elliptic curves over the prime field $\mathbb{F}_p$ with $k = 2$ and a final exponent of $p-1$. Algorithm 4 and 4.2 describe the variant of the Weil algorithm considered here. The output of the pairing is an element in $\mathbb{F}_{p^2}$. The cells of u and v will be denoted by $[u_0][u_1]$ and $[v_0][v_1]$ respectively.

Again, a number of different locations can be targeted and various types of faults can be injected to cause different effects. The types of faults which aid in the extraction of the secret are far fewer than in the η_G pairing. This is because direct access to the output of the Miller loop is not available.

Let $\omega_D(P_i, Q_i)'$ denote the Weil pairing in which a fault is injected into the last round in the Miller loop. The fault corrupts data in an unspecified way, i.e. either $[u_0]$, $[u_1]$, $[v_0]$ or $[v_1]$ are corrupted. The case where $[u_0]$ is corrupted, will be described for demonstrative purposes, however the same end result is witnessed for this type of fault regardless of the cell targeted. Division of the valid pairing by the fault pairing will yield

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{g_r^{p-1}}{g_r'^{p-1}} = \frac{(u \cdot v)^{p-1}}{(u' \cdot v)^{p-1}} = \frac{([u_0][u_1])^{p-1}}{([u_0]'[u_1])^{p-1}}.$$

This type of disruption of the execution has the following consequences, where raising an element in $\mathbb{F}_{p^2}$ to the power of $p-1$ involves a conjugation and division.

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{\dfrac{([\lambda_i(x_A + x_Q) - y_A][-y_Q])}{([\lambda_i(x_A + x_Q) - y_A][y_Q])}}{\dfrac{([\lambda_i(x_A + x_Q) - y_A]'[-y_Q])}{([\lambda_i(x_A + x_Q) - y_A]'[y_Q])}}$$

Algorithm 4. Adds elliptic curve points and calculated the most recent contribution to the Miller variable g

INPUT: $A = (x_A, y_A), B = (x_B, y_B), C = (x_C, y_C), D = (x_D, y_D), P = (x_P, y_P), Q = (x_Q, y_Q)$
OUTPUT: $g \in \mathbb{F}_{p^2}$ and updates A and C
1: $(A', \lambda_1) \leftarrow A + B$
2: $(C', \lambda_2) \leftarrow C + D$
3: $u \leftarrow \lambda_1(x_A + x_Q) - y_A + y_Q i$
4: $v \leftarrow y_P + (y_C - \lambda_2(x_C + x_P))i$
5: $A \leftarrow A'$, $C \leftarrow C'$
6: **return** $u * v$

Algorithm 5. Computation of $\omega_D(P, Q)$ on $E(\mathbb{F}_p) : y^2 = x^3 + ax + b$, where P is a point of prime order r on $E(\mathbb{F}_p)$ and Q is a point on the twisted curve $E'(\mathbb{F}_p)$

INPUT: $P = (x_P, y_P), Q = (x_Q, y_Q)$
OUTPUT: $m \in \mathbb{F}_{p^2}$
1: $m \leftarrow 1$
2: $A \leftarrow P$
3: $C \leftarrow Q$
4: $n \leftarrow r - 1$
5: **for** $i \leftarrow \lfloor \lg(r) \rfloor - 2$ **to** 0 **do**
6: $\quad m \leftarrow m^2 * g(A, A, C, C, P, Q)$
7: $\quad$ **if** $n_i = 1$ **then**
8: $\quad\quad m \leftarrow m * g(A, P, C, Q, P, Q)$
9: $\quad$ **end if**
10: **end for**
11: **return** $m \leftarrow \frac{\bar{m}}{m}$

This is equivalent to

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \left(\frac{([\lambda_i(x_A + x_Q) - y_A][-y_Q])}{([\lambda_i(x_A + x_Q) - y_A][y_Q])} \cdot \frac{([\lambda_i(x_A + x_Q) - y_A]'[y_Q])}{([\lambda_i(x_A + x_Q) - y_A]'[-y_Q])} \right).$$

If this equation is expanded, a difficult multivariate non-linear equation is derived. The cancelations that were possible on the η_G pairing, which allow access to a simple factor, are no longer possible. This is due to the final exponentiation.

Since a general data corruption fault of the Weil pairing will not suffice to extract the secret, other fault types are examined. Another more powerful and targeted attack that will facilitate extraction of the secret elliptic curve point, targets the sign of either $[u_1]$ or $[v_0]$ and so the coordinate y_Q or y_P. This type of fault attack is referred to as a sign change fault attack since a single sign bit is flipped [4].

Again it is assumed that one valid pairing $\omega_D(P, Q)$ and one faulty pairing $\omega_D(P, Q)'$, can be calculated. Let the fault attack cause a change in sign in the

y_Q component of the elliptic curve point Q during the last round of the Miller loop, i.e. u is altered as follows

$$u' = [\lambda_i(x_A + x_Q) - y_A][-y_Q].$$

Since the implementation of the Weil pairing being assessed requires a final exponentiation, division of the valid pairing by a faulty pairing will yield the following:

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{g_r^{p-1}}{g_r'^{p-1}} = \frac{(u \cdot v)^{p-1}}{(u' \cdot v)^{p-1}} = \frac{(u)^{p-1}}{(u')^{p-1}} = \frac{([\lambda_i(x_A + x_Q) - y_A][y_Q])^{p-1}}{([\lambda_i(x_A + x_Q) - y_A][-y_Q])^{p-1}}$$

Raising an element in $\mathbb{F}_{p^2}$ to the power of $p-1$ is simply a conjugation and division,

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{\frac{([\lambda_i(x_A + x_Q) - y_A][-y_Q])}{([\lambda_i(x_A + x_Q) - y_A][y_Q])}}{\frac{([\lambda_i(x_A + x_Q) - y_A][y_Q])}{([\lambda_i(x_A + x_Q) - y_A][-y_Q])}}$$

which is equivalent to

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \left(\frac{[\lambda_i(x_A + x_Q) - y_A][-y_Q]}{[\lambda_i(x_A + x_Q) - y_A][y_Q]}\right)^2.$$

Therefore,

$$\sqrt{\frac{\omega_D(P,Q)}{\omega_D(P,Q)'}} = \pm\left(\frac{[\lambda_i(x_A + x_Q) - y_A][-y_Q]}{[\lambda_i(x_A + x_Q) - y_A][y_Q]}\right) \tag{16}$$

If we let $(N_R,N_C) \in \mathbb{F}_{p^2}$ equal the resulting value from this operation, two linear equations can be derived,

$$(\lambda_i - N_R\lambda_i)x_Q + N_C y_Q + (x_A\lambda_i - y_A - N_R x_A\lambda_i + N_R y_A) = 0 \tag{17}$$

$$(N_C x_A\lambda_i)x_Q + (N_R + 1)y_Q + (N_C x_A\lambda + i - N_C y_A) = 0 \tag{18}$$

Since there exists two possible square roots, there are two possibilities for each equation. Depending on whether P or Q is the secret will affect how the secret is extracted. The most straightforward extraction in this scenario is where Q is the secret. Using the elliptic curve equation E, $\pm\sqrt{x_Q^3 + ax_Q - b}$ can be substituted for y_Q in either of the above equations. This will yield a cubic equation in one variable, which is solvable by Cardano's method [6]. For example (17) reduces to

$$x_Q^3 + A_0x_Q^2 + B_0x_Q + C_0 = 0 \tag{19}$$

where

$$A_0 = -\frac{(\lambda_i - N_R\lambda_i)^2}{N_C^2}$$
$$B_0 = \frac{(a - 2(\lambda_i - N_R\lambda_i)(x_A\lambda_i - y_A - N_R x\lambda_i + N_R y_A))}{N_C^2}$$
$$C_0 = -b - \frac{(x_A\lambda_i - y_A - N_R x_A\lambda_i + N_R y_A)^2}{N_C^2}$$

and (18) reduces to

$$x_Q^3 + A_0 x_Q^2 + B_0 x_Q + C_0 = 0 \tag{20}$$

where

$$A_0 = -\frac{(N_C x_A\lambda_i)^2}{(N_R+1)^2}$$
$$B_0 = \frac{(a + 2(N_C x_A\lambda_i)(N_C x_A\lambda_i - N_C y_A))}{(N_R+1)^2}$$
$$C_0 = -b + \frac{(N_C x_A\lambda_i)^2}{(N_R+1)^2}$$

Given x_Q, y_Q can then be found by simply calculating $\pm\sqrt{x_Q^3 + ax_Q - b}$. A numerical example of this attack is given in appendix B. This type of attack is also possible if the sign of y_P is similarly targeted in the line v, in which case the scenario where P is secret will present the most straightforward avenue for extraction of the secret.

As previously explained, the optimal time in which to inject the fault is the final round of the Miller loop. If the fault is injected in earlier rounds in the Miller loop, then the effort to extract the secret is increased. Any round preceding the final round sees the Miller variable m being squared. Therefore, to access equations similar to (17) or (18), multiple square roots calculations will be required. For example, let the fault corrupt the sign of y_Q in the second last round of the Miller loop. The relationship between the output of the pairings and the data of interest is now

$$\frac{\omega_D(P,Q)}{\omega_D(P,Q)'} = \frac{((m^{2^{r-1}} \cdot g)^2)^{p-1}}{((m^{2^{r-1}} \cdot g')^2)^{p-1}} = \left(\frac{(u)^2}{(u')^2}\right)^{p-1}$$

and so

$$\sqrt{\sqrt{\frac{\omega_D(P,Q)}{\omega_D(P,Q)'}}} = \pm\left(\frac{[\lambda_i(x_A + x_Q) - y_A][-y_Q]}{[\lambda_i(x_A + x_Q) - y_A][y_Q]}\right).$$

This requires the computation of two square roots and so potentially four cubic equations (depending on whether the square root exists or not). Hence,

the earlier in the loop the fault is injected the greater number of cubic equations there will be to solve and test. It is expected that this problem can be combatted by using SPA to identify the target loop.

4.3 The Resistance of Tate Pairing to a Data Corruption Fault

The Tate pairing $e(P,Q)$, has the most complex final exponent of $(q^k-1)/r$. This not only ensures that the output of the pairing is unique, but it also serves as a defensive mechanism against fault attacks as we will now demonstrate.

We assessed the BKLS algorithm for the Tate pairing [21] over the large prime field $p=q$ with embedding degree $k=2$. Again, the heart of the Miller loop involves the calculation of the Miller variable m, which consists of the calculation of g. In each round, g is calculated by the evaluation of line function u, which is calculated as

$$u = [y_A - \lambda(x_Q + x_A)][-y_Q].$$

As before, in each round g is multiplicatively incorporated into the Miller variable. Elements in the extension field and consequently the output of the pairing is an element in $\mathbb{F}_{p^2}$.

The sign change fault attack of the Weil pairing, will be used to examine the strength of the Tate pairing. This is for two main reasons. Firstly, the Tate pairing is most similar to the Weil pairing. Secondly the sign change fault attack is the most powerful attack and so if Tate withstands this attack, it is inferred that it will withstand less powerful attacks such as the general data corruption fault.

Once again it is assumed that a fault has been injected into the final round of the Miller loop. The contribution of the line function for that round can thus be isolated as

$$\frac{e(P,Q)}{e(P,Q)'} = \frac{g_r^{\frac{p^2-1}{r}}}{g_r'^{\frac{p^2-1}{r}}} = \frac{(u)^{\frac{p^2-1}{r}}}{(u')^{\frac{p^2-1}{r}}} = \left(\frac{[y_A - \lambda(x_Q + x_A)][-y_Q]}{[y_A - \lambda(x_Q + x_A)][y_Q]}\right)^{\frac{p^2-1}{r}}$$

Breaking the exponent into its factors as was demonstrated in equation (16), this can be simplified to

$$\sqrt{\frac{e(P,Q)}{e(P,Q)'}} = \pm\left(\frac{[y_A - \lambda(x_Q + x_A)][y_Q]}{[y_A - \lambda(x_Q + x_A)][-y_Q]}\right)^{\frac{p+1}{r}}. \tag{21}$$

However, the reversal of the exponent $(p+1)/r$ is infeasible. To access

$$\frac{[y_A - \lambda(x_Q + x_A)][y_Q]}{[y_A - \lambda(x_Q + x_A)][-y_Q]}, \tag{22}$$

and subsequently derive the secret elliptic curve point (whether it be P or Q), a specific n-th root, where $n = (p+1)/r$, must be calculated. Since n divides the group order once, there exists a simple formulae to compute a n-th root, i.e.

$$\left(\sqrt{\frac{e(P,Q)}{e(P,Q)'}}\right)^{n^{-1} \pmod{s}} \tag{23}$$

where $s = (p+1)/n$. However, The particular root of interest does not exhibit any special form, unlike the case of [18], and is a full quadratic element. Therefore, no extra information is available to aid in determining which is the root we are interested in. In addition, since there exists n n-th roots, where n is large, the cost of computing and testing all n-th roots, renders such an attack infeasible.

5 Countermeasures

Apart from choosing a pairing algorithm which has a complex final exponentiation, a number of fault obfuscation and detection mechanisms can be used to prevent the described attacks. In this section, various techniques will be presented.

5.1 Fault Obfuscation Mechanisms

The aim is to implement methods which given the data corruption faults described above, disables the adversary from extracting the secret. Page and Vercauteren [18] describe a number of techniques that blind the input point known to the adversary. For example, by computing

$$\rho(sP, Q)^{s^{-1}},$$

where P is public and Q is the secret, the values required for use in equations (9), (10) and (11) and (17) or (18), i.e x_P, y_P, x_A and y_A, will no longer be computable by an adversary.

5.2 Fault Detection Mechanisms

Numerous software and hardware mechanisms already exist to detect a fault attack that are not algorithm specific [1]. For example, the simple act of executing the algorithm twice to check if the results are the same can be applied to any algorithm. The property of bilinearity inherent in pairings allows a method of fault detection, which is specific to pairings. By checking whether

$$\rho(P, Q)^{sr} = \rho(sP, rQ), \tag{24}$$

any faults injected will be caught. This can be applied to all pairings and should pick up any type of fault. The only drawback of this fault detection mechanism is that it is quite costly, requiring two pairing computations and additional

point scalar multiplication of two points and exponentiation of an element in the extension field $\mathbb{F}_{q^k}$.

Other more cost effective methods of fault detection could involve random checking of whether the intermediate points used in the computation are still points on the curve, i.e check if $x_i, y_i \in y^2 = x^3 + ax + b$ or $y^2 + y = x^3 + x + b$. A check could be carried out in every round but this will require $2r$ evaluations which could be expensive. There is also a possibility that this check will not be reliable. For example, the sign change fault attack of the Weil pairing returns a point that is still on the curve and so will pass this check.

5.3 Hiding the n-th Root

In the event of the exact n-th root being identified, to combat a sign change fault attack, it is possible that the elliptic curve parameters might be set up such that n^2 divides the group order once (i.e. p and r are specially chosen to meet this requirement). Now the problem of finding just one n-th root, i.e solving (23), is equivalent to solving a Discrete Log problem [10].

6 Conclusion

We have investigated the effect of data corruption fault attacks on three categories of pairing implementations. We have shown that the success of these attacks depends on the difficulty of reversing the final exponentiation.

It is the nature of the final exponentiation to produce a unique value. However, in doing so it also destroys information, making it difficult to recover the value which was exponentiated.

References

1. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The Sorcerers Apprentice Guide to Fault Attacks. In: Proceedings of the IEEE, Special Issue on Cryptography and Security, vol. 96(2), pp. 370–382 (2006)
2. Barreto, P., Galbraith, S., O'hEigeartaigh, C., Scott, M.: Efficient Pairing Computation on Supersingular Abelian Varieties. Cryptology ePrint Archive: Report, 2004/375. URL: http://eprint.iacr.org/2004/375
3. Barreto, P., Kim, H., 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. Blomer, J., Otto, M., Seifert, J.-P.: Sign Change Fault Attacks on Elliptic Curve Cryptosystems. In: Workshop on Fault Detection and Tolerance in Cryptography - FDTC 2005. LNCS, Springer, Heidelberg (2005)
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. Cardano, G.: Cardano's Formula for Solving a Cubic Equation. Wikipedia, URL: http://en.wikipedia.org/wiki/CubicEquation
7. Duursma, I.M., Lee, H.S.: Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
8. Galbraith, S., OhEigeartaigh, C., Sheedy, C.: Simplified Pairing Computation and Security Implication. Cryptology ePrint Archive: Report (2006) URL: http://eprint.iacr.org/2006/
9. Hess, F., Smart, N., Vercauteren, F.: The Eta Pairing Revisited. Cryptology ePrint Archive: Report, 2006/110. URL: http://eprint.iacr.org/2006/110
10. Johnston, A.: On the Difficulty of Prime Root Computation in Certain Finite Cyclic Groups. PhD thesis, Royal Holloway University of London (2006) URL: http://www.ma.rhul.ac.uk/techreports/
11. Joye, M., Quisquater, J.J.: Efficient Computation of Full Lucas Sequences. Electronic Letters 32(6), 537–538 (1996)
12. Joye, M., Yen, S.: The Montgomery Powering Ladder. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291–302. Springer, Heidelberg (2002)
13. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
14. Kwon, S.: Efficient Tate Pairing Computation for Supersingular Elliptic Curves over Binary Fields. Cryptology ePrint Archive: Report 2004/303. URL: http://eprint.iacr.org/2004/303
15. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press (1996)
16. Miller, V.: The Weil Pairing, and its Efficient Calculation. Journal of Cryptology 17, 235–261 (2004)
17. Montgomery, P.L.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Mathematics of Computation 48(177), 243–264 (1987)
18. Page, D., Vercauteren, F.: Fault and Side-Channel Attacks on Pairing Based Cryptography. In: Fault Detection and Tolerance in Cryptography - FDTC 2005 (2005)
19. Satoh, T.: On Polynomial Interpolations of Homomorphisms from Finite Fields to Elliptic Curves. In: LMS JCM
20. Scott, M.: Multiprecision Integer and Rational Arithmetic C/C++ Library - MIRACL. URL: http://www.shamus.ie
21. Scott, M.: Computing the Tate Pairing. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
22. Scott, M., Costigan, N., Abdulwahab, W.: Implementing Cryptographic Pairings on Smart Cards. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 134–147. Springer, Heidelberg (2006)
23. Verheul, E.: Evidence that XTR is more Secure than Supersingular Elliptic Curve Cryptosystems. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 195–210. Springer, Heidelberg (2001)

A Numerical Example of a Data Corruption Fault of the η Pairing: Corrupting the Cell v_0

Note, in each of the examples described in this appendix it is assumed that the secret is the elliptic curve point Q.

A.1 Data Values

Field parameters
Reduction polynomial: $x^{271} + x^{58} + 1$
Elliptic curve: $y^2 + y = x^3 + x$

Input Points
P: (69E714DA0E9BF21C4E9D400F4ED644F8C80E983D60026CD2AEC39AB010CEE129B34D,
50882E348FD0133A0120F005BF65CB8C21A8837DAD9E5BD58FA9CF1BBFC24AEDD0E4)
Q: (7B92FE9FBAB4ED6F8C8902D159020B9BA8C7C01A9DF27FC05FA9036C9727AB203CE9,
7C71F818ABD7D6F408AB5935D6A114476B837BE1ADA14C49DD13C071D1E044C15F44)

Data used in round of Miller loop targeted by the fault
x_A: 38F3F251F96093C28B2B82D9EB12D93BD4E5D0C123D851E1AC5A3BF6BD998DD40BA
y_A: 3EF89128D586F7F0082C10BD03D081549F29AC42A51C67B5D628271359CD30594C85
x_C: 69E714DA0E9BF21C4E9D400F4ED644F8C80E983D60026CD2AEC39AB010CEE129B34C
y_C: 396F3AEE814BE1264FBDB00AF1B38F74E9A61B40CD9C3707216A55ABAF0CABC463A8
λ: 18543BF3AAF8010C87D9C01123FC0BC6D29B6C25C6A783AEAC8247420DF6C25F9FA6

Output from two (valid, faulty) pairing executions
Output from Valid Execution $\eta(P, Q)$:
[25113BDCDD9200B876451861074C193BB570A864612A13079C93D3D8F1D94166A70C,
5339132DF97AEEFE6401F2C56B28DB23D467501C4EA6FB1DBDE47E63181B4FAB1043,
54689B2C103FD9AD37E24464D37C43D4C869F7562D064195D3DEED7258C9DE8547D2,
20BE058D44CFFA54C008EDED71D776D2AF63EB1A7CAA41B0C6B101E4DE444DA4E01F]
Output from Faulty Execution $\eta(P, Q)'$:
[4F38B1694502C41ADE928596BCA3CB37A65BEDAE81B3F3195986732B16EFF50F39BC,
1B72EBFAB9953ACFA9D15025F7C8A6CDFC0122F5B650B4F3555AC7747EB25B4DE344,
726CD21B5F6A060EBDB9DB744D603B1EB57C6E6A0EE95760DF4CBB863ACD238C1882,
62FCBFD09FC2700B456C2EFC1E8F24389C528C43511A7E9CE4831CE0B155DC73DF9F]

A.2 Steps to the Attack

Step 1: Divide the valid pairing by the faulty pairing.
[53DD742B0B855A7233697CEB0A0E8C2C1433834EA6608F4F67FB26BEFB4EAB27BC98,
47C7B8A0729BA78C76FA41B95198B8BB24404B0CB14D17EFBA54C18189167FAFB099,
47C7B8A0729BA78C76FA41B95198B8BB24404B0CB14D17EFBA54C18189167FAFB099,0]

Step 2: Solve the congruence,

$$v_0 = \frac{N_0 + 4N_1 + 3N_2 + 2N_3 + 1}{2N_0 + N_1}$$

Since we are dealing with binary fields, any even multiples of elements in $\mathbb{F}_{2^m}$ will dissolve to zero. Therefore,

$$v_0 = \frac{N_0 + N_2 + 1}{N_1}$$

which evaluates as
v_0 = 1275EA45B42F1F73C21442DE17D44F6360C95827FDF01312F16A99DC87E94A098FA4

By simply adding/subtracting the known value $x_C + 1$, the secret coordinate x_Q is found,
$v_0 + x_C + 1 = x_Q =$
7B92FE9FBAB4ED6F8C8902D159020B9BA8C7C01A9DF27FC05FA9036C9727AB203CE9

B Numerical Example of a Sign Change Fault of Weil

B.1 Data Values

Field parameters
p: C8DF90FD89D7A9827ED3A42409D36AED13BA334D9E716169FC674B6CB2F98E407A50
E9E0FBDC15512D1F74A679C08D3AC42F292C8A95DF30758293235E04739B
a: C8DF90FD89D7A9827ED3A42409D36AED13BA334D9E716169FC674B6CB2F98E407A50
E9E0FBDC15512D1F74A679C08D3AC42F292C8A95DF30758293235E047398
b: 491224746BC3C29BAB0157FE84580BB3E53912605012B88BCCEFE240FB679E4605EA
BBBBA9A642BEA71A3840107915913F998E71662C2501A2E35122C569BF68
r: 8000000000000000000000000000000000020001
Elliptic curve: $y^2 = x^3 + ax + b$

Input Points
P:
(C27938C41E38949CA9177135C99374D27659D9489955F65F312EEF10375C78CAE23
48E0E8E7B32D5021E181CABC42EFE30D44038950F23E7F49980D2654C09B8,59F5C3
910EB567CDB4B16BB2A0B2A8DFEA3ED92A64A4524789057656 8B779A846ECBC9FEA1
50CD492223836EE4796B23751D547B4DC06CD48E230DCAE43632DA)
Q:
(B1F0D753A62BC0458E230371B2E663321CEDC5DC0BBBFA88B16BB4A549310D7297A
E3F90F87876C63D898891085FDB29C61005E94C85DB372879FA953420F48B,A747D6
4825676DB1176A4A2CF79EAD209525FBFFE008749CEB0458C8433501979524635D34
36CE5D232FF1CF122F01B659E36CA0EFB1AED3318EE4B32FD8423C)

Data used in round of Miller loop targeted by the fault
x_A: 3A385AA26F93FC9279B2A69C326E3845C91302AD818122EA21480D53034A7A8
1A60C46778707619423A3D0299A18F29C35692AC0A4CE1923235FD5A0DB1E0584

y_A: 71BD2AB938A12B43B67C6A39C9D0134558B6E7CBA14A0328C0C238D3172F6EC
5C36E53E79EA5EF348C67DE4452AF353F9DCE41DE2CC362414DEA4E6621100602
λ_i: A4D9B86BCB52F6896909EF178128D2EF55AFA2DCDD9035C90ABD58C0A88AC3A
0E16A910DA759664F9BBE7BCF396842A134ECA15D5390EFFE2B9BD52D29AAF28C)

Output from two (valid, faulty) pairing executions
Output from Valid Execution $\omega(P, Q)$:
[66192B7F5D59DF1CB6238052468D262D3EEE879E056FD7A7B52B4181C58AFAD1486
AC1DAEA498CC73943159C0F161CD3426DBE36FDA84D0DD520BF237CBD326E,9804ED
2E2F5FEF2CD13165A2FAA44040BE921F98C1E7036EE9CCB57F324313B60C4D8B9036
46F32607F13B4F31C6AB519E0EA18926B8B39E3A3DD93861E9D781]
Output from Faulty Execution $\omega(P, Q)'$:
[7414AEDA4113AF4A59A41591901EEA09C81A620153CAE069730F5616A14A484BD9D
4D5FE4515A9E19E11F8D6F8CDFC1AD5FA2A7C59B1D3C7F68D887C18A9DC2,79AF68D
3AB50ED5D9628A79EDC59EC9FF820AD7D3FB38C6713576F74AED35C28454C677119A
4D170E4700AEA9F4F8293EAF531E770FE25BDF19E08E8780C630D]

B.2 Steps to the Attack

Step 1: Divide the valid pairing by the faulty pairing.
[52002F3A6C90C9AE9BB0D7B6FB0C5B56C4DFD86DE24C6A6F95BC40E8656077631CD
6BD666596A20D82063685940CC77BBA9CCCFAD034283AA8F04F57CA379062,7412C5
50674512E2E168850B67D36E60EA9C2B053FF19985868FCF4E8DFCAC40A7B97329A7
AE2A11EECCE357DAF09D8C260AFCE63346022955CEBD2147135E9]

Step 2: Calculate the square root.
[1D7B3A07709ACC56F84A24ABB88EBE62FD4580542E612739DF8BC408F2A96075D1C
2689F2C8CB86F0AB7F794EBADB0A28AF982B437206E38AE4C66D49924F682,A1FECD
9DFEAE289E71CA3017312EA1EC55B4E0E816978002ADB6AE8483888D51C83A2BED12
8C1341F044EF0D228A592D68F829AF820C40F7F104227CE9BB3AF5]
This exploits the relationship,

$$\sqrt{\frac{\omega(P,Q)}{\omega(P,Q)'}} = \pm\left(\frac{[\lambda_i(x_A + x_Q) - y_A][-y_Q]}{[\lambda_i(x_A + x_Q) - y_A][y_Q]}\right)$$

where two roots ($\pm\sqrt{\ }$) will be produced. Here we just show the correct root, which will be denoted by N_R and N_C.

Step 3: From this relationship we can generate two equations which relate to the grouping of real and complex values. We will just show the real here:

$$(\lambda_i - N_R\lambda_i)x_Q + N_C y_Q + \lambda_i x_A - y_A - \lambda_i x_A + N_R y_A = 0$$

We can calculate the coefficients of this equation since they are composed of known/computable values.
$(\lambda_i - N_R\lambda_i) =$

64DCD4C10F7816EF8D7C389BA1709DA605EE81C5F67339764546C2C4B09756DAFC51
19AA299957AE05AE8FEB5A64DFBE801E0073DB161053319B6B6CD5256A1
$N_C =$
A1FECD9DFEAE289E71CA3017312EA1EC55B4E0E816978002ADB6AE8483888D51C83A
2BED128C1341F044EF0D228A592D68F829AF820C40F7F104227CE9BB3AF5
$\lambda_i x_A - y_A - N_R \lambda_i x_A + N_R y_A =$
895341DAD464B48011914A33021B77238FDEF077FBF59E124C007043AA5AB53C223D
7E0383761712E4ED4A87AF08242A31A94AC7D7EFB4E7F2F1AC3504924DC9

To solve this we will need another equation with the same unknowns. However since there exists a relationship between x_Q and y_Q, one equation will suffice.

Step 4: Substitute $\sqrt{x_Q^3 + ax_Q - b}$ for y_Q. This gives a cubic equation $x_Q^3 + A_0 x_Q^2 + B_0 x_Q + C_0 = 0$ where

$$A_0 = -\frac{(\lambda_i - N_R\lambda_i)^2}{N_C^2}$$
$$B_0 = \frac{(a - 2(\lambda_i - N_R\lambda_i)(x_A\lambda_i - y_A - N_R x_A\lambda_i + N_R y_A))}{N_C^2}$$
$$C_0 = -b - \frac{(x_A\lambda_i - y_A - N_R x_A\lambda_I + N_R y_A)^2}{N_C^2}$$

$A_0 =$
6802A674045A31776FFB8CFBB67AD446CE7AA807B8FC47B3DFA77C4FA8BC5879C0E0
5F5C00806413631186314B64E930BC5D05EF35D81B095AAB80B16D25C6E2
$B_0 =$
9960E747477925F838D6E65391C97CC2986BED3699CC28748FB80414AB1B709E47E5
267C0DE54B5084B4EAE935C88F09741189C3AF1B9AE7DE82B31D88E72D88
$C_0 =$
AF3B578B224A5C538C6E21DAA2A0319C890377BCB93057B0D2BEF9965F94ADB1276D
26C9151C4A7FEE8BC76CBDEF24D0CF80F32EB1F04BCF1AD4B3D14959BE48

Step 5: Solve with Cardano's Method. Given A_0, B_0, C_0 and the modulus p (which are all computable values by an adversary), solve for x_Q. The roots that our Cardano method produces are
root 1 =
B1F0D753A62BC0458E230371B2E663321CEDC5DC0BBBFA88B16BB4A549310D7297AE
3F90F87876C63D898891085FDB29C61005E94C85DB372879FA953420F48B
root 2 =
75C70FD2C28F0367D32F173281F4A611FF2DB79FAE53C328A1AC4B6379AD21170818
3462E800A5B7AB5AA5BD6A3D2597356877BA46BFD955EB8462D0CE4B565B
root 3 =
2049460A69A5DE02C59A0A82850F84F3CDE4117C9D6BD6EC60F1A80FA58957D93FB0
07216BEAA110E4934CD357F3083D088CEC64C0DEECA7C5B482F4C76D56E
where as you can see the first root corresponds to our secret coordinate x_Q.

Proxy Re-encryption Systems for Identity-Based Encryption

Toshihiko Matsuo

NTT DATA CORPORATION
matsuotsh@nttdata.co.jp

Abstract. A proxy re-encryption system allows the proxy to transform ciphertexts encrypted under Alice's public key into the different ciphertexts that can be decrypted by Bob's secret key. In this paper, we propose new proxy re-encryption systems; one for the transformation from ciphertexts encrypted under a traditional PKI-based public key into the ciphertexts that can be decrypted by an secret key for Identity-Based Encryption, and the other one for the transformation from ciphertexts encrypted in IBE manner into the different ciphertexts that can be decrypted by the other secret key for the IBE.

Keywords: proxy re-encryption system, public key encryption, identity-based encryption.

1 Introduction

1.1 Background

A proxy re-encryption system allows the proxy to transform ciphertexts computed under Alice's public key into the different ciphertexts that can be decrypted by using Bob's secret key. This system works as follows; Alice or a trusted third party generates a re-encryption key and sets it in a proxy. On receiving Alice's ciphertexts, the proxy transforms the ciphertext by running the re-encryption algorithm with the re-encryption key, and sends the transformed ciphertext to Bob. Bob decrypts it by his secret key. As it can be seen that Alice delegates her decryption rights to Bob via proxy, we call Alice a *delegator* and Bob a *delegatee*. The proxy re-encryption system should at least satisfy the following requirements; 1) a proxy alone cannot obtain the underlying plaintext, 2) and Bob cannot obtain the underlying plaintext without the proxy cooperating. The proxy re-encryption system can be a primitive for various attractive applications, and thus it has been active research area [BBS98, J99, DI03, ZMSR04, AFGH05, GA06].

One of the most promising application is the access control system over the network storage [MO97, AFGH05]. In this system, Alice performs a content holder who stores some contents encrypted under her public key in the network storage. The proxy performs an access controller who transforms the stored ciphertexts into the different ciphertexts that can be decrypted by Bob's secret

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 247–267, 2007.

key when Alice allows Bob to access her contents. Since the proxy can transform the stored ciphertexts without Alice's secret key, it can reduce the amount of trust in the access control server. Beside the access control system, the proxy re-encryption system can be applied to the secure e-mail forwarding system [AFGH05], the outsourced filtering of encrypted spam [AFGH05], the law enforcement [DI03] and so on.

1.2 Our Motivation and Contribution

Recently, *Identity-Based Encryption* (for short, IBE) has been one of the most active research area [BF01, BB04a, BB04b, GS04, W05, G06]. In the IBE system, a sender Catherine encrypts a message to an IBE receiver Alice by using Alice's identity as a public key. Providing that Alice sets her e-mail address to the public key and it includes the revocation date, Catherine can easily make sure not only that the public key belongs to Alice, but also when the public key is revoked. Therefore, the IBE system dramatically improves the key management workload while it is heavy burden in the traditional PKI-based public key encryption (for short, PKE) system. The IBE system necessarily requires a third party called Public Key Generator (PKG) which generates all secret keys for IBE users by using its master-secret key, and thus the IBE system works where the PKG operation can be allowed. As each user has own policy, rule, role or purpose in the ciphertext communication, one might adopts the IBE system because of its simple key management and the other one might employ the PKE system if she does not accepts the PKG operation for some reason. Then a lot of messages encrypted in the different manner circulate among the world, and this circumstance yields the demand for the proxy re-encryption systems transforming PKE ciphertexts into the different IBE ciphertexts (type 1), IBE ciphertexts into the different IBE ciphertexts (type 2), IBE ciphertexts into the different PKE ciphertexts (type 3), and PKE ciphertexts into the different PKE ciphertexts (type 4). However, there is no system for type 1 transformation and only a few systems for type 2 transformation are proposed so far [DI03, GA06]. Therefore, we propose two systems for type 1 and type 2 transformation.

Our first proposal, *hybrid proxy re-encryption system*, is the first system achieving type 1 transformation. Our second proposal for type 2 transformation, *identity-based proxy re-encryption system*, holds the following advantages compare to the previous proposals.

- Our system achieves optimal secret key size, that is, it needs no additional secret key besides the secret key of the underlying IBE system for delegatees to decrypt re-encrypted ciphertexts while it is required in [DI03].
- Our system achieves optimal ciphertext size, that is, the size of re-encrypted ciphertexts is the exactly same as that of the corresponding original ciphertexts while it is required to extend original ciphertext size for re-encryption in [GA06].[1]

[1] Actually, the re-encryption system achieving optimal ciphertext size is also proposed in [GA06]; however, there are some undesirable restriction on that system.

– Our system does not need additional algorithm or process for decrypting re-encrypted ciphertexts while it is required in [GA06].
– Our system is semantically secure in the standard model while previous systems [DI03, GA06] are semantically secure in the random oracle model (our security notion is slightly weaker than that defined in [GA06]).

1.3 Organization

The rest of this paper consists of four sections. Sec. 2 gives some definitions and preliminaries to understand our study. In Sec. 3, we present the hybrid proxy re-encryption system. We propose the identity-based proxy re-encryption system in Sec. 4 and finally conclude this study in Sec. 5.

2 Preliminaries

In the following, we sometimes use notation described in this section without notice. We denote the concatenation of a and b by $a||b$. We also denote random choice from a set S by $\stackrel{R}{\leftarrow} S$.

2.1 Bilinear Groups

Let $\mathbb{G}$ and $\mathbb{G}_1$ be multiplicative cyclic groups of prime order p, and g be a generator of $\mathbb{G}$. We say that $\mathbb{G}_1$ has an admissible bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ if the following conditions hold.

1. $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ for all a, b.
2. $\hat{e}(g, g) \neq 1$.
3. There is an efficient algorithm to compute $\hat{e}(g^a, g^b)$ for all a, b and g.

2.2 Assumption

Definition 1. *For randomly chosen integers $a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, a random generator $g \stackrel{R}{\leftarrow} \mathbb{G}$, and an element $R \stackrel{R}{\leftarrow} \mathbb{G}_1$, we define the advantage of an algorithm $\mathcal{A}$ in solving the decision Bilinear Diffie-Hellman (dBDH) problem as follows:*

$$\mathtt{Adv}_{\mathbb{G}}^{dbdh}(\mathcal{A}) = \left|\Pr[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, R) = 0]\right|$$

where the probability is over the random choice of generator $g \in \mathbb{G}$, the randomly chosen integers a, b, c, the random choice of $R \in \mathbb{G}_1$, and the random bits used by A. We say that the (k, t, ϵ)-dBDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the dBDH problem in $\mathbb{G}$ under a security parameter k.

2.3 Digital Signature Scheme

A digital signature scheme is made up of three algorithms, $\mathbf{KeyGen}_{\Sigma}$, **Sign**, and **Verify**, for generating keys, signing, and verifying signatures, respectively.

$\mathbf{KeyGen}_{\Sigma}(k)$. Given a security parameter k, generate a signing key sk_{Σ} and the corresponding verification key vk_{Σ}.
$\mathbf{Sign}(sk_{\Sigma}, M)$. Given the signing key sk_{Σ} and a message M, generate a signature σ.
$\mathbf{Verify}(vk_{\Sigma}, M, \sigma)$. Given the verification key vk_{Σ}, the message M and its signature σ, output 1 if σ is valid, otherwise output 0.

Though the standard notion of security for a digital signature scheme is called existential unforgeability under a chosen message attack [GMR88], we introduce the slightly different notion, strong existential unforgeability under a passive attack that is defined using the following game between a challenger and an adversary $\mathcal{A}$:

SetUp. The challenger runs algorithm $\mathbf{KeyGen}_{\Sigma}$ to obtain a signing key sk_{Σ} and the corresponding verification key vk_{Σ}. The adversary $\mathcal{A}$ is given vk_{Σ}.
Message-signature pair. Suppose that q is an integer. The challenger selects messages $M_1, \ldots, M_q$ from the message domain and makes the signatures $\sigma_1, \ldots, \sigma_q$ for each message M_i $(1 \leq i \leq q)$ where $\sigma_i = \mathbf{Sign}(sk_{\Sigma}, M_i)$. The challenger gives the message-signature pair sets $\mathsf{S}_{ms} = \{(M_i, \sigma_i)\}_{1 \leq i \leq q}$ to the adversary $\mathcal{A}$.
Output. Eventually, A outputs a pair (M', σ') and wins the game if $(M', \sigma') \notin \mathsf{S}_{ms}$ and $\mathbf{Verify}(vk_{\Sigma}, M', \sigma') = 1$.

Definition 2. *We define $\mathcal{A}$'s advantage in the games as follows.*

$$\mathtt{Adv}^{eu}(\mathcal{A}) = \Pr[(M', \sigma') \notin \mathsf{S}_{ms} \wedge \mathbf{Verify}(vk_{\Sigma}, M', \sigma') = 1]. \tag{1}$$

We say that the digital signature scheme is (k, t, q, ϵ)-strong existentially unforgeable under a passive attack if for any t time adversary $\mathcal{A}$ that observes at most q message-signature pairs under a security parameter k, we have that $\mathtt{Adv}^{eu}(\mathcal{A}) < \epsilon$.

2.4 PKI-Based Public Key Encryption System

A traditional PKI-based Public Key Encryption (PKE) system consists of the following algorithms.

$\mathbf{KeyGen}_{\mathsf{PKE}}(k, aux)$. Given a security parameters k and auxiliary input aux, generate a secret key sk and the corresponding public key pk.
$\mathbf{Enc}_{\mathsf{PKE}}(pk, aux, M)$. Given the public key pk with aux, compute the encryption of a message M, C_{PK}.
$\mathbf{Dec}_{\mathsf{PKE}}(sk, aux, C_{\mathsf{PK}})$. Given the secret key sk with aux, decrypt the ciphertext C_{PK}.

2.5 Identity Based Encryption System

An Identity Based Encryption (IBE) system consists of the following algorithms.

- $\mathbf{SetUp}_{\mathsf{IBE}}(k)$. Given a security parameter k, generate a pair $(parms, mk)$, where $parms$ denotes the public parameters and mk is the master-secret key.
- $\mathbf{KeyGen}_{\mathsf{IBE}}(mk, parms, \mathsf{ID})$. Given the master-secret key mk and an identity ID with $parms$, generate a secret key sk_{ID} for ID.
- $\mathbf{Enc}_{\mathsf{IBE}}(\mathsf{ID}, parms, M)$. Given a message M and the identity ID with $parms$, compute the encryption of M, C_{ID}, for ID.
- $\mathbf{Dec}_{\mathsf{IBE}}(sk_{\mathsf{ID}}, parms, C_{\mathsf{ID}})$. Given the secret key sk_{ID}, decrypt the ciphertext C_{ID}.

In setup, a trusted third party, PKG, runs $\mathbf{SetUp}_{\mathsf{IBE}}$ and generates its master-secret key and public parameters. When an IBE user requests a secret key corresponding to her identity (i.e. public key), PKG generates the secret key by running $\mathbf{KeyGen}_{\mathsf{IBE}}$, and give it to the user via secure and authenticated channel. A sender encrypts a message by running $\mathbf{Enc}_{\mathsf{IBE}}$ with the receiver's identity and public parameters. The receiver decrypts a ciphertext by running $\mathbf{Dec}_{\mathsf{IBE}}$ with her secret key.

Security. IBE security [BF01] is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- **Setup.** The challenger $\mathcal{C}$ runs the $\mathbf{SetUp}_{\mathsf{IBE}}$ algorithm and gives $\mathcal{A}$ the resulting system parameters, $parms$, keeping the master-secret key mk to itself.
- **Phase 1.** $\mathcal{A}$ adaptively queries $\mathcal{C}$ as follows: $\mathcal{A}$ requests the secret key for ID from $\mathcal{C}$. $\mathcal{C}$ generates the secret key sk_{ID} by running algorithm $\mathbf{KeyGen}_{\mathsf{IBE}}$ and returns them to $\mathcal{A}$. After some number of queries, $\mathcal{A}$ selects two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and a target identity ID^*, and sends them to $\mathcal{C}$.
- **Challenge.** Given $(M_0, M_1, \mathsf{ID}^*)$, $\mathcal{C}$ picks a random bit $d \in \{0, 1\}$ and sets the challenge ciphertext to $C_{\mathsf{ID}_d^*} = \mathbf{Enc}_{\mathsf{IBE}}(\mathsf{ID}^*, parms, M_d)$, which is sent to $\mathcal{A}$.
- **Phase 2.** $\mathcal{A}$ continues to issue queries as in Phase 1 with the restriction that $\mathcal{A}$ cannot issue secret key queries for ID^*.
- **Guess.** Finally, $\mathcal{A}$ outputs a guess $d' \in \{0, 1\}$.

The adversary $\mathcal{A}$ wins if $d' = d$. We say that the identity based encryption system is IND-ID-CPA secure if $|\Pr[d' = d] - 1/2|$ is negligible.

Definition 3. *We define $\mathcal{A}$'s advantage in an IND-ID-CPA games as follows*

$$\mathtt{Adv}^{id}_{\mathsf{IBE}}(\mathcal{A}) = 2(\Pr[d' = d] - 1/2) \tag{2}$$

We say that the an IBE system is (k, t, q, ϵ)-identity, adaptive chosen plaintext secure if for any t time IND-ID-CPA adversary $\mathcal{A}$ that makes at most q chosen secret key queries under a security parameter k we have that $\mathtt{Adv}^{id}_{\mathsf{IBE}}(\mathcal{A}) < \epsilon$. As shorthand, we say that an IBE system is (k, t, q, ϵ) IND-ID-CPA secure.

Canetti *et al.* [CHK03, CHK04] defined a weaker notion of security in which the adversary commits ahead of time to the public key it will attack. We refer to this notion as selective identity, chosen plaintext secure IBE (IND-sID-CPA). The game is exactly the same as IND-ID-CPA except that the adversary $\mathcal{A}$ discloses to the challenger the target identity ID^* before the Setup phase. The restrictions on secret key queries from Phase 2 also hold in Phase 1.

3 Hybrid Proxy Re-encryption System

In this section, we introduce our new proxy re-encryption system, *hybrid proxy re-encryption system*. It consists of a PKE system, an IBE system, and additional algorithms that allow ciphertexts encrypted under a PKE public key to be transformed into the different ciphertexts that can be decrypted by an IBE secret key.

3.1 Definition

There are four parties involved in a hybrid proxy re-encryption system, delegator, proxy, delegatee and its PKG. On receiving a ciphertext encrypted in the PKE manner by delegator's public key, the proxy re-encrypts it into ciphertexts that the delegatee who holds an IBE secret key can decrypt, using a re-encryption key generated by the delegator for a particular delegatee.

A hybrid proxy re-encryption consists of: 1) the four algorithms making up an IBE system $\mathbf{SetUp}_{\mathsf{IBE}}$, $\mathbf{KeyGen}_{\mathsf{IBE}}$, $\mathbf{Enc}_{\mathsf{IBE}}$, and $\mathbf{Dec}_{\mathsf{IBE}}$, 2) the three algorithms making up a PKE system $\mathbf{KeyGen}_{\mathsf{PKE}}, \mathbf{Enc}_{\mathsf{PKE}}$, and $\mathbf{Dec}_{\mathsf{PKE}}$, 3) and four algorithms for re-encryption, which are —

- $\mathbf{EGen}(sk_{\mathsf{ID}}, parms)$. Given an IBE secret key sk_{ID} for the IBE user ID with IBE public parameters $parms$, generate e_{ID} for re-encryption key generation.
- $\mathbf{KeyGen}_{\mathsf{PRO}}(sk, e_{\mathsf{ID}}, parms)$. Given a PKE secret key sk and e_{ID} with $parms$, generate a re-encryption key rk_{ID} that re-encrypts PKE ciphertexts into the IBE ciphertexts for ID.
- $\mathbf{ReEnc}(rk_{\mathsf{ID}}, parms, C_{\mathsf{PK}}, \mathsf{ID})$. Given the re-encryption key rk_{ID}, a ciphertext C_{PK} encrypted under the traditional public key, and ID with $parms$, re-encrypt ciphertext C_{PK} into C_{ID} that can be decrypted by the IBE user ID.
- $\mathbf{Check}(parms, C_{\mathsf{PK}}, pk)$. Given C_{PK} and pk with $parms$, output 0 if C_{PK} is a malformed ciphertext. Otherwise, output 1.

Let the PKG employ the digital signature scheme $(\mathbf{KeyGen}_{\Sigma}, \mathbf{Sign}, \mathbf{Verify})$ described in Sec. 2.3; however we do not describe it in the above for conciseness. When a PKE user delegates her decryption rights to an IBE user, the hybrid proxy re-encryption system works as follows.

– *SetUp:*
  1. The PKG generates its signing key sk_{Σ} and the corresponding verification key vk_{Σ} by running $\mathbf{KeyGen}_{\Sigma}$. The PKG also generates its

master-secret key mk and public parameters $parms$ by running $\mathbf{SetUp}_{\mathsf{IBE}}$. The PKG makes $(parms, vk_{\Sigma})$ public, keeping (mk, sk_{Σ}) to itself.

2. The PKE user generates its secret key sk and the corresponding public key pk by running $\mathbf{KeyGen}_{\mathsf{PKE}}$ with the input $parms$, and makes pk public, keeping sk to itself.

– *Re-encryption key generation and deployment:*

1. When one requests the delegation from the PKE user (i.e. delegator) to the IBE user ID (i.e. delegatee),
   - If no IBE secret key has issued to the delegatee, the PKG generates sk_{ID} by running $\mathbf{KeyGen}_{\mathsf{IBE}}$, and computes e_{ID} by running **EGen**. The PKG makes a digital signature σ_e for $\mathsf{ID}||e_{\mathsf{ID}}$ by running **Sign**. Then PKG issues $(sk_{\mathsf{ID}}, \mathsf{ID}||e_{\mathsf{ID}}, \sigma_e)$ to the delegatee. The delegatee sends $(\mathsf{ID}||e_{\mathsf{ID}}, \sigma_e)$ to the delegator.
   - Otherwise, the delegatee sends previously issued $\mathsf{ID}||e_{\mathsf{ID}}$ and the corresponding signature σ_e to the delegator.
2. On receiving $(\mathsf{ID}||e_{\mathsf{ID}}, \sigma_e)$, the delegator verifies it by running **Verify** with vk_{Σ}.
   - If it is valid then the delegator generates a re-encryption key rk_{ID} by running $\mathbf{KeyGen}_{\mathsf{PRO}}$ with the input e_{ID}. The delegator sets rk_{ID} in the proxy.
   - Otherwise the delegator rejects.

– *Re-encryption:* Suppose that one sends a ciphertext C_{PK} to the delegatee ID via the proxy. On receiving the PKE ciphertext C_{PK}, the proxy runs the algorithm **Check** with the input $(parms, C_{\mathsf{PK}}, pk)$.
   - If **Check** outputs 0 then the proxy rejects the re-encryption request.
   - Otherwise, the proxy re-encrypts C_{PK} into C_{ID} by running **ReEnc**, and sends C_{ID} to the delegatee ID.

– *Decryption:* The delegatee decrypts C_{ID} by running $\mathbf{Dec}_{\mathsf{IBE}}$ with the IBE secret key sk_{ID}.

3.2 Security Notion

In the following, each value appeared in i-th query by the adversary and in the corresponding answer is denoted with letter i. We sometimes denote the delegatee's identity in i-th query by ID_i.

Chosen Plaintext Security: We model chosen plaintext security for a hybrid proxy re-encryption system as a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In this game, the adversary is allowed to adaptively choose the IBE secret key queries and re-encryption key queries. Intuitively, these queries imply the situation that: (1)the adversary compromises arbitrary IBE users and obtains their secret keys, (2)the adversary compromises arbitrary proxy and obtains the re-encryption keys, (3)and the adversary requests the re-encryption key generation of the delegator. Since the adversary obviously wins the game if it obtains

both delegatee's secret key and the corresponding re-encryption key involving the same identity, she is not allowed to ask such query. More precisely, IND-ID-CPA security is defined as follows:

Setup. The challenger $\mathcal{C}$ generates (sk_Σ, vk_Σ) by running $\mathbf{KeyGen}_\Sigma$. $\mathcal{C}$ generates $(parms, mk)$ by running $\mathbf{SetUp}_{\mathsf{IBE}}$. $\mathcal{C}$ also generates (pk, sk) by running $\mathbf{KeyGen}_{\mathsf{PKE}}$. $\mathcal{C}$ gives $(parms, pk, vk_\Sigma)$ to $\mathcal{A}$, keeping (mk, sk, sk_Σ) to itself.

Phase 1. Given $(parms, pk, vk_\Sigma)$, $\mathcal{A}$ adaptively queries the challenger $\mathcal{C}$. When $\mathcal{A}$ queries $\mathcal{C}$, it responds as follows:

- **Secret key queries.** When $\mathcal{A}$ queries $\mathcal{C}$ at a point ID_i, $\mathcal{C}$ generates a secret key sk_{ID_i} for ID_i by running $\mathbf{KeyGen}_{\mathsf{IBE}}$. $\mathcal{C}$ computes e_{ID_i} by running **EGen** with the input sk_{ID_i}. $\mathcal{C}$ generates a signature σ_{e_i} for $\mathsf{ID}_i||e_{\mathsf{ID}_i}$ by running **Sign**, and returns $(sk_{\mathsf{ID}_i}, \mathsf{ID}_i||e_{\mathsf{ID}_i}, \sigma_{e_i})$ to $\mathcal{A}$.
- **Type-1 re-encryption key queries.** When $\mathcal{A}$ queries $\mathcal{C}$ at a point ID_i, $\mathcal{C}$ generates an IBE secret key sk_{ID_i} by running $\mathbf{KeyGen}_{\mathsf{IBE}}$, and computes e_{ID_i} by running **EGen** with the input sk_{ID_i}. $\mathcal{C}$ generates a signature σ_{e_i} for $\mathsf{ID}_i||e_{\mathsf{ID}_i}$ by running **Sign**. $\mathcal{C}$ runs $\mathbf{KeyGen}_{\mathsf{PRO}}$ with the inputs e_{ID_i}, and returns the resulting re-encryption key rk_{ID_i} with $(\mathsf{ID}_i||e_{\mathsf{ID}_i}, \sigma_{e_i})$ to $\mathcal{A}$.
- **Type-2 re-encryption key queries.** Suppose that $\mathcal{A}$ queries $\mathcal{C}$ about $(\mathsf{ID}_i||e_{\mathsf{ID}_i}, \sigma_{e_i})$. If $(\mathsf{ID}_i||e_{\mathsf{ID}_i}, \sigma_{e_i})$ has already generated in the answering for secret key query, $\mathcal{C}$ rejects the query. Otherwise $\mathcal{C}$ verifies $(\mathsf{ID}_i||e_{\mathsf{ID}_i}, \sigma_{e_i})$ by running **Verify** with vk_Σ and works as follows;
  - If it is valid then $\mathcal{C}$ runs $\mathbf{KeyGen}_{\mathsf{PRO}}$ with the inputs e_{ID_i}, and returns the resulting re-encryption key rk_{ID_i}.
  - Otherwise $\mathcal{C}$ rejects the query.

Challenge. After some queries, $\mathcal{A}$ selects two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and sends them to $\mathcal{C}$. $\mathcal{C}$ picks $d \stackrel{R}{\leftarrow} \{0,1\}$ and computes

$$C_{\mathsf{PK}_d} = \mathbf{Enc}_{\mathsf{PKE}}(pk, parms, M_d).$$

$\mathcal{C}$ returns C_{PK_d} to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ continues to issue queries as in Phase 1, and $\mathcal{C}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs a guess $d' \in \{0,1\}$.

The adversary $\mathcal{A}$ wins if $d' = d$. The hybrid proxy re-encryption system is secure in the sense of IND-ID-CPA if $|\Pr[d' = d] - 1/2|$ is negligible.

Definition 4. *Let $\mathcal{A}$ be an adversary against the hybrid proxy re-encryption system. Define the IND-ID-CPA advantage of $\mathcal{A}$ as follows.*

$$\mathtt{Adv}^{id}_{\mathsf{hyd}}(\mathcal{A}) = 2(\Pr[d' = d] - 1/2). \tag{3}$$

We say that a hybrid proxy re-encryption system is (k, t, q, ϵ) adaptive chosen plaintext secure if for any t time IND-ID-CPA adversary $\mathcal{A}$ that makes at most q chosen queries under a security parameter k we have that $\mathtt{Adv}^{id}_{\mathsf{hyd}}(\mathcal{A}) < \epsilon$. As shorthand, we say that a hybrid proxy re-encryption system is (k, t, q, ϵ) IND-ID-CPA secure.

Note that this game encompasses the notion of semantic security for the PKE system, as well as that for the IBE system, and also the notion that a set of re-encryption keys cannot be "combined"to form new re-encryption keys for other identities. For example, if the PKE system is not semantically secure, then the adversary can win the game by simply distinguishing the challenge ciphertext.

3.3 Construction

We describe our hybrid proxy re-encryption system involving the ElGamal-type PKE system and the BB-IBE system [BB04a]. Let the PKG employ a digital signature scheme ($\mathbf{KeyGen}_\Sigma, \mathbf{Sign}, \mathbf{Verify}$). We describe the following algorithms making up the system:

– *The underlying IBE system (BB-IBE system):*

$\mathbf{SetUp}_{\mathsf{IBE}}(k)$. Given a security parameter k, select a random generator $g \in \mathbb{G}$ and random elements $g_2, h \in \mathbb{G}$. Pick a random $\alpha \in \mathbb{Z}_p^*$. Set $g_1 = g^\alpha$, $mk = g_2^\alpha$, and $parms = (g, g_1, g_2, h)$. Let mk be the master-secret key and let $parms$ be the public parameters.

$\mathbf{KeyGen}_{\mathsf{IBE}}(mk, parms, \mathsf{ID})$. Given $mk = g_2^\alpha$ and ID with $parms$, pick a random $u \in \mathbb{Z}_p^*$. Set

$$sk_{\mathsf{ID}} = (d_0, d_1) = \left(g_2^\alpha (g_1^{\mathsf{ID}} h)^u, g^u\right).$$

$\mathbf{Enc}_{\mathsf{IBE}}(\mathsf{ID}, parms, M)$. To encrypt a message $M \in \mathbb{G}_1$ under the public key $\mathsf{ID} \in \mathbb{Z}_p^*$, pick a random $r \in \mathbb{Z}_p^*$ and compute

$$C_{\mathsf{ID}} = \left(g^r, (g_1^{\mathsf{ID}} h)^r, M\hat{e}(g_1, g_2)^r\right) \in \mathbb{G}^2 \times \mathbb{G}_1.$$

$\mathbf{Dec}_{\mathsf{IBE}}(sk_{\mathsf{ID}}, parms, C_{\mathsf{ID}})$. Given ciphertext $C_{\mathsf{ID}} = (C_1, C_2, C_3)$ and the secret key $sk_{\mathsf{ID}} = (d_0, d_1)$ with $prams$, compute

$$M = \frac{C_3 \hat{e}(d_1, C_2)}{\hat{e}(d_0, C_1)}.$$

– *The underlying PKE system (ElGamal-type PKE system):*

$\mathbf{KeyGen}_{\mathsf{PKE}}(k, parms)$. Given a security parameter k and $parms$, pick a random $\beta, \theta, \delta \in \mathbb{Z}_p^*$. Set $g_3 = g^\theta$, $g_4 = g_1^\beta$ and $g_5 = h^\delta$. The public key is $pk = (g_3, g_4, g_5)$. The secret key is $sk = (\theta, \beta, \delta)$.

$\mathbf{Enc}_{\mathsf{PKE}}(pk, parms, M)$. Given $pk = (g_3, g_4, g_5)$ and a message M with $parms$, pick a random $r \in \mathbb{Z}_p^*$ and compute

$$C_{\mathsf{PK}} = \left(g_3^r, g_4^r, g_5^r, M\hat{e}(g_1, g_2)^r\right) \in \mathbb{G}^3 \times \mathbb{G}_1.$$

$\mathbf{Dec}_{\mathsf{PKE}}(sk, parms, C_{\mathsf{PK}})$. Given $C_{\mathsf{PK}} = (C_1, C_2, C_3, C_4)$ and the secret key $sk = (\theta, \beta, \delta)$ with $parms$, compute $M = C_4/\hat{e}(C_2^{1/\beta}, g_2)$.

– *The delegation system:*

EGen$(sk_{\mathsf{ID}}, parms)$. Given $sk_{\mathsf{ID}} = (d_0, d_1) = (g_2^{\alpha}(g_1^{\mathsf{ID}}h)^u, g^u)$ for ID with $parms$, set $e_{\mathsf{ID}} = d_1 = g^u$.

KeyGen$_{\mathsf{PRO}}(sk, e_{\mathsf{ID}}, parms)$. Given $sk = (\theta, \beta, \delta)$ and $e_{\mathsf{ID}} = g^u$ for ID with $parms$, set $rk_{\mathsf{ID}} = (\theta, g^{u/\beta}, \delta)$.

ReEnc$(rk_{\mathsf{ID}}, parms, C_{\mathsf{PK}}, \mathsf{ID})$. Given a PKE ciphertext $C_{\mathsf{PK}} = (C_1, C_2, C_3, C_4)$, the re-encryption key $rk_{\mathsf{ID}} = (\theta, g^{u/\beta}, \delta)$ and ID with $parms$, re-encrypt the ciphertext C_{PK} into C_{ID} as follows.

$$C_{\mathsf{ID}} = (C_1', C_2', C_3') = \left(C_1^{1/\theta}, C_3^{1/\delta}, C_4\hat{e}(g^{u/\beta}, C_2^{\mathsf{ID}})\right) \in \mathbb{G}^2 \times \mathbb{G}_1.$$

Check$(parms, C_{\mathsf{PK}}, pk)$. Given $C_{\mathsf{PK}} = (C_1, C_2, C_3, C_4)$ and $pk = (g_3, g_4, g_5)$ with $parms$, set $v_1 = \hat{e}(C_1, g_4)$, $v_2 = \hat{e}(C_2, g_3)$, $v_3 = \hat{e}(C_2, g_5)$ and $v_4 = \hat{e}(C_3, g_4)$. If $v_1 = v_2$ and $v_3 = v_4$ then output 1, otherwise output 0.

In the proposed system, the PKG can decrypt all re-encrypted ciphertexts, thus one might wonder that this contradicts our motivation described in Sec. 1.2, that is, a PKE user who does not allow the PKG operation seems to fully trust the PKG; however, there is no contradiction. The PKG, or delegated IBE users, can not decrypt any original ciphertext without the proxy cooperating. Thus the delegator still control what ciphertexts she allows to decrypt if the proxy only re-encrypts the ciphertexts sent from entities allowed by the delegator. Though it is out of scope how the proxy works under such policy, we consider it is feasible by using actual techniques.[2]

3.4 Security Analysis

We first describe why the proxy re-encrypting does not make the underlying public key cryptosystems weak.

In our system, the re-encryption key $rk_{\mathsf{ID}} = (\theta, g^{u/\beta}, \delta)$ involves the delegator's decryption key β and the second component of delegatee's IBE secret key $d_1 = g^u$. Thus it might reveal some information about β and g^u; however this does not make the underlying public key cryptosystems weak. This is because

- it is computationally hard to recover β completely from the public key and the re-encryption key if the discrete logarithm problem is hard, and
- the underlying IBE can be proved semantically secure even if the second component of the secret key d_1 is exposed. (See Lemma 1 in Appendix A).

Therefore our proxy re-encryption system is secure as long as the re-encryption key is generated and deployed appropriately, and the digital signature system is used to ensure appropriate re-encryption key generation.

The above observation yields the security notion in section 3.2. Then, it is sufficient to show our system being secure if the following theorem holds.

Theorem 1. *Suppose that the (k, t, ϵ)-dBDH assumption holds and the PKG's digital signature scheme is (k, t', q, ϵ')-strong existentially unforgeable. Then the*

[2] It does not matter whether cryptographic or non-cryptographic way.

hybrid proxy re-encryption system is (k, t'', q, ϵ'') *IND-ID-CPA secure for any* q, k, $\epsilon'' \leq \epsilon + \epsilon'$, *and* $t'' + t' < t - \Theta(\tau_e q + \tau_s q + \tau_v q)$ *where* τ_e *is the maximum time for an exponentiation in* $\mathbb{G}$, τ_s *is the maximum time for running* **Sgin**, *and* τ_v *is the maximum time for running* **Verify**.

Proof. Let $\mathcal{A}$ be an adversary against the hybrid proxy re-encryption system in the IND-ID-CPA sense. We construct an adversary B which solves the dBDH problem in $\mathbb{G}$ by utilizing $\mathcal{A}$. Providing that $\mathcal{B}$ is given an input $(g, \Gamma_1, \Gamma_2, \Gamma_3, X) = (g, g^a, g^b, g^c, X)$, where $X = \hat{e}(g,g)^{abc}$ or $X = R \stackrel{R}{\leftarrow} \mathbb{G}_1$. We describe how $\mathcal{B}$ works in the following.

Initialization. $\mathcal{B}$ generates a blank list QAL to write down query-answer pairs for every query.

Setup. $\mathcal{B}$ selects a (k, t', q, ϵ')-strong existentially unforgeable digital signature scheme $(\mathbf{KeyGen}_\Sigma, \mathbf{Sign}, \mathbf{Verify})$ and generates (sk_Σ, vk_Σ) by running $\mathbf{KeyGen}_\Sigma$. To generate the system parameters, $\mathcal{B}$ picks $x, y, z, w \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and sets $g_1 = \Gamma_1$, $g_2 = \Gamma_2$, $h = g^z$ $g_3 = g^x$, $g_4 = g^y$ and $g_5 = h^w$. It gives $\mathcal{A}$ the system parameters $parms = (g, g_1, g_2, h)$, $pk = (g_3, g_4, g_5)$ and vk_Σ. Note that the corresponding PKG's master-secret key, which is unknown to $\mathcal{B}$, is $g_2^a = g^{ab} \in \mathbb{G}$.

Phase 1. Given pk, $parms$ and vk_Σ, $\mathcal{A}$ asks some queries to the challenger. When $\mathcal{A}$ queries the challenger, $\mathcal{B}$ works as follows.

- **Secret key queries.** Suppose that $\mathcal{A}$ queries the challenger at a point ID_i.
  - If $\mathsf{ID}_i \neq 0$ then $\mathcal{B}$ selects $r_i \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, sets

$$sk_{\mathsf{ID}_i} = (d_0, d_1) = \left(g_2^{\frac{-z}{\mathsf{ID}_i}} (g_1^{\mathsf{ID}_i} g^z)^{r_i}, g_2^{\frac{-1}{\mathsf{ID}_i}} g^{r_i}\right)$$

    and $e_{\mathsf{ID}_i} = d_1$. $\mathcal{B}$ computes $\mathbf{Sign}(sk_\Sigma, \mathsf{ID}_i || e_{\mathsf{ID}_i}) = \sigma_{e_i}$, and returns $(sk_{\mathsf{ID}_i}, \mathsf{ID}_i || e_{\mathsf{ID}_i}, \sigma_{e_i})$ to $\mathcal{A}$. $\mathcal{B}$ adds the query and the answer to the list QAL.
  - Otherwise $\mathcal{B}$ rejects the query.
- **Type-1 re-encryption key queries.** When $\mathcal{A}$ queries the challenger at a point ID_i, $\mathcal{B}$ selects $r_i' \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, sets $rk_{\mathsf{ID}_i} = (x, g_1^{r_i'}, w)$ and $e_{\mathsf{ID}_i} = g^{y r_i'}$. $\mathcal{B}$ generates a signature σ_{e_i} for $\mathsf{ID}_i || e_{\mathsf{ID}_i}$ by running **Sign**. $\mathcal{B}$ returns rk_{ID_i} with $(\mathsf{ID}_i || e_{\mathsf{ID}_i}, \sigma_{e_i})$ to $\mathcal{A}$. $\mathcal{B}$ adds the query and the answer to the list QAL.
- **Type-2 re-encryption key queries.** Suppose that $\mathcal{A}$ queries the challenger about $(\mathsf{ID}_i || e_{\mathsf{ID}_i}, \sigma_{e_i})$. $\mathcal{B}$ checks the list QAL.
  - If the IBE secret key sk_{ID_i} corresponding to $(\mathsf{ID}_i || e_{\mathsf{ID}_i}, \sigma_{e_i})$ is in the list then $\mathcal{B}$ rejects the query.
  - If rk_{ID_i} corresponding to $(\mathsf{ID}_i || e_{\mathsf{ID}_i}, \sigma_{e_i})$ is in the list then $\mathcal{B}$ returns rk_{ID_i} to $\mathcal{A}$.
  - Otherwise, $\mathcal{B}$ computes $v = \mathbf{Verify}(vk_\Sigma, \mathsf{ID}_i || e_{\mathsf{ID}_i}, \sigma_{e_i})$.
    - If $v = 1$ then $\mathcal{B}$ halts.
    - Otherwise, $\mathcal{B}$ rejects the query.

Challenge. After some queries, $\mathcal{A}$ selects two equal length plaintexts $M_0, M_1 \in \mathcal{M}$. Given (M_0, M_1), $\mathcal{B}$ selects $d \stackrel{R}{\leftarrow} \{0,1\}$ and sets

$$C_{\mathsf{PK}_d} = (\Gamma_3^x, \Gamma_3^y, \Gamma_3^{zw}, M_d X).$$

$\mathcal{B}$ returns C_{PK_d} to $\mathcal{A}$. Notice that if $X = \hat{e}(g,g)^{abc} = \hat{e}(g_1, g_2)^c$ then C_{PK_d} is a valid encryption of M_d. On the other hand, if X is uniform and independent in $\mathbb{G}_1$ then C_{PK_d} is independent of d in the adversary's view.

Phase 2. $\mathcal{A}$ continues to issue queries as in Phase 1, and $\mathcal{B}$ responds as before.

Solve. Finally, $\mathcal{A}$ outputs a guess $d' \in \{0,1\}$. $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $d' = d$ then $\mathcal{B}$ outputs 1 meaning $X = \hat{e}(g,g)^{abc}$. Otherwise, it outputs 0 meaning $X = R$.

We claim that $\mathcal{B}$ generates a valid secret key and the corresponding auxiliary information for ID_i. To see this, let $\tilde{u}_i = r_i - \frac{b}{\mathsf{ID}_i}$. Then we have that

$$\begin{aligned}(d_{\mathsf{ID}_i}, e_{\mathsf{ID}_i}) = \left(g_2^{\frac{-z}{\mathsf{ID}_i}} (g_1^{\mathsf{ID}_i} g^z)^{r_i}, g_2^{\frac{-1}{\mathsf{ID}_i}} g^{r_i}\right) &= \left(\frac{g_2^a (g_1^{\mathsf{ID}_i} g^z)^{r_i}}{(g_1^{\mathsf{ID}_i} g^z)^{\frac{b}{\mathsf{ID}_i}}}, g^{r_i - \frac{b}{\mathsf{ID}_i}}\right) \\ &= \left(g_2^a (g_1^{\mathsf{ID}_i} g^z)^{r_i - \frac{b}{\mathsf{ID}_i}}, g^{r_i - \frac{b}{\mathsf{ID}_i}}\right) \\ &= \left(g_2^a (g_1^{\mathsf{ID}_i} h)^{\tilde{u_i}}, g^{\tilde{u_i}}\right)\end{aligned}$$

We also claim that $\mathcal{B}$ can perfectly simulate the re-encryption key for ID_i since it looks random and independent of any other values if the adversary does not obtain the corresponding secret key.

$\mathcal{B}$ fails to simulate the challenger if $\mathcal{B}$ halts in the Type-2 re-encryption key query. Otherwise $\mathcal{B}$ perfectly simulates the challenger. The maximum probability of that $\mathcal{B}$ halts is obviously upper-bounded by $\mathtt{Adv}^{eu}(\mathcal{A})$. Therefore, we conclude the theorem.

4 Identity Based Proxy Re-encryption System

An identity-based proxy re-encryption system consists of an IBE system and additional algorithms that allow ciphertexts encrypted under one's IBE public key to be transformed into the different ciphertexts that can be decrypted by the other's IBE secret key. In this section, we describe our identity-based proxy re-encryption system.

4.1 Definition

There are five entities involved in an identity-based proxy re-encryption system, delegator, proxy, delegatee, PKG and *Re-encryption Key Generator*, RKG.[3] In this system, each of delegator and delegatee is an IBE user. The RKG generates re-encryption keys and sets them into the proxy via secure channel while

[3] The PKG and the RKG might be operated by one entity.

the delegator does in the hybrid proxy re-encryption system. An identity-based proxy re-encryption system consists of: 1) the four algorithms making up an IBE system $\mathbf{SetUp}_{\mathsf{IBE}}$, $\mathbf{KeyGen}_{\mathsf{IBE}}$, $\mathbf{Enc}_{\mathsf{IBE}}$, and $\mathbf{Dec}_{\mathsf{IBE}}$, 2) and five algorithms for re-encryption, which are –

EGen$(sk_{\mathsf{ID}}, parms)$. Given an IBE secret key sk_{ID} for ID with $parms$, generate e_{ID} for re-encryption key generation.

$\mathbf{KeyGen}_{\mathsf{RKG}}(mk, parms)$. Given an IBE master-secret key mk with $parms$, generate a secret key sk_R for re-encryption.

$\mathbf{KeyGen}_{\mathsf{PRO}}(sk_R, e_{\mathsf{ID}'}, parms, \mathsf{ID}, \mathsf{ID}')$. Given sk_R, $e_{\mathsf{ID}'}$, the delegator's identity ID and the delegatee's identity ID' with $parms$, generate a re-encryption key $rk_{\mathsf{ID}\to\mathsf{ID}'}$.

ReEnc$(rk_{\mathsf{ID}\to\mathsf{ID}'}, parms, C_{\mathsf{ID}}, \mathsf{ID}, \mathsf{ID}')$. Given the delegator's identity ID, the delegatee's identity ID', the re-encryption key $rk_{\mathsf{ID}\to\mathsf{ID}'}$, and an IBE ciphertext C_{ID} with $parms$, re-encrypt C_{ID} into the different IBE ciphertext $C_{\mathsf{ID}'}$.

Check$(parms, C_{\mathsf{ID}}, \mathsf{ID})$. Given the delegator's identity ID and an IBE ciphertext C_{ID} with $parms$, output 0 if C_{ID} is a malformed ciphertext for ID. Otherwise, output 1.

We assume that the PKG adopts the digital signature scheme described in Sec. 2.3. When one delegates her decryption rights to the other, the identity-based proxy re-encryption system works as follows.

– *SetUp:* The PKG generates
  1. its signing key sk_Σ and the corresponding verification key vk_Σ by running $\mathbf{KeyGen}_\Sigma$,
  2. its master-secret key mk and public parameters $parms$ by running $\mathbf{SetUp}_{\mathsf{IBE}}$,
  3. and a secret key sk_R for the RKG by running $\mathbf{KeyGen}_{\mathsf{RKG}}$.

  The PKG makes $(vk_\Sigma, parms)$ public and sets sk_R in the RKG, keeping (mk, sk_Σ) to itself.
– *Re-encryption key generation and deployment:* When one requests the delegation from ID (i.e. delegator) to ID' (i.e. delegatee), the RKG makes sure that ID approves the delegation to ID'. If ID denies it, the RKG rejects the request. Otherwise the RKG works as follows.
  - If no IBE secret key has issued to the delegatee ID', the PKG generates $sk_{\mathsf{ID}'}$ by running $\mathbf{KeyGen}_{\mathsf{IBE}}$, and computes $e_{\mathsf{ID}'}$ by running **EGen**. The PKG makes a digital signature σ'_e for $\mathsf{ID}'||e_{\mathsf{ID}'}$ by running **Sign**. Then PKG issues $(sk_{\mathsf{ID}'}, \mathsf{ID}'||e_{\mathsf{ID}'}, \sigma'_e)$ to the delegatee.
  - On receiving $(\mathsf{ID}'||e_{\mathsf{ID}'}, \sigma'_e)$ from the delegatee, the RKG verifies it by running **Verify** with vk_Σ.
    * If it is valid then the RKG generates a re-encryption key $rk_{\mathsf{ID}\to\mathsf{ID}'}$ by running $\mathbf{KeyGen}_{\mathsf{PRO}}$ with the input $e_{\mathsf{ID}'}$. The RKG sets $rk_{\mathsf{ID}\to\mathsf{ID}'}$ in the proxy.
    * Otherwise the RKG rejects the request.

– *Re-encryption:* Suppose that one sends a ciphertext C_{ID} to the delegatee ID'. On receiving the ciphertext C_{ID}, the proxy rejects the re-encryption request if $rk_{\mathsf{ID}\to\mathsf{ID}'}$ does not exists. Otherwise, the proxy runs the algorithm **Check** with the input $(parms, C_{\mathsf{ID}}, \mathsf{ID})$.
  - If **Check** outputs 0 then the proxy rejects the re-encryption request.
  - Otherwise, the proxy re-encrypts C_{ID} into $C_{\mathsf{ID}'}$ by running **ReEnc**, and sends $C_{\mathsf{ID}'}$ to the delegatee ID'.
– *Decryption:* The delegatee decrypts $C_{\mathsf{ID}'}$ by running $\mathbf{Dec}_{\mathsf{IBE}}$ with the secret key $sk_{\mathsf{ID}'}$.

4.2 Security Notion

In the following, each value appeared in i-th query by the adversary and in the corresponding answer is denoted with letter i. We sometimes denote a delegator's identity by ID_i and a delegatee's one by ID'_i that the adversary asks in i-th query. $\mathsf{ID}_i \to \mathsf{ID}'_i$ represents the delegation from ID_i to ID'_i.

Chosen Plaintext Security: We model chosen plaintext security for an identity-based proxy re-encryption system as a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In this game, the adversary is allowed to adaptively choose the secret key queries, re-encryption key queries and re-encryption queries. Intuitively, these queries imply the situation that: (1)the adversary compromises arbitrary IBE users and obtains their secret keys, (2)the adversary compromises arbitrary proxy and obtains the re-encryption keys, (3)the adversary requests the re-encryption key generation of the RKG, (4)and the adversary obtains re-encrypted ciphertexts by using proxy as an oracle. Since the adversary obviously wins the game if it obtains the secret key for the target identity, she is not allowed to ask such queries. Besides this, the adversary also wins the game if she obtains both of the delegatee's secret key and the corresponding re-encryption key involving the same identity because the re-encryption key is independent of the delegator's secret key. Therefore she is also not allowed to ask such queries. More precisely, IND-ID-CPA security is defined as follows:

Setup. The challenger $\mathcal{C}$ selects a digital signature scheme ($\mathbf{KeyGen}_{\Sigma}$, **Sign**, **Verify**). $\mathcal{C}$ generates
1. $(sk_{\Sigma}, vk_{\Sigma})$ by running $\mathbf{KeyGen}_{\Sigma}$,
2. $(parms, mk)$ by running $\mathbf{SetUp}_{\mathsf{IBE}}$, and
3. sk_R by running $\mathbf{KeyGen}_{\mathsf{RKG}}$.

$\mathcal{C}$ gives $(parms, vk_{\Sigma})$ to $\mathcal{A}$, keeping (mk, sk_{Σ}, sk_R) to itself.

Phase 1. Given $(parms, vk_{\Sigma})$, $\mathcal{A}$ adaptively queries $\mathcal{C}$. When $\mathcal{A}$ queries $\mathcal{C}$, it responds as follows:
– **Secret key queries.** When $\mathcal{A}$ queries $\mathcal{C}$ at a point ID_i, $\mathcal{C}$ generates a secret key sk_{ID_i} for ID_i by running $\mathbf{KeyGen}_{\mathsf{IBE}}$. $\mathcal{C}$ computes e_{ID_i} by running **EGen** with the input sk_{ID_i}. $\mathcal{C}$ generates a signature σ_{e_i} for $\mathsf{ID}_i||e_{\mathsf{ID}_i}$ by running **Sign**, and $\mathcal{C}$ returns $(sk_{\mathsf{ID}_i}, \mathsf{ID}_i||e_{\mathsf{ID}_i}, \sigma_{e_i})$ to $\mathcal{A}$.

- **Type-1 re-encryption key queries.** When $\mathcal{A}$ queries $\mathcal{C}$ about $\mathsf{ID}_i \rightarrow \mathsf{ID}'_i$, $\mathcal{C}$ generates an IBE secret key $sk_{\mathsf{ID}'_i}$ by running $\mathbf{KeyGen}_{\mathsf{IBE}}$, and computes $e_{\mathsf{ID}'_i}$ by running **EGen** with the input $sk_{\mathsf{ID}'_i}$. $\mathcal{C}$ generates a signature $\sigma_{e'_i}$ for $\mathsf{ID}'_i||e_{\mathsf{ID}'_i}$ by running **Sign**. $\mathcal{C}$ runs $\mathbf{KeyGen}_{\mathsf{PRO}}$ with the inputs $e_{\mathsf{ID}'_i}$, and returns the resulting re-encryption key $rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i}$ with $(\mathsf{ID}'_i||e_{\mathsf{ID}'_i}, \sigma_{e'_i})$ to $\mathcal{A}$.
- **Type-2 re-encryption key queries.** Suppose that $\mathcal{A}$ queries $\mathcal{C}$ about $(\mathsf{ID}_i \rightarrow \mathsf{ID}'_i, \mathsf{ID}'_i||e_{\mathsf{ID}'_i}, \sigma_{e'_i})$. If $(\mathsf{ID}'_i||e_{\mathsf{ID}'_i}, \sigma_{e'_i})$ has already generated in the answering for secret key query, then $\mathcal{C}$ rejects the query. Otherwise $\mathcal{C}$ verifies $(\mathsf{ID}'_i||e_{\mathsf{ID}'_i}, \sigma_{e'_i})$ by running **Verify** with vk_{Σ} and works as follows;
  - If it is valid then $\mathcal{C}$ runs $\mathbf{KeyGen}_{\mathsf{PRO}}$ with the input $e_{\mathsf{ID}'_i}$, and returns the resulting re-encryption key $rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i}$.
  - Otherwise $\mathcal{C}$ rejects the query.
- **Re-encryption queries.** Suppose that $\mathcal{A}$ queries $\mathcal{C}$ about $(sk_{\mathsf{ID}'_i}, C_{\mathsf{ID}_i}, \mathsf{ID}_i \rightarrow \mathsf{ID}'_i)$. If $sk_{\mathsf{ID}'_i}$ has never issued to $\mathcal{A}$ then $\mathcal{C}$ rejects the query. Otherwise, $\mathcal{C}$ runs **Check** with the input $(parms, C_{\mathsf{ID}_i}, \mathsf{ID}_i)$.
  - If **Check** outputs 0 then $\mathcal{C}$ rejects the query.
  - Otherwise, $\mathcal{C}$ generates $e_{\mathsf{ID}'_i}$ by running **EGen** with $sk_{\mathsf{ID}'_i}$ as an input. $\mathcal{C}$ generates $rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i}$ by running $\mathbf{KeyGen}_{\mathsf{PRO}}$ with the input $e_{\mathsf{ID}'_i}$. $\mathcal{C}$ re-encrypts C_{ID_i} into $C_{\mathsf{ID}'_i}$ by running **ReEnc** with the input $rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i}$. $\mathcal{C}$ returns $C_{\mathsf{ID}'_i}$ to $\mathcal{A}$.

Challenge. After some queries, $\mathcal{A}$ selects two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and a target identity ID^* which no secret key for ID^* has issued, and sends them to $\mathcal{C}$. Given $(M_0, M_1, \mathsf{ID}^*)$, $\mathcal{C}$ selects $d \overset{R}{\leftarrow} \{0,1\}$ and computes

$$C_{\mathsf{ID}^*_d} = \mathbf{Enc}_{\mathsf{IBE}}(\mathsf{ID}^*, parms, M_d).$$

$\mathcal{C}$ returns $C_{\mathsf{ID}^*_d}$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ continues to issue queries as in Phase 1, and $\mathcal{C}$ responds as before except the following case.

- If $\mathcal{A}$ makes the secret key query at the point ID^*, then $\mathcal{C}$ rejects.
- If $\mathcal{A}$ makes the re-encryption query such that $\mathsf{ID}_i = \mathsf{ID}^*$, then $\mathcal{C}$ rejects.

Guess. Finally, $\mathcal{A}$ outputs a guess $d' \in \{0,1\}$.

The adversary $\mathcal{A}$ wins if $d' = d$. An identity-based proxy re-encryption system is secure in the sense of IND-ID-CPA if $|\Pr[d' = d] - 1/2|$ is negligible.

Definition 5. *Let $\mathcal{A}$ be an adversary against the identity-based proxy re-encryption system. Define the IND-ID-CPA advantage of $\mathcal{A}$ as follows.*

$$\mathtt{Adv}^{id}_{\mathsf{ibp}}(\mathcal{A}) = 2(\Pr[d' = d] - 1/2). \tag{4}$$

We say that an identity-based proxy re-encryption system is (k, t, q, ϵ) adaptive chosen plaintext secure if for any t time IND-ID-CPA adversary $\mathcal{A}$ that makes at most q chosen queries under a security parameter k we have that $\mathtt{Adv}^{id}_{\mathsf{ibp}}(\mathcal{A}) < \epsilon$. As shorthand, we say that an identity-based proxy re-encryption system is (k, t, q, ϵ) IND-ID-CPA secure.

We define the selective adversary who is identical to the above adversary except that it discloses to the challenger the target identity ID^* before the setup. This yields that the adversary might make queries for ID^* in Phase 1. We denote the selective IND-ID-CPA by IND-sID-CPA and the advantage of the selective adversary by $\mathtt{Adv}^{sid}_{\mathsf{ibp}}$. The definition is as same as that of Definition. 5.

4.3 Construction

Let the PKG employ a digital signature scheme $(\mathbf{KeyGen}_{\Sigma}, \mathbf{Sign}, \mathbf{Verify})$. Our identity-based proxy re-encryption system involves BB-IBE system described in Sec. 3.3 and the following algorithms.

– *The delegation system:*
  EGen$(sk_{\mathsf{ID}}, parms)$. Given $sk_{\mathsf{ID}} = (d_0, d_1) = (g_2^{\alpha}(g_1^{\mathsf{ID}}h)^u, g^u)$ with $parms$, set $e_{\mathsf{ID}} = d_1$.
  KeyGen$_{\mathsf{RKG}}(mk, parms)$. Given $mk = \alpha$ with $parms$, set $sk_R = \alpha$.
  KeyGen$_{\mathsf{PRO}}(sk_R, e_{\mathsf{ID}'}, parms, \mathsf{ID}, \mathsf{ID}')$. Given $sk_R = \alpha$, $e_{\mathsf{ID}'} = g^{u'}$ with $parms$, set $rk_{\mathsf{ID}\to\mathsf{ID}'} = (\mathsf{ID} \to \mathsf{ID}', g^{u'\alpha})$.
  ReEnc$(rk_{\mathsf{ID}\to\mathsf{ID}'}, parms, C_{\mathsf{ID}}, \mathsf{ID}, \mathsf{ID}')$. Given the delegator's identity ID, the delegatee's identity ID', $rk_{\mathsf{ID}\to\mathsf{ID}'} = (\mathsf{ID} \to \mathsf{ID}', g^{u'\alpha})$, $C_{\mathsf{ID}} = (C_1, C_2, C_3)$ with $parms$, re-encrypt the ciphertext C_{ID} into $C_{\mathsf{ID}'}$ as follows.

$$C_{\mathsf{ID}'} = (C_1', C_2', C_3') = \left(C_1, C_2, C_3\hat{e}(C_1^{\mathsf{ID}'-\mathsf{ID}}, g^{u'\alpha})\right) \in \mathbb{G}^2 \times \mathbb{G}_1.$$

  Check$(parms, C_{\mathsf{ID}}, \mathsf{ID})$. Given the delegator's identity ID and $C_{\mathsf{ID}} = (C_1, C_2, C_3)$ with $parms$, compute $v_0 = \hat{e}(C_1, g_1^{\mathsf{ID}}h)$ and $v_1 = \hat{e}(C_2, g)$. If $v_0 = v_1$ then output 1. Otherwise output 0.

In this system, we let the PKG set $mk = \alpha$ while $mk = g_2^{\alpha}$ described in Sec. 3.3.

Remark: We consider the case that a malicious player modifies a target ciphertext $C_{\mathsf{ID}^*} = (g^{r^*}, (g_1^{\mathsf{ID}^*}h)^{r^*}, M^*\hat{e}(g_1, g_2)^{r^*})$ into the different ciphertext for ID (for short C_{ID}) such that she can derive some information about M^* from the underlying message of C_{ID} by utilizing the proxy as an oracle. The algorithm **Check** prevents such modification where $\mathsf{ID} \neq \mathsf{ID}^*$ because passing through the check implies that C_{ID} is the form of $C_{\mathsf{ID}} = (g^r, (g_1^{\mathsf{ID}}h)^r, M\hat{e}(g_1, g_2)^r)$, and it is obviously hard to make such modification since the underlying BB-IBE is semantically secure.

4.4 Security Analysis

In this section, we show that the proposed system is semantically secure.

Theorem 2. *Suppose that the (k, t, ϵ)-dBDH assumption holds and the PKG's digital signature scheme is (k, t', q, ϵ')-strong existentially unforgeable. Then the identity-based proxy re-encryption system is (k, t'', q, ϵ'') IND-sID-CPA secure for any q, k, $\epsilon'' \leq \epsilon + \epsilon'$, and $t'' + t' < t - \Theta(\tau_e q + \tau_s q + \tau_v q)$ where τ_e is the maximum time for an exponentiation in $\mathbb{G}$, τ_s is the maximum time for running* **Sgin**, *and τ_v is the maximum time for running* **Verify**.

Proof. Let $\mathcal{A}$ be an adversary against the identity-based proxy re-encryption system in the IND-sID-CPA sense. We construct an adversary $\mathcal{B}$ which solves the dBDH problem in $\mathbb{G}$ by utilizing $\mathcal{A}$. Providing that $\mathcal{B}$ is given an input $(g, \Gamma_1, \Gamma_2, \Gamma_3, X) = (g, g^a, g^b, g^c, X)$, where $X = \hat{e}(g,g)^{abc}$ or $X = R \stackrel{R}{\leftarrow} \mathbb{G}_1$. We describe how $\mathcal{B}$ works in the following.

Initialization. The selective identity game begins with $\mathcal{A}$ first outputting a target identity ID^*. $\mathcal{B}$ generates two blank lists SKL and RKL.

Setup. $\mathcal{B}$ selects a (k, t', q, ϵ')-strong existentially unforgeable digital signature scheme $(\mathbf{KeyGen}_\Sigma, \mathbf{Sign}, \mathbf{Verify})$ and generates (sk_Σ, vk_Σ) by running $\mathbf{KeyGen}_\Sigma$. To generate the system parameters, algorithm $\mathcal{B}$ picks $z \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and sets $g_1 = \Gamma_1$, $g_2 = \Gamma_2$, $h = g_1^{-\mathsf{ID}^*} g^z$, and $parms = (g, g_1, g_2, h)$. $\mathcal{B}$ gives $(parms, vk_\Sigma)$ to $\mathcal{A}$. Note that the corresponding PKG's master-secret key, which is unknown to $\mathcal{B}$, is $g_2^a = g^{ab} \in \mathbb{G}$.

Phase 1. Given $parms$ and vk_Σ, $\mathcal{A}$ asks some queries to the challenger. When $\mathcal{A}$ queries the challenger, $\mathcal{B}$ works as follows.

- **Secret key queries.** Suppose that $\mathcal{A}$ queries the challenger at a point ID_i. If $\mathsf{ID}_i = \mathsf{ID}^*$ then $\mathcal{B}$ rejects the query. Otherwise, $\mathcal{B}$ selects $r_i \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, and sets

$$sk_{\mathsf{ID}_i} = (d_0, d_1) = \left(g_2^{\frac{-z}{\mathsf{ID}_i - \mathsf{ID}^*}} (g_1^{\mathsf{ID}_i - \mathsf{ID}^*} g^z)^{r_i}, g_2^{\frac{-1}{\mathsf{ID}_i - \mathsf{ID}^*}} g^{r_i}\right)$$

  and $e_{\mathsf{ID}_i} = d_1$. $\mathcal{B}$ computes $\mathbf{Sign}(sk_\Sigma, \mathsf{ID}_i || e_{\mathsf{ID}_i}) = \sigma_{e_i}$, and returns $(sk_{\mathsf{ID}_i}, \mathsf{ID}_i || e_{\mathsf{ID}_i}, \sigma_{e_i})$ to $\mathcal{A}$. $\mathcal{B}$ adds the query and the answer to the list SKL.
- **Type-1 re-encryption key queries.** When $\mathcal{A}$ queries the challenger about $\mathsf{ID}_i \rightarrow \mathsf{ID}'_i$, $\mathcal{B}$ selects $r'_i \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, sets $rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i} = (\mathsf{ID}_i \rightarrow \mathsf{ID}'_i, g_1^{r'_i})$ and $e_{\mathsf{ID}'_i} = g^{r'_i}$. $\mathcal{B}$ generates a signature $\sigma_{e'_i}$ for $\mathsf{ID}'_i || e_{\mathsf{ID}'_i}$ by running $\mathbf{Sign}$. $\mathcal{B}$ returns $rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i}$ with $(\mathsf{ID}'_i || e_{\mathsf{ID}'_i}, \sigma_{e'_i})$ to $\mathcal{A}$. $\mathcal{B}$ adds $(rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i}, \mathsf{ID}'_i || e_{\mathsf{ID}'_i}, \sigma_{e'_i})$ to the list RKL.
- **Type-2 re-encryption key queries.** Suppose that $\mathcal{A}$ queries the challenger about $(\mathsf{ID}_i \rightarrow \mathsf{ID}'_i, \mathsf{ID}'_i || e_{\mathsf{ID}'_i}, \sigma_{e'_i})$. If the IBE secret key $sk_{\mathsf{ID}'_i}$ corresponding to $(\mathsf{ID}'_i || e_{\mathsf{ID}'_i}, \sigma_{e'_i})$ is in the list SKL then $\mathcal{B}$ rejects the query. Otherwise,
  - If $rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i}$ corresponding to $(\mathsf{ID}'_i || e_{\mathsf{ID}'_i}, \sigma_{e'_i})$ is in the list RKL then $\mathcal{B}$ returns $rk_{\mathsf{ID}_i \rightarrow \mathsf{ID}'_i}$ to $\mathcal{A}$.
  - Otherwise, $\mathcal{B}$ computes $v = \mathbf{Verify}(vk_\Sigma, \mathsf{ID}'_i || e_{\mathsf{ID}'_i}, \sigma_{e'_i})$.
    - If $v = 1$ then $\mathcal{B}$ halts.
    - Otherwise, $\mathcal{B}$ rejects the query.
- **Re-encryption queries.** Suppose that $\mathcal{A}$ queries $\mathcal{C}$ about $(sk_{\mathsf{ID}'_i}, C_{\mathsf{ID}_i}, \mathsf{ID}_i \rightarrow \mathsf{ID}'_i)$ where $C_{\mathsf{ID}_i} = (C_1, C_2, C_3)$. If $sk_{\mathsf{ID}'_i}$ is not in the list SKL or $\mathsf{ID}_i = \mathsf{ID}^*$ then $\mathcal{B}$ rejects the query. Otherwise $\mathcal{B}$ computes $v_0 = \hat{e}(C_1, g_1^{\mathsf{ID}_i} h)$ and $v_1 = \hat{e}(C_2, g)$.

- If $v_0 \neq v_1$ then $\mathcal{B}$ rejects the query.
- Otherwise, $\mathcal{B}$ generates a secret key $sk_{\mathsf{ID}_i} = (d_0, d_1)$ as the way of that at **Secret key queries**. $\mathcal{B}$ decrypts C_{ID_i} to obtain the plaintext M_i by using sk_{ID_i}. $\mathcal{B}$ sets $C_{\mathsf{ID}'_i} = (C_1, C_2, M_i\hat{e}(C_1, d'_0)/\hat{e}(C_2, d'_1))$ where d'_0 and d'_1 are components of $sk_{\mathsf{ID}'_i}$. $\mathcal{B}$ returns $C_{\mathsf{ID}'_i}$ to $\mathcal{A}$.

Challenge. After some queries, $\mathcal{A}$ selects two equal length plaintexts $M_0, M_1 \in \mathcal{M}$. Given (M_0, M_1), $\mathcal{B}$ selects $d \stackrel{R}{\leftarrow} \{0,1\}$ and sets

$$C_{\mathsf{ID}^*_d} = (\Gamma_3, \Gamma_3^z, M_d X).$$

$\mathcal{B}$ returns $C_{\mathsf{ID}^*_d}$ to $\mathcal{A}$. Notice that if $X = \hat{e}(g,g)^{abc} = \hat{e}(g_1, g_2)^c$ then $C_{\mathsf{ID}^*_d}$ is a valid encryption of M_d. On the other hand, if X is uniform and independent in $\mathbb{G}_1$ then $C_{\mathsf{ID}^*_d}$ is independent of d in the adversary's view.

Phase 2. $\mathcal{A}$ continues to issue queries as in Phase 1, and $\mathcal{B}$ responds as before.

Solve. Finally, $\mathcal{A}$ outputs a guess $d' \in \{0,1\}$. $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $d' = d$ then $\mathcal{B}$ outputs 1 meaning $X = \hat{e}(g,g)^{abc}$. Otherwise, it outputs 0 meaning $X = R$.

We claim that $\mathcal{B}$ generates the valid secret key sk_{ID_i} for ID_i. To see this, let $\tilde{u}_i = r_i - \frac{b}{\mathsf{ID}_i - \mathsf{ID}^*}$. Then we have that

$$\begin{aligned}
sk_{\mathsf{ID}_i} = (d_0, d_1) &= \left(g_2^{\frac{-z}{\mathsf{ID}_i - \mathsf{ID}^*}} (g_1^{\mathsf{ID}_i - \mathsf{ID}^*} g^z)^{r_i}, g_2^{\frac{-1}{\mathsf{ID}_i - \mathsf{ID}^*}} g^{r_i} \right) \\
&= \left(\frac{g_2^a (g_1^{\mathsf{ID}_i - \mathsf{ID}^*} g^z)^{r_i}}{(g_1^{\mathsf{ID}_i - \mathsf{ID}^*} g^z)^{\frac{b}{\mathsf{ID}_i - \mathsf{ID}^*}}}, g^{r_i - \frac{b}{\mathsf{ID}_i - \mathsf{ID}^*}} \right) \\
&= \left(g_2^a (g_1^{\mathsf{ID}_i - \mathsf{ID}^*} g^z)^{r_i - \frac{b}{\mathsf{ID}_i - \mathsf{ID}^*}}, g^{r_i - \frac{b}{\mathsf{ID}_i - \mathsf{ID}^*}} \right) \\
&= \left(g_2^a (g_1^{\mathsf{ID}_i} h)^{\tilde{u}_i}, g^{\tilde{u}_i} \right)
\end{aligned}$$

We also claim that $\mathcal{B}$ can perfectly simulate the re-encryption key for ID_i since it looks random and independent of any other values if the adversary does not obtain the corresponding IBE secret key for ID_i.

$\mathcal{B}$ fails to simulate the challenger if $\mathcal{B}$ halts in the Type-2 re-encryption key query. Otherwise $\mathcal{B}$ perfectly simulates the challenger. The maximum probability of that $\mathcal{B}$ halts is obviously upper-bounded by $\mathtt{Adv}^{eu}(\mathcal{A})$. Therefore, we conclude the theorem.

4.5 Toward the Chosen Ciphertext Security

Green and Ateniese [GA06] proposed the semantically secure identity-based proxy re-encryption system and constructed the CCA-secure system based on the former system, applying CHK conversion [CHK04] to it. It might be able to construct the CCA-secure system based on our proposed system by using the same technique. It is the further study.

5 Conclusion

In this study, we proposed two proxy re-encryption systems; one for the decryption right delegation from a PKE user to IBE users, and the other one for the delegation among IBE users. The former is the first "hybrid "proxy re-encryption system, and the latter has some advantage over the previously proposed identity-based systems. We introduced the security notion and proved that both our systems are semantically secure based on the dBDH assumption, in the standard model. We presented neither a hybrid system nor an identity-based system secure in the CCA sense. This is the further study.

Acknowledgements

We would like to thank Dan Boneh, Eu-Jin Goh and the anonymous reviewers of Pairing 2007 for giving helpful suggestions.

References

[AFGH05] Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: Proceedings of the 12th Annual Network and Distributed System Security Symposium - NDSS'05, pp. 83–107 (2005)

[BB04a] Boneh, D., Boyen, X.: Efficient selective-id secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[BB04b] 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)

[BBS98] Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)

[BF01] Boneh, D., Franklin, M K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

[CHK03] Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) Advances in Cryptology – EUROCRPYT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)

[CHK04] 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)

[DI03] Dodis, Y., Ivan, A.: Proxy cryptography revisited. In: Proceedings of the 10th Annual Network and Distributed System Security Symposium - NDSS'03 (2003)

[G06] Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)

[GMR88] Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Computing 17(2), 281–301 (1988)

[GA06] Green, M., Ateniese, G.: Identity-Based Proxy Re-Encryption http://eprint.iacr.org/2006/473

[GS04] Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security, pp. 548–566. Springer, Heidelberg (2002)

[J99] 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)

[MO97] Mambo, M., Okamoto, E.: Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. IEICE Trans. Fund. Electronics Communications and Computer Science E80-A/1, 54–63 (1997)

[S84] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–52. Springer, Heidelberg (1985)

[W05] Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

[ZMSR04] Zbou, L., Marsh, M.A., Schneider, F.B., Redz, A.: Distributed blinding for ElGamal re-encryption, Technical Report 2004-1924, Cornell Computer Science Department (2004)

A Lemma 1

Boneh and Boyen [BB04a] proved BB-IBE system being semantically secure if both components of secret key remains secret; however we can prove that the disclosure of second component of BB-IBE secret key does not make BB-IBE system weak.

Lemma 1. *Suppose that the BB-IBE system is* (k, t, q, ϵ) *selective-identity, adaptive chosen plaintext (IND-sID-CPA) secure, then, for any* q*,* k*, and* $t' < t - \Theta(\tau q)$*, BB-IBE system is* (k, t', q, ϵ) *IND-sID-CPA secure against the adversary specified in Sec. 2.5 but it obtains the second component of IBE secret key for any identity it selects where* τ *is the maximum time for an exponentiation in* $\mathbb{G}$*.*

Proof. Let $\mathcal{A}$ be an IND-sID-CPA adversary specified in Sec. 2.5 but it obtains the second component of IBE secret key for any identity it selects. We construct an original IND-sID-CPA adversary $\mathcal{B}$ by utilizing $\mathcal{A}$. We describe how $\mathcal{B}$ works in the following.

Initialization. When $\mathcal{A}$ selects the target identity ID^*, $\mathcal{B}$ forwards it to its challenger. $\mathcal{B}$ selects a random $u \in \mathbb{Z}_p^*$ and gives g^u to $\mathcal{A}$ whenever $\mathcal{A}$ requests the second component of the secret key for ID^*.

Setup. On receiving public parameters *parms* from the challenger, $\mathcal{B}$ forwards it to $\mathcal{A}$.

Phase 1. When $\mathcal{A}$ requests the secret key for ID, $\mathcal{B}$ forwards the request to the challenger. $\mathcal{B}$ obtains the corresponding secret key and forwards it to $\mathcal{A}$. When $\mathcal{A}$ requests the second component of secret key for ID, $\mathcal{B}$ requests the secret key for ID from the challenger. $\mathcal{B}$ obtains the corresponding secret key, then forwards its second component to $\mathcal{A}$.

Challenge. When $\mathcal{A}$ selects (M_0, M_1), $\mathcal{B}$ forwards it to the challenger. $\mathcal{B}$ obtains the challenge ciphertext and forwards it to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ continues to issue queries as in Phase 1 with the restriction that $\mathcal{A}$ cannot issue secret key queries for ID^*. $\mathcal{B}$ responds the queries as in Phase 1.

Guess. If $\mathcal{A}$ outputs a guess $\tilde{b} \in \{0, 1\}$, $\mathcal{B}$ outputs $\tilde{b}$.

It is obvious that $\mathcal{B}$ wins the game whenever $\mathcal{A}$ wins. We conclude the proof.

Fair Blind Signatures Revisited

Emeline Hufschmitt and Jacques Traoré

France Telecom Research and Development,
42 rue des Coutures, BP 6243, F-14066 Caen cedex
{emeline.hufschmitt,jacques.traore}@orange-ftgroup.com

Abstract. This paper presents a formal model for fair blind signature schemes and a provably secure scheme based on bilinear maps. A blind signature scheme is a protocol for obtaining a signature on a message which is unknown from the signer. Furthermore, the signer cannot link his transcript of a protocol to the resulting message-signature pair. Fair blind signatures were introduced by Stadler *et al.* at *Eurocrypt'95* in [37]. A fair blind signature scheme is a blind signature scheme allowing two types of blindness revocation: link a signature to the session which conducted this signature (Session Tracing) or, conversely, identify a signature knowing a signing session (Signature Tracing). Various fair blind signature schemes have been proposed in the past years, but none of them presents a secure fair blind signature scheme that allows polynomially many signatures to be securely issued, even if Abe *et al.*'s claimed it in [3]. In this paper, we first show a flaw in the blindness of most (fair) blind signature schemes where the signer is able to link signatures if he chooses his keys in an appropriate way. Then, we show a flaw in the proof of unforgeability of Abe *et al.*' scheme and propose a stronger security model than theirs. It possesses all the needed properties for fair blind signature schemes: blindness, traceability and non frameability for both revocations (the one-more unforgeability is implied by these properties). Finally, we describe a new fair blind signature scheme based on bilinear maps. This scheme thwarts the flaw against previous blind signatures and is proved secure in the random oracle model with respect to our model.

Keywords: Blind signatures, Anonymity Revocation, Security Model, electronic voting.

1 Introduction

The concept of blind signature was introduced by Chaum in 1982 [14]. The goal of a blind signature protocol is to enable a user to obtain a signature from a signer so that the signer does not learn any information about the message he signed. It also assures that the user cannot obtain more than l valid signatures after l interactions with the signer, even if the signatures are issued in an adaptive and concurrent manner (making *parallel attacks* impossible). The security of blind schemes have been formalized in [24] and [34]. In [24], the authors proved the existence of secure blind signatures assuming the one-way trapdoor family. Unfortunately, their

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 268–292, 2007.

construction is only theoretical. Practical and secure blind signature schemes have been proposed by Pointcheval and Stern ([34], [35]). In particular, they show that blind signature schemes coming from witness-indistinguishable protocols (such as the Okamoto version [30] of the Schnorr and Guillou-Quisquater identification schemes) are secure in the random oracle model, as long as the number l of issued signatures is polylogarithmically bounded (i.e., $l \leq (\log k)^c$, where k is a security parameter and c a constant).

In [33], Pointcheval presents a generic transformation that renders schemes restricted to issue logarithmically many signatures (such as [35]) into stronger ones that can securely issue polynomially many signatures, at the cost of two extra data moves. As the underlying schemes require three data moves, the resulting schemes need five moves of data between the signer and the user. Moreover, these modified schemes are proved secure against restricted forms of *parallel attacks*.

Those signatures are used in electronic voting systems (*e.g.* the Votopia system [21]) and e-cash [14]. But in a blind scheme, there is no way of going back once the signature has been issued. For some applications (such as e-cash), we need to be able to trace the signatures or the users to avoid frauds (money-laundering, black mailing, ...). This is where fair blind signatures came up.

Fair blind signature schemes (FBSS for short) were introduced by Stadler *et al.* at *Eurocrypt'95* in [37]. These schemes involve four types of protagonists: a user $\mathcal{U}$, a signer $\mathcal{S}$, a Revocation Authority RA and a Judge J. If needed, the Revocation Authority is able to revoke the blindness in two ways:

- given a transcript of a signature issuing session conducted with an authenticated user, the authority can identify the resulting signature (Signature Tracing), or
- given a signature, the authority can identify the issuing session that yielded the signature, which eventually identifies the user who conducted the session (Session or Identity Tracing[1]).

The Revocation Authority also produces a proof of his claims to the Judge. The Judge can be summoned to testify the validity of the revocations.

Various FBSS have been proposed so far with direct applications to fair e-cash ([12], [22]) and to electronic voting [13]. Stadler *et al.* presented in [37] three schemes using the Cut and Choose method or Oblivious Transfer, but this led to inefficient schemes and the second scheme was proved insecure thereafter [38]. Brickell *et al.* proposed the same year an anonymous electronic payment scheme using fair blind signatures [9], but their construction was not efficient either. Other schemes ([20], [16]) have been proposed but they do not provide formal proofs.

More recently, new blind signature schemes have been proposed. In [10], Camenisch *et al.* present the first efficient blind signature scheme secure in the standard model, but proven unforgeable only for the case of sequential attacks. In

[1] Depending on the applications, one might find more convenient to recover the user's identity immediately from the signature.

[29], Okamoto presents a new efficient signature scheme based on bilinear maps. He then turns it into a partially blind signature scheme and proves its security under the 2SDH assumption (introduced in his article). The proofs are made without random oracles. Kiayias *et al.* also present in [26] a new blind scheme which is also proven secure without random oracle, in the common reference string model. The security of their scheme is based on the LRSW assumption (introduced in [28]). They also turn the scheme into a partially blind signature scheme. Two generic constructions for blind signatures have been proposed recently ([18], [23]) and the two articles present generic schemes and formal models for blind signatures. Our aim is to work on a variant of blind signatures, the so-called FBSS, and none of these schemes fits the definition of FBSS.

The most interesting scheme concerning FBSS was proposed by Abe and Ohkubo [3] and is based on the blind signature scheme of Abe [1]. This scheme is efficient and its security relies on the discrete logarithm problem. Abe and Ohkubo present in this article, the first security model for FBSS and attempt to prove in the random oracle model that their scheme allows a polynomial number of signatures to be securely issued. However this model does not possess all the requirements of FBBS (such as non Frameability) and some definitions are incomplete[2]. To overcome these defects, we define a new security model.

Our Contribution: In this article, we design an attack against (fair) blind signature schemes based on Schnorr's authentication scheme [36]. We also show that Abe and Ohkubo failed to prove the claimed polynomial security of their scheme. We then present a new security model for FBSS which is an improvement of Abe and Ohkubo's model [3]. It defines all the needed properties for FBSS and details the attacks conducted by the attacker. We looked at traceable signatures (introduced in [25]) and group signatures security model (as proposed in [7]) to formalize our definitions. But these two models are not meant for fair blind signatures and this is why we need to redefine some of the properties and add the missing ones. In the perspective of presenting a secure FBSS we then describe a new scheme using bilinear maps and prove its security with respect to our model. This scheme is not generic but is quite efficient and its security is based on a formal model, and holds even when multiple executions of the protocol are performed concurrently (*i.e.* in an arbitrarily-interleaved manner). We rely on the join protocol of Boneh *et al.* group signature scheme [8] to build our system. This implies that our security is based on the q-SDH assumption [8].

Organization of the Paper: In Section 2, we describe the flaw of most of the previous (fair) blind signature schemes. In Section 3, we explain why the security analysis (proof of unforgeability) of [3] is wrong. In Section 4, we recall some mathematical tools and cryptographic protocols useful to our construction. In Section 5, we present the security model for fair blind signature schemes. In Section 6, we present our new fair blind signature scheme. Finally, we briefly describe in Section 7 the security analysis of our scheme. We detail the proofs of security in the appendix.

[2] For example, they do not give the access to adversary to several oracles.

2 A Flaw in Blind Signature Schemes

In this part, we see how it is possible for a signer to link signatures by choosing his keys in a particular way. We will use the Schnorr's blind signature scheme for our attack, but this works for any Schnorr like (fair) blind signature scheme (see [31] for the constructions). This kind of attack has been studied for ElGamal based schemes ([27]) but not blind schemes.

In a normal execution of the protocol, the signer's keys are (x, h) where $x \in \mathbb{Z}_q^*$ and $h = g^x \bmod p$ (p is a prime number assumed to be chosen at random so that $p-1$ as a large prime factor q). In this attack, we assume the signer is dishonest and he chooses his keys in a special way. He first chooses $x \in \mathbb{Z}_q^*$. Then he computes its modified public key as $h := \beta g^x \bmod p$ where g is of order q mod p and β is of small order mod p.

Aim of the Signer. During this attack the signer is able to link the signatures he produced to the corresponding requesting users. We describe it with two users.

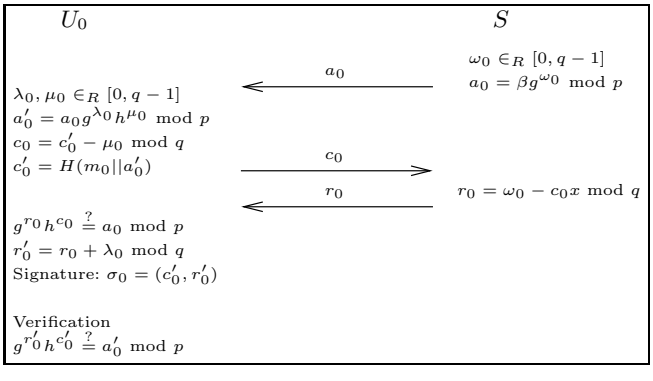


Fig. 1. Schnorr Protocol with dishonest signer - User U_0

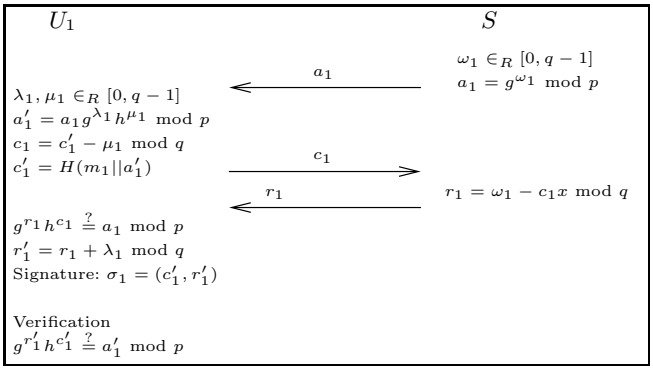


Fig. 2. Schnorr Protocol with dishonest signer - User U_1

In a first part, the signer $\mathcal{S}$ interacts with the user U_0 and with non negligible probability the verification algorithm succeed. Then, the signer interacts with the user U_1 and this time too, the probability of the verification algorithm to succeed is not negligible. Then the signer seeing the two blind signatures is able to link the signatures to the users.

Description of the Attack. In a first part, the signer conducts his attack with the user U_0. The protocol is played as described in figure 1.

To succeed, the signer need the verification equation of the user and the verification protocol to be correct. For simplicity, we omit the subscripts.

$$g^r h^c = (g^{\omega - cx}) h^c \bmod p = g^\omega \beta^c \bmod p = \beta g^\omega \qquad \text{if } c \equiv 1 \bmod ord(\beta) = a$$
$$g^{r'} h^{c'} = g^{r+\lambda} h^{c+\mu} \bmod p = g^\lambda h^\mu g^\omega \beta^c \bmod p = a' \qquad \text{if } c \equiv 1 \bmod ord(\beta)$$

His probability of success is of $1/(ord(\beta))$.

For the user U_1, the protocol is described in figure 2. This time, the signer is playing the real protocol, but still with his modified keys. As previously, his probability of success is: $1/(ord(\beta))$.

When seeing the two signatures σ_0, σ_1, the signer is able to distinguish them by computing $a_i'^q \bmod p$. If it is equal to 1, he knows that the signature belongs to U_1 otherwise it is to U_0. This attack succeeds with probability $(1/ord(\beta))^2$.

To avoid this attack the user need to check whether if $h^q \equiv 1 \bmod p$ when he receives h.

3 Abe and Ohkubo Scheme

In this section, we show a flaw in the security proof of one-more unforgeability of [3][3]. In fact, the proof of one-more unforgeability in [3] relies on the one described in [1]. So, we will only explain why the latter is wrong.

The situation is as follows. Suppose that we have an adversary that is successful in one-more forgery. During its attack, it will receive a sequence of challenges, say $D = (d_1, \ldots, d_n)$. Assume that if we launch a rewinding simulation with randomly chosen D and D', we get a "collision" which allows us to solve the discrete logarithm problem. But, suppose that what we can do in our simulation is to alter only one element of D to generate D' due to some technical reason.

Let D_i' be a sequence that differs only in the i-th position in D. That is, $D_i' = (d_1, \ldots, d_i', \ldots, d_n)$ where d_i' is randomly chosen. In [1] Abe claimed that such D_i' and D still result in a collusion, but it was not correct. It might exist an adversary that is successful in one-more forgery for sufficiently many D such that, for any i and any D_i, D and D_i' will not yield a collision, *i.e.*, the adversary fails in one-more forgery with D_i'. Such a bad case does not happen when the success probability of the adversary, say ε, is 1. But it might be the case if ε is for example $< 1/2$. Anyway, a proof that only denies the existence of such a highly efficient adversary is not significant enough to state security. The mistake comes

[3] Due to the lack of space, we do not describe the scheme.

from the evaluation of success probability of his reduction (case 2 of lemma 2 in [1]). In this case 2 of lemma 2, Abe evaluates the probability P for the adversary to output two valid signatures (with distinct δ values[4]) when it has received D and D'_i. He claimed that this probability is not negligible (and if P was not negligible, we could effectively solve the discrete logarithm problem) (see case 2 of lemma 2 in [1] for more details). However, he only proves the following fact: randomly chosen $d_1, d_2, \ldots, d_{i-1}, d_{i+1}, \ldots, d_n$ satisfies:

$P := \Pr_{d'_i}[D$ and D'_i lead to two valid signatures with distinct δ values$] \geq \psi_{max}$ with probability $\geq 1 - \psi_{max}$ (see [1] for the signification of ψ_{max}). Since in case 2 of lemma 2, ψ_{max} is assumed to be negligible, the above relation is of no help to conclude that P is not negligible.

The fact that the main theorem (theorem 3 and especially lemma 2) is incorrect has been confirmed by Abe himself in [2]. He also informed us that he did not find a fix for this problem. However, since his scheme is witness indistinguishable, a proof from the full version of [4] applies to this variant, though it only states poly-logarithmic security.

4 Preliminaries

4.1 Notation and Mathematical Tools

- Proofs of knowledge: we note $Pok(\alpha : f(\alpha, \ldots))$ the proof of knowledge of a value α that satisfies the predicate f. We also use signatures of knowledge which are derived from proofs of knowledge, using the Fiat-Shamir heuristic ([17]).
- Bilinear maps: the bilinear maps used in cryptography are the Weil and Tate pairings on some elliptic curves. Let $\mathbb{G}_1$and $\mathbb{G}_2$ be two groups of order some large prime q, denoted multiplicatively and $\mathbb{G}_T$ a cyclic multiplicative group with the same order.

  An *admissible bilinear map* is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ that is:
  - bilinear: $e(g^a, h^b) = e(g, h)^{ab}$ for all $(g, h) \in (\mathbb{G}_1, \mathbb{G}_2)$ and all $a, b \in \mathbb{Z}$,
  - non-degenerate: for g and h two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, we have $e(g, h) \neq 1$,
  - computable: there exists an efficient algorithm to compute $e(g, h)$ for any $(g, h) \in (\mathbb{G}_1, \mathbb{G}_2)$.

4.2 Double ElGamal Encryption

The ElGamal encryption is an IND-CPA scheme but it is known that by double encrypting the same message under an IND-CPA scheme and using a simulation-sound proof of equality of plaintext we obtain an IND-CCA2 scheme. In our scheme, we need an IND-CCA2 scheme, and we will use the double ElGamal encryption. This double encryption was proved IND-CCA2 in [19].

[4] See [1] for the signification of δ.

Let p be a strong prime, such that $q|p-1$ is also a large prime, and g be an element of $\mathbb{Z}_p^*$ of order q. We denote by G, the subgroup of $\mathbb{Z}_p^*$ of elements of order q. For the double ElGamal encryption, the private and public keys are $(x,\ y) \in \mathbb{Z}_q^2, (h_1 = g^x, h_2 = g^y)$. The double ElGamal encryption is two encryptions of the message $\mathtt{Encrypt}_{pk}^{\mathtt{2EG}}(m) = (g^r, m.h_1^r, g^s, m.h_2^s) = (T_1, T_2, T_3, T_4)$ along with a simulation-sound proof of equality of plaintexts. The decryption is given by $\mathtt{Decrypt}_{sk}^{\mathtt{2EG}}(T_1, T_2, T_3, T_4) = T_2.(T_1^x)^{-1} = T_4.(T_3^y)^{-1}$, and the verification that the proof is correct.

4.3 Paillier Encryption and Extractable Commitment

In our scheme, we employ the public-key encryption scheme introduced by Paillier [32]. Let p and q be random primes for which $\mathsf{p}, \mathsf{q} > 2, \mathsf{p} \neq \mathsf{q}, |\mathsf{p}| = |\mathsf{q}|$ and $\gcd(\mathsf{pq}, (\mathsf{p}-1)(\mathsf{q}-1)) = 1$; let $\mathsf{n} = \mathsf{pq}, \pi = lcm(\mathsf{p}-1, \mathsf{q}-1), K = \pi^{-1} \bmod \mathsf{n}$, and $\mathsf{g} = (1+\mathsf{n})$; then the public key is $pk = (\mathsf{n}, \mathsf{g})$ and the secret key is $sk = (\mathsf{p}, \mathsf{q})$.To encrypt $m \in \mathbb{Z}_\mathsf{n}$, the user chooses $r \in_\mathcal{R} \mathbb{Z}_\mathsf{n}^*$ and $M = \mathtt{Encrypt}_{pk}^{\mathtt{Pai}}(m, r)$ is given by $M = \mathsf{g}^m r^\mathsf{n} \bmod \mathsf{n}^2$. The decryption algorithm, $\mathtt{Decrypt}_{sk}^{\mathtt{Pai}}$, is given by $\mathtt{Decrypt}_{sk}^{\mathtt{Pai}}(M) = \frac{(M^{\pi K} \bmod \mathsf{n}^2)-1}{\mathsf{n}} \bmod \mathsf{n} = m$.

Commitment Schemes. A commitment scheme consists of three algorithms:

- A public key generation algorithm $\mathcal{G}en$, $pk \leftarrow \mathcal{G}en$
- A commitment algorithm $\mathcal{C}om$ which is used to produce a commitment on a message m and the decommitment information r, $(c,\ r) \leftarrow \mathcal{C}om_{pk}(m)$
- A decommitment algorithm $\mathcal{D}ecom$ which is used to verify the decommitment information r and the message m with respect to the commitment c, $\{0,1\} \leftarrow \mathcal{D}ecom(c, m, r)$.

A commitment scheme satisfies two properties: hiding, the receiver can not obtain any information about m given $\mathcal{C}om_{pk}(m, r)$, and binding, the committer cannot change his mind about m later.

In an extractable commitment, there is a trapdoor information xk associated to each public key pk that allows the trapdoor owner to compute m from any $\mathcal{C}om_{pk}(m, r)$. Paillier encryption scheme can be used as such an extractable commitment.

4.4 Computational Assumptions

We use the following computational assumptions to prove the security of our scheme.

Definition 1 (The Decisional Composite Residuosity assumption) *There is no p.p.t. distinguisher for n-th residues modulo n^2. In other words, there is no p.p.t. adversary that can distinguish $\mathbb{Z}_{n^2}^n$ from $\mathbb{Z}_{n^2}^*$, where $\mathbb{Z}_{n^2}^n = \{z \in \mathbb{Z}_{n^2}^* | \exists y \in \mathbb{Z}_{n^2}^* : z = y^n \bmod n^2\}$.*

Definition 2 (q-Strong Diffie-Hellman assumption)
Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order some large prime, and ψ be an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Let γ be a random element of $\mathbb{Z}_p$, g_2 be a random generator of $\mathbb{G}_2$ and $g_1 = \psi(g_2) \in \mathbb{G}_2$.

We say that an algorithm $\mathcal{A}$ has advantage ε in solving q-SDH in $(\mathbb{G}_1, \mathbb{G}_2)$ if

$$\Pr\left[\mathcal{A}(g_1, g_2, g_2^{\gamma}, \ldots, g_2^{(\gamma^q)}) = (g_1^{\frac{1}{\gamma+x}}, x)\right] \geq \varepsilon$$

The (q, t, ε)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least ε in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

Definition 3 (The eXternal Diffie-Hellman assumption)
Given three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as well as a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ while the DDH problem is easy in $\mathbb{G}_2$, the XDH assumption states that the DDH problem is hard in $\mathbb{G}_1$.

5 A Model for Dynamic Fair Blind Signature Schemes

In this section, we redefine the properties of Abe *et al.*'s model and add new properties to ensure the security of FBSS. We construct a formal model and describe the experiment for each property. We use the formalism and notations introduced in [7] to achieve our goal.

5.1 Algorithms and Their Usage

- $\mathcal{G}_{\mathcal{S}}$ is a signer key generation algorithm that takes on input the security parameter k and that outputs a private and a public signer key, respectively denoted by sk_S and pk_S: $(sk_S, pk_S) \leftarrow \mathcal{G}_{\mathcal{S}}(1^k)$.
- $\mathcal{G}_{RA}$ is a revocation key generation algorithm that takes on inputs the public key pk_S of a signer and the security parameter, and that outputs a private and a public revocation key, respectively denoted by rsk and rpk: $(rpk, rsk) \leftarrow \mathcal{G}_{RA}(1^k, pk_{\mathcal{S}})$.
- $\mathcal{G}_{\mathcal{U}}$ is a user key generation algorithm that takes on inputs the public key pk_S of a signer and the identity I_{id} of a user $\mathcal{U}$ and that outputs a private and a public user key respectively denoted by $usk_{I_{id}}$ and $upk_{I_{id}}$[5]: $(upk_{id}, usk_{id}) \leftarrow \mathcal{G}_{\mathcal{U}}(1^k, pk)$.
- $\mathcal{S}ig$ is an interactive protocol between the signer $\mathcal{S}$ that takes on input his private key, and a user $\mathcal{U}$ that takes on input his message m, his private key and his identity I_{id}. If $\mathcal{S}$ accepts the protocol, his output is a transcript $view_i$ of the protocol. If $\mathcal{U}$ accepts the protocol, his output is a couple (m, σ), where σ is a signature on m. Otherwise, the protocol fails.
- $\mathcal{V}f$ is a verification algorithm that takes on inputs a message m, a signature σ and the signer's public key pk_S. It outputs 1 if σ is a valid signature of m with respect to pk_S, and 0 otherwise.

[5] The user public key and his identity can be set to be the same.

- $\mathcal{R}_{sig}$ is a revocation algorithm that takes on inputs the transcript $view_i$ of the signer during the target session and the revocation key rsk, and outputs the signature identifier I_{sig} that identifies the signature yielded from the target session, plus a proof π_1 that the identifier was correctly computed: $(I_{sig}, \pi_1) \leftarrow \mathcal{R}_{sig}(view_i, rsk)$.
- $\mathcal{R}_{id}$ is a revocation algorithm that takes on inputs a target message/signature pair Σ_m and the revocation key rsk, and outputs the identifier I_{id} of the session from which the target signature-message pair has been obtained, plus a proof π_2 that I_{id} was correctly computed: $(I_{id}, \pi_2) \leftarrow \mathcal{R}_{id}(\Sigma_m, rsk)$.
- $\mathcal{M}_{sig}$ (*resp.* $\mathcal{M}_{id}$) is a matching algorithm that examines whether I_{sig} (*resp.* I_{id}) matches a message/signature pair Σ_m (*resp.* an identity I_{id} of a user) or not and if the proof is valid. It outputs 1 if they match and the proof is correct, and 0 if otherwise.

The scheme is specified as a tuple: FBSS = ($\mathcal{G}_S$,$\mathcal{G}_{RA}$, $\mathcal{G}_{\mathcal{U}}$, $\mathcal{S}ig$, $\mathcal{V}f$, $\mathcal{R}_{sig}$,$\mathcal{R}_{id}$, $\mathcal{M}_{sig}$,$\mathcal{M}_{id}$).

5.2 The Oracles

The correctness and security definitions are formulated via experiments in which the attack capabilities of the adversary are modeled by providing him access to some of the oracles. We use the following notation:

- HU is the set of honest identities and CU is the set of corrupted identities,
- Set is the set of message/signature pairs obtained with honest users. We call these signatures honest signatures.
- Trans is the set of transcripts[6].

We now formalize the different oracles:

- AddU(.) is a *add user oracle*. By calling this oracle with argument an identity I_{id}, the adversary adds a new identity. The oracle adds I_{id} to the set HU and computes his public key $upk_{I_{id}}$. His private key, $usk_{I_{id}}$, is kept secret and the adversary is returned $upk_{I_{id}}$.
- CrptU(.,.) is a *corrupt user oracle*. The adversary calls this oracle with argument the identity I_{id} of a user and sets his public key to $upk_{I_{id}}$ and his private key to $usk_{I_{id}}$. The identity is added to the set CU.
- USK(.) is a *user secret key oracle* enabling the adversary to obtain the private key $usk_{I_{id}}$ of an identity I_{id}. This identity is then put in the set CU.
- User(.,.,.) is an *honest user signing oracle*. This oracle is used to simulate an execution of the protocol between a corrupted signer and an honest user. The adversary gives to the oracle an identity I_{id} of an honest user and his public key $upk_{I_{id}}$. If the user accepts the protocol, the adversary is given a transcript of the protocol which is added to Trans, and the tuple (m, σ, I_{id}) is added to Set.

[6] Note that, even if the protocol fails, the associated transcript lies in Trans.

- $\mathsf{Sign}(.,.)$ is a *signing oracle*. We use this oracle to simulate an execution of the protocol between a (corrupted or not) user and an honest signer. This oracle enables the adversary to specify an identity I_{id} and a message m, and then it executes the signing protocol using this identity. If both the signer and the user accept the protocol, then the adversary is given the signature σ corresponding to the identity I_{id} and a transcript of the execution which is added to Trans.
- $\mathsf{Choose}_b(.,.,.,.)$ is a *challenge oracle*. The adversary provides two identities I_{id_0} and I_{id_1} and a message m. The adversary obtains the signature of m of I_{id_b} as long as they define honest users. The oracle records the message and the users to ensure that the adversary does not later call a traceability oracle on them.
- $\mathsf{Tsig}(.)$ (*resp.* $\mathsf{Tid}(.,.)$) is a *signature (*resp. *identity) traceability oracle*. The adversary can call this oracle with arguments a transcript of a session (*resp.* a message/signature pair of a user) to obtain the output of the $\mathcal{R}_{sig}$ (*resp.* $\mathcal{R}_{id}$) algorithm.

5.3 Notions of Security

In this part we define the experiments defining the properties of security of our scheme.

Correctness
The correctness implies that a valid signature σ on a message m with respect to $pk_{\mathcal{S}}$ will always be accepted. The revocation algorithms, given either the signature/message pair or the transcript of the target session must correctly identify the user who performs the signature, or the signature/message pair. We formalize correctness via an experiment involving an adversary. To any fair blind signature scheme FBSS, any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we associate the experiment $\mathbf{Exp}^{\text{corr}}_{FBSS,\mathcal{A}}(k)$; and the advantage is

$$\mathbf{Adv}^{\text{corr}}_{FBSS,\mathcal{A}}(k) = Pr[\mathbf{Exp}^{\text{corr}}_{FBSS,\mathcal{A}}(k) = 1].$$

We say that the FBSS is *correct* if we have $\mathbf{Adv}^{\text{corr}}_{FBSS,\mathcal{A}}(k) = 0$ for any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$.

Experiment $\mathbf{Exp}^{\text{corr}}_{FBSS,\mathcal{A}}(k)$:

$(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow \mathcal{G}_{\mathcal{S}}(1^k); (rpk, rsk) \leftarrow \mathcal{G}_{RA}$
HU $\leftarrow \emptyset$; CU $\leftarrow \emptyset$; Trans $\leftarrow \emptyset$
$(m, Id) \leftarrow \mathcal{A}(pk_{\mathcal{S}}, rpk, \mathsf{AddU})$
if $Id \notin$ HU **then** return 0 **if** $usk = \varepsilon$ **then** return 0
$\sigma \leftarrow \mathcal{S}ig(Id, m, usk)$
if $\mathcal{V}f(pk_{\mathcal{S}}, m, \sigma) = 0$ **then** return 1
if $\forall view_i \in$ Trans, $(I^i_{sig}, \pi^i_1) \leftarrow \mathcal{R}_{Sig}(view_i, rsk)$ and $\mathcal{M}_{Sig}(I^i_{sig}, (m, \sigma), \pi^i_1) = 0$
then return 1
$(I_{id}, \pi_2) \leftarrow \mathcal{R}_{id}(m, \sigma)$
if $I_{id} \neq Id$ or $\mathcal{M}_{id}(I_{id}, i, \pi_2) = 0$ **then** return 1

Blindness

The blindness implies that no one is able to extract valuable information from a message/signature pair or to link two message/signature pairs. To fair blind signature scheme FBSS, any adversary $\mathcal{A}$, a bit $b \in \{0,1\}$ and any $k \in \mathbb{N}$, we associate the experiment $\mathbf{Exp}^{\text{blind}-\text{b}}_{FBSS,\mathcal{A}}(k)$; and the advantage is

$$\mathbf{Adv}^{\text{blind}}_{FBSS,\mathcal{A}}(k) = Pr[\mathbf{Exp}^{\text{blind}-1}_{FBSS,\mathcal{A}}(k) = 1] - Pr[\mathbf{Exp}^{\text{blind}-0}_{FBSS,\mathcal{A}}(k) = 1].$$

We say that the FBSS is *blind* if the function $\mathbf{Adv}^{\text{blind}}_{FBSS,\mathcal{A}}$ is negligible for any polynomial-time adversary $\mathcal{A}$.

In this experiment, $\mathcal{A}$ has access to the private key of the signer, $sk_{\mathcal{S}}$ and is able to create honest and corrupted users (via the oracles $\mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}$). Using the User algorithm $\mathcal{A}$ can interact with honest users. At some point[7], the adversary chooses two users and a message and, using the Choose_b oracle, receives a signature. The adversary returns a bit b.

Experiment $\mathbf{Exp}^{\text{blind}-\text{b}}_{FBSS,\mathcal{A}}(k)$

$(pk_{\mathcal{S}}, sk_{\mathcal{S}}, rpk) \leftarrow \mathcal{G}_{\mathcal{S}}(1^k)$
$(rpk, rsk) \leftarrow \mathcal{G}_{RA}(1^k)$
$b' \leftarrow \mathcal{A}(pk_{\mathcal{S}}, sk_{\mathcal{S}} : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Choose}_b, \mathsf{User}, \mathsf{TSig}, \mathsf{TSid})$
return b'

Traceability

Identity Tracing. The adversary wants to produce a new valid signature which cannot be linked to an identity. We formalize the identity tracing via an experiment involving an adversary. To fair blind signature scheme FBSS, any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we associate the experiment $\mathbf{Exp}^{\text{IdTrac}}_{FBSS,\mathcal{A}}(k)$; and the advantage is

$$\mathbf{Adv}^{\text{IdTrac}}_{FBSS,\mathcal{A}}(k) = Pr[\mathbf{Exp}^{\text{IdTrac}}_{FBSS,\mathcal{A}}(k) = 1].$$

We say that the FBSS is *identity traceable* if $\mathbf{Adv}^{\text{IdTrac}}_{FBSS,\mathcal{A}}$ is negligible for any polynomial-time adversary $\mathcal{A}$.

The adversary can create both honest and corrupted users and obtained private keys of users of his choice (via the oracles $\mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}$). He can also ask for signatures to an honest signer on messages of his choice, playing with honest or corrupted users (via the oracle Sign). At the end of the experiment he outputs a new message/signature pair and we say he wins if the identity revocation algorithm gives a non valid answer (which means that the I_{id} returned by $\mathcal{R}_{id}$ does not have the proper construction) or if the Judge is not able to find a match in the database of users identities (meaning that $\mathcal{M}_{id}$ returns 0 for the message/signature pair). We assume in this attack, that RA is not fully corrupted (it could indeed simply refuse to trace an identity), however, $\mathcal{A}$ has access to rsk, the private key of RA.

[7] The adversary is allowed to query several times the oracle Choose_b, but as explained in [7], we can restrict our experiment to adversaries that make exactly one query to this oracle.

Experiment $\mathbf{Exp}^{\text{IdTrac}}_{FBSS,\mathcal{A}}(k)$:

$(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow \mathcal{G}_{\mathcal{S}}(1^k); (rpk, rsk) \leftarrow \mathcal{G}_{RA}(1^k)$
$(m, \sigma) \leftarrow \mathcal{A}(pk_{\mathcal{S}}, rpk, rsk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Sign})$
`if` $\mathcal{V}f(pk_{\mathcal{S}}, m, \sigma) = 0$ `then` return 0
$\mathcal{R}_{id}(m, \sigma, rsk) = (I_{id}, \pi_2)$
`if` $I_{id} = \perp$ or $\mathcal{M}_{id}(I_{id}, m, \sigma, \pi_2) = 0$ `then` return 1 `else` return 0

Signature Tracing. The adversary wants to produce a valid message/signature pair such that the signer cannot give a transcript which traces this signature, or outputs two signatures linked to the same transcript. We formalize the identity tracing via an experiment involving an adversary. To fair blind signature scheme FBSS, any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we associate the experiment $\mathbf{Exp}^{\text{SigTrac}}_{FBSS,\mathcal{A}}(k)$; and the advantage is

$$\mathbf{Adv}^{\text{SigTrac}}_{FBSS,\mathcal{A}}(k) = Pr[\mathbf{Exp}^{\text{SigTrac}}_{FBSS,\mathcal{A}}(k) = 1].$$

We say that the FBSS is *signature traceable* if $\mathbf{Adv}^{\text{SigTrac}}_{FBSS,\mathcal{A}}$ is negligible for any polynomial-time adversary $\mathcal{A}$.

The adversary can create new honest users, he can also create a group of corrupted users and interacts with an honest signer. All the transcripts are put in the Trans set. At the end of the experiment, two cases might happen. Firstly, $\mathcal{A}$ outputs one message/signature pair and we say he wins if for all I_{sig} returned by RA, for all *view* registered in Trans, the Judge always answers 0. Secondly, $\mathcal{A}$ outputs two message/signature pairs and RA finds a *view* such that the Judge accepts it for both signatures. As for the identity traceability, RA is not fully corrupted.

Experiment $\mathbf{Exp}^{\text{SigTrac}}_{FBSS,\mathcal{A}}(k)$:

$(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow \mathcal{G}_{\mathcal{S}}(1^k); (rpk, rsk) \leftarrow \mathcal{G}_{RA}$
Trans $\leftarrow \emptyset$
$(m, \sigma) \leftarrow \mathcal{A}(pk_{\mathcal{S}}, rpk, rsk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Sign})$
`if` $\mathcal{V}f(pk_{\mathcal{S}}, m, \sigma) = 0$ `then` return 0
`if` $\forall view_i \in$ Trans, $\mathcal{R}_{sig}(view_i, rsk) = (I_{sig_i}, \pi_1)$ and $\mathcal{M}_{sig}(I_{sig_i}, m, \sigma, \pi_1) = 0$
`then` return 1 `else` return 0
or
$(m_1, \sigma_1), (m_2, \sigma_2) \leftarrow \mathcal{A}(pk_{\mathcal{S}}, rsk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Sign})$
`if` $\mathcal{V}f(pk_{\mathcal{S}}, m_1, \sigma_1) = 0$ or `if` $\mathcal{V}f(pk_{\mathcal{S}}, m_2, \sigma_2) = 0$
`then` return 0
`if` $\exists view_i \in$ Trans such that $\mathcal{R}_{id}(view_i, rsk) = (I_{sig}, \pi_1)$
and $\mathcal{M}_{sig}(I_{sig}, m_1, \sigma_1, \pi_1) = \mathcal{M}_{sig}(I_{sig}, m_2, \sigma_2, \pi_1) = 1$
`then` return 1 `else` return 0

Non Frameability

Non identity Frameability. In this attack, the Revocation Authority and the signer are both fully corrupted. The aim of the adversary is to provide a proof that an (honest) user obtained a valid signature whereas this very user never asked for this signature. We formalize the non identity frameability via an

experiment involving an adversary. To fair blind signature scheme FBSS, any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we associate the experiment $\mathbf{Exp}^{\text{NonIdFram}}_{FBSS,\mathcal{A}}(k)$; and the advantage is

$$\mathbf{Adv}^{\text{NonIdFram}}_{FBSS,\mathcal{A}}(k) = Pr[\mathbf{Exp}^{\text{NonIdFram}}_{FBSS,\mathcal{A}}(k) = 1].$$

We say that the FBSS is *non identity frameable* if $\mathbf{Adv}^{\text{NonIdFram}}_{FBSS,\mathcal{A}}$ is negligible for any polynomial-time adversary $\mathcal{A}$.

This time, $\mathcal{A}$ has access to both the private keys of the signer $\mathcal{S}$ and RA. He is still able to create honest and corrupted users and can interact with them playing a corrupted signer. Every time a signature is honestly issued (meaning the user is honest and accepts the protocol), the message/signature pair and the I_{id} of the user are put in Set. At the end, $\mathcal{A}$ returns a message/signature pair and he wins if this signature is valid, the R_{id} algorithm gives a valid answer, the message/signature/identity tuple is not in Set, I_{id} is the identity of an honest user and the Judge accepts the proof of RA. In this experiment, $\mathcal{A}$ is more powerful than for the traceability: he is able to fully corrupt both the signer and RA.

Experiment $\mathbf{Exp}^{\text{NonIdFram}}_{FBSS,\mathcal{A}}(k)$:

$(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow \mathcal{G}_{\mathcal{S}}(1^k) \qquad (rpk, rsk) \leftarrow \mathcal{G}_{RA}(1^k)$
Set $\leftarrow \emptyset$; HU $\leftarrow \emptyset$; CU $\leftarrow \emptyset$
$(m, \sigma) \leftarrow \mathcal{A}(pk_{\mathcal{S}}, sk_{\mathcal{S}}, rsk, rpk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{User})$
`if` $\mathcal{V}f(pk_{\mathcal{S}}, m, \sigma) = 0$ `then` return 0
$\mathcal{R}_{id}(m, \sigma, rsk) = (I_{id}, \pi_2)$
`if` $(m, \sigma, I_{id}) \notin$ Set, $I_{id} \in$ HU and $\mathcal{M}_{id}(I_{id}, m, \sigma, \pi_2) = 1$
`then` return 1 `else` return 0

Non Signature Frameability. In this attack, the Revocation Authority and the signer are both fully corrupted. The aim of the adversary is to produce a signature which is linked to an honest transcript, :*i.e.*, the adversary provides a false signature attributed to a transcript. We formalize the non signature frameability via an experiment involving an adversary. To fair blind signature scheme FBSS, any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$, we associate the experiment $\mathbf{Exp}^{\text{NonSigFram}}_{FBSS,\mathcal{A}}(k)$; and the advantage is

$$\mathbf{Adv}^{\text{NonSigFram}}_{FBSS,\mathcal{A}}(k) = Pr[\mathbf{Exp}^{\text{NonSigFram}}_{FBSS,\mathcal{A}}(k) = 1].$$

We say that the FBSS is *non signature frameable* if $\mathbf{Adv}^{\text{NonSigFram}}_{FBSS,\mathcal{A}}$ is negligible for any polynomial-time adversary $\mathcal{A}$.

$\mathcal{A}$ has access to the AddU, CrptU, USK oracles to create users, and to the User oracle to interact with honest users. Any time $\mathcal{A}$ calls the User oracle, the transcript is put in Trans. In order to win, the adversary has to produce a non listed signature (meaning his output is not in Set) that RA and the Judge can link to an existing (honest) transcript.

Experiment $\mathbf{Exp}^{\text{NonSigFram}}_{FBSS,\mathcal{A}}(k)$:

$(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow \mathcal{G}_{\mathcal{S}}(1^k)$ $\quad$ $(rpk, rsk) \leftarrow \mathcal{G}_{RA}(1^k)$
Trans $\leftarrow \emptyset$; Set $\leftarrow \emptyset$
$(m, \sigma) \leftarrow \mathcal{A}(pk_{\mathcal{S}}, sk_{\mathcal{S}}, rpk, rsk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{User})$
`if` $\mathcal{V}f(pk_{\mathcal{S}}, m, \sigma) = 0$ `then` return 0
`if` $\mathcal{R}_{id}(m, \sigma) = Id$ and $(m, \sigma, Id) \in$ Set `then` return 0
`if` $\exists view \in$ Trans such that $\mathcal{R}_{sig}(view, rsk) = (I_{sig}, \pi_1)$
and $\mathcal{M}_{sig}(I_{sig}, m, \sigma, \pi_1) = 1$ `then` return 1 `else` return 0

One-More Unforgeability
The adversary is able to interact l times with the (honest) signer. At the end of the experiment, we say that the adversary breaks the $(l, l+1)$-*one more unforgeability* if it produces $l+1$ valid signatures. We formalize the one-more unforgeability via an experiment involving an adversary.

Experiment $\mathbf{Exp}^{\text{Unforg}}_{FBSS,\mathcal{A}}(k)$:

$(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow \mathcal{G}_{\mathcal{S}}(1^k)$ $\quad$ $(rpk, rsk) \leftarrow \mathcal{G}_{RA}(1^k)$
$((m_1, \sigma_1), \ldots (m_{k+1}, \sigma_{k+1})) \leftarrow \mathcal{A}(pk_{\mathcal{S}}, rpk, rsk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Sign})$
`if` $\forall j \in [1, k+1], \mathcal{V}f(pk, m_j, \sigma_j) = 1$
and $(m_i, \sigma_i) \neq (m_j, \sigma_j)$ for $1 \leq i < j < k+1$
and at most k interactions with Sign where started
return 1 `else` return 0

When we take a closer look at this definition, we see that this property is completely contained in the traceability and non frameability properties. This was the same for group signature schemes as described in [7].

6 A New Fair Blind Signature Scheme

In this part, we present a new FBSS based on bilinear maps. Our blind scheme allows two revocations, and, in the session revocation, RA is able to find the identity of a user. Our scheme is based on the join protocol of [8] and its signature is a non-interactive proof of knowledge. In the first step, the user $\mathcal{U}$ and the signer $\mathcal{S}$ interacts, in order for $\mathcal{U}$ to get a "pseudo-blind" signature on his message. To do so, $\mathcal{U}$ commits to his message and to his participation to a shared secret. He then proves to $\mathcal{S}$ the knowledge of these values. In the same time, using the Paillier encryption scheme, he encrypts his secrets, and these encryptions are used as extractable commitments, thus yielding concurrent zero-knowledge [15]. Then, using the technique of [8], $\mathcal{S}$ signs the commitment. He sends back this signature and $\mathcal{U}$ constructs his signature as a signature of knowledge of the signed commitment.

To allow the revocation, we add two computations to this process: the first one is jointly computed by $\mathcal{U}$ and $\mathcal{S}$ during the first step and is used to trace the signature. The second one is computed by the user and added in the signature in order to allow RA to revoke the identity. We use the double ElGamal encryption

to achieve these two revocations. This scheme is IND-CCA2 if we assume the XDH assumption and if the encryption contains a proof of equality of the two encrypted values. If one does not want to use the XDH assumption, one could use instead the double Linear encryption [8] which is IND-CCA2 as well, assuming the Decision Linear assumption. For a length reason, our scheme uses the double ElGamal encryption.

6.1 Setup and Key Generation

Common parameters:
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map,
- g_1 (*resp.* g_2) is a generator of $\mathbb{G}_1$ (*resp.* $\mathbb{G}_2$) and p is the order of $\mathbb{G}_1$ (*resp.* $\mathbb{G}_2$).
- $h_1, h_2, h_3, h_4 \in \mathbb{G}_1$,
- $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is an isomorphism,
- l_p is the length of p.

Paillier Parameters: let p and q be random primes satisfying $\mathsf{p}, \mathsf{q} > 2, \mathsf{p} \neq \mathsf{q}, |\mathsf{p}| = |\mathsf{q}|$ and $\gcd(\mathsf{pq}, (\mathsf{p}-1)(\mathsf{q}-1)) = 1$; let $\mathsf{n} = \mathsf{pq}$, and $\mathsf{g} = (1+\mathsf{n})$; the public key is $pk = (\mathsf{n}, \mathsf{g})$ and the secret key is $sk = (\mathsf{p}, \mathsf{q})$. We assume $|\mathsf{n}| \geq l_p$.

Signer Parameters: Using the algorithm $\mathcal{G}_{\mathcal{S}}$, the signer $\mathcal{S}$ obtains the two keys $sk_{\mathcal{S}} := \gamma \in \mathbb{Z}_p$ and $pk_{\mathcal{S}} = g_2^{\gamma} = \Gamma$.

Revocation Authority Parameters: using the algorithm $\mathcal{G}_{RA}$, the Revocation Authority RA obtains its parameters: $rsk = (\xi_1, \xi_2) \in \mathbb{Z}_p^*$ and $rpk = (u, v) \in \mathbb{G}_1$ such that $u = {g_1}^{\xi_1}$ and $v = {g_1}^{\xi_2}$ hold.

User parameters: the user $\mathcal{U}$ computes his identity $Id_{\mathcal{U}} = h_2^{x_u}$ where $x_u \in \mathbb{Z}_p$. His identity is his public key, and the secret associated to this identity is his secret key, $(upk = Id_{\mathcal{U}}, usk = x_u)$.

The public parameters are the elements of the set $params = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, l_p, \psi, g_1, g_2, h_1, h_2, h_3, h_4, \mathsf{g}, \mathsf{n}, pk_s, rpk\}$.

6.2 Description of the Protocol

Initialization and Registration. Both the user and the signer receive the public parameters. The signer gets his private key $sk_{\mathcal{S}}$. The user $\mathcal{U}$ generates his identity and checks the validity of the signer's public key. Then, $\mathcal{U}$ registers himself to the signer $\mathcal{S}$ by sending a Paillier encryption of his secret x_u and proving its knowledge. $\mathcal{S}$ keeps this encryption in a database of identities. $\mathcal{U}$ does not need to compute a new identity for each signature but has to identify himself at first.

First Step

- $\mathcal{U}$ chooses a random value r, his participation to a shared secret s', and computes a commitment C of his message m and those two values. He computes Paillier encryptions of m, r, s'.
- He computes a double ElGamal encryption Δ_1' of the value $h_1^{s'}$.

- $\mathcal{U}$ sends to $\mathcal{S}$ all these values and proves the knowledge of their representation in one round, using the Fiat-Shamir heuristic.
- The signer checks the proof and signs the commitment C using the technique of the join protocol of [8]. To do so, he picks his participation s'' to the shared secret and a value x. He also computes a double ElGamal encryption Δ_1 of the value $h_1^{s'+s''}$ using the encryption of $\mathcal{U}$. $\mathcal{S}$ stores this encryption in a list which will be used for revocation. $\mathcal{S}$ cannot produce a proof that the two encrypted values are equal (he does not know the value $h_1^{s'+s''}$) but this encryption and the proof given by $\mathcal{U}$ are sufficient for the Judge to check the authenticity of the encryption.
- $\mathcal{S}$ sends the signature on C, s'' and x, and $\mathcal{U}$ verifies the validity of the signature.

This first step is described in more detail in Figure 3.
In the proof pok_1, the user proves that:

- he knows r, s' and m,
- m, r, s' are the same in the Paillier encryptions and in C,
- s' encrypted in Δ_1' is the same as in C.

Second Step. After receiving all the information he needs, the user has to convince the receiver, in a non-interactive way, that he knows (A, x, s, x_u, r), where (A, x, s, x_u, r) are as described in Figure 3, and that they are coherent with the verification equation.

The user adds two values, in his signature: h_1^s for the identity tracing and Δ_2 such that

$$\Delta_2 = \mathtt{Encrypt}_{rpk}^{\mathtt{2EG}}(Id_{\mathcal{U}}) = (g_1^a, g_1^{x_u} u^a, g_1^b, g_1^{x_u} v^b)$$

for the signature tracing. He also adds in the signature, a proof that Δ_2 is an encryption of $Id_{\mathcal{U}}$.

Then a valid signature on a message m consists in:

- the value h_1^s,
- the ElGamal encryption Δ_2 of $Id_{\mathcal{U}}$,
- a signature of knowledge P which proves that $\mathcal{U}$ knows $(\mathsf{A}, \mathsf{x}, \mathsf{x_u}, \mathsf{s}, \mathsf{r})$ such that:

(P_1): $e(\mathsf{A}, g_2)^{\mathsf{x}}.e(\mathsf{A}, \Gamma).e(h_1, g_2)^{-\mathsf{s}}.e(h_2, g_2)^{-\mathsf{x_u}}.e(h_3, g_2)^{-m}.e(h_4, g_2)^{-\mathsf{r}} = e(g_1, g_2)$
(P_2): $\mathsf{x_u}$ is equal to $\log_{h_2} Id_{\mathcal{U}}$, where $Id_{\mathcal{U}}$ is encrypted in Δ_2
(P_3): s is equal to $\log_{h_1} h_1^s$

A signature is of the form (h_1^s, Δ_2, P). Because of the signature of knowledge P, we need to consider for a practical point of view, that two signatures are different if they do not have the same value h_1^s. For our proofs, we will consider that a signature differs from the other ones if one of the underlying secrets (A, x, x_u, s, r, m) is different.

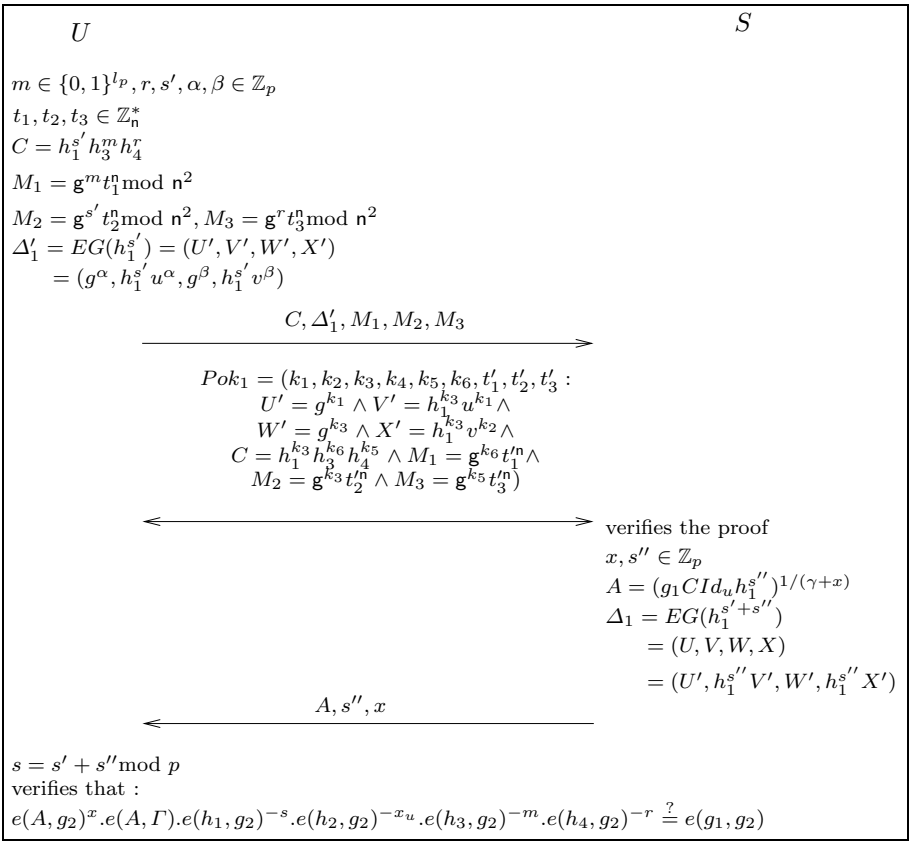


Fig. 3. Protocol

Signature Verification. The receiver is given a couple (m, σ), where σ is a signature on the message m. He verifies the signature of knowledge which proves that m was correctly signed. He also possesses Δ_2 if a revocation is needed, and knows by the signature of knowledge that an identity is correctly encrypted in it.

Identity Tracing. Given a signature, the Revocation Authority decrypts Δ_2 and finds $Id_{\mathcal{U}}$:

$$\mathsf{Decrypt}_{rsk}^{2\mathsf{EG}}(g_1^a, g_1^{x_u} u^a, g_1^b, g_1^{x_u} v^b) = g_1^{x_u} g_1^{xa} (g_1^a)^{-x} = g_1^{x_u} g_1^{xb} (g_1^b)^{-x} = Id_{\mathcal{U}}.$$

In order to prove to the Judge that it gave the real decryption, RA proves that $\log_{g_1^a}(R_1) = \log_{g_1} u$ or that $\log_{g_1^b}(R_2) = \log_{g_1} v$ where $R_1 = (g_1^{x_u} u^a)/u^a$ and $R_2 = (g_1^{x_u} v^b)/Id_{\mathcal{U}}$.

Signature Tracing. Given a transcript of a session, the Revocation Authority decrypts Δ_1 and finds h_1^s which is part of the signature:

$$h_1^s = \mathsf{Decrypt}_{rsk}^{\mathtt{2EG}}(g_1^\alpha, h_1^s u^a, g_1\beta, h_1^s v^\beta).$$

The proof that the decryption is correct is similar to the previous one.

7 Security Analysis

In this section we define the security properties our scheme provides.

Theorem 1 (Blindness). *The proposed fair blind signature scheme satisfies the blindness requirement, in the random oracle model, under the DCR assumption.*

Theorem 2 (Traceability). *The proposed fair blind signature scheme provides signature traceability and identity traceability, in the random oracle model, under the q-SDH assumption.*

Theorem 3 (Frameability). *The proposed fair blind signature scheme is non signature frameable and non identity frameable, in the random oracle model, under the DL assumption.*

The proofs are detailed in the appendix. We prove the *non identity frameability* under the one-more discrete logarithm assumption [6] to get a better reduction, but the proof can also be done under the discrete logarithm assumption.

Acknowledgments. we would like to thanks Mayasuki Abe for his helpful comments on his papers.

References

1. M. Abe. A three-move blind signature scheme for polynomially many signatures. In *Eurocrypt '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151, 2001.
2. M. Abe. Personnal communication, 2002.
3. M. Abe and M. Ohkubo. Provably secure fair blind signatures with tight revocation. In *Asiacrypt '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 583–601, 2001.
4. M. Abe and T.Okamoto. Provably secure partially blind signatures. In *CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 271–286, 2000.
5. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, 2003.
6. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *J. Cryptology*, 16(3):185–215, 2003.

7. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153, 2005.
8. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Crypto 04*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, 2004.
9. E. Brickell, P. Gemmel, and D. Kravitz. Trustee-based tracing extension to anonymous cash and the making of anonymous change. In *6th ACM-SIAM*, pages 457–466. ACM Press, 1995.
10. J. Camenisch, M. Koprowski, and B. Warinschi. Efficient blind signatures without random oracles. In *SCN*, pages 134–148, 2004.
11. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Crypto 04*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, 2004.
12. J. Camenisch, U. M. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In *ESORICS*, volume 1146, pages 33–43, 1996.
13. S. Canard, M. Gaud, and J. Traoré. Defeating malicious servers in a blind signatures based voting system. In *Financial Cryptography 2006*, 2006.
14. D. Chaum. Blind signatures for untraceable payments. In *Crypto '83*, Lecture Notes in Computer Science, page 153, 1984.
15. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, 2000.
16. A. die Solages and J. Traoré. An efficient fair off-line electronic cash system with extensions to checks and wallets with observers. In *Financial Cryptography'98, Proceedings*, volume 1465 of *Lecture Notes in Computer Science*, pages 275–295, 1998.
17. A. Fiat and A. Shamir. How to prove yourself: Pratical solutions of identifications and signature problems. In *CRYPTO '86, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, 1986.
18. M. Fischlin. Round-optimal composable blind signatures in the common reference string model. In *CRYPTO'06*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77, 2006.
19. P.-A. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368, 2001.
20. Y. Frankel, Y. Tsiounis, and M. Yung. "indirect discourse proof": Achieving efficient fair off-line e-cash. In *ASIACRYPT '96, Proceedings*, volume 1163 of *Lecture Notes in Computer Science*, pages 286–300, 1996.
21. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *ASIACRYPT '92, Proceedings*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251, 1992.
22. M. Gaud and J. Traoré. On the anonymity of fair offline e-cash systems. In *Financial Cryptography*, pages 34–50, 2003.
23. C. Hazay, C.-Y. Koo, and Y. Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In *TCC'07*, to appear, 2007.
24. A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures (extended abstract). In *CRYPTO '97,Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164, 1997.
25. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589, 2004.

26. A. Kiayias and H.-S. Zhou. Concurrent blind signatures without random oracles. In *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 49–62, 2006.
27. C. Hoon Lim and P. Joong Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *CRYPTO '97,Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–63, 1997.
28. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199, 2000.
29. T. Okamoto. Efficient blind and partially blind signatures without random oracles. In *Theory of Cryptography, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 80–99, 2006. Revisited version Cryptology ePrint Archive, Report 2006/102 http://eprint.iacr.org/.
30. T. Okamoto, A. Fujioka, and E. Fujisaki. An efficient digital signature scheme based on an elliptic curve over the ring z_n. In *CRYPTO '92, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 54–65, 1992.
31. T. Okamoto and K. Ohta. Divertible zero knowledge interactive proofs and commutative random self-reducibility. In *EUROCRYPT '89,Proceedings*, volume 434 of *Lecture Notes in Computer Science*, pages 134–148, 1989.
32. P. Paillier. Public-key cryptosystems based on composite degree residuoisity classes. In *EUROCRYPT '99, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 129–140, 1999.
33. D. Pointcheval. Strengthened security for blind signatures. In *EUROCRYPT '98, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 391–405, 1998.
34. D. Pointcheval and J. Stern. Provably secure blind signature schemes. In *ASIACRYPT '96, Proceedings*, volume 1163 of *Lecture Notes in Computer Science*, pages 252–265, 1996.
35. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
36. C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
37. M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In *EUROCRYPT '95, Proceeding*, volume 921 of *Lecture Notes in Computer Science*, pages 209–219, 1995.
38. J. Traoré. Making unfair a "fair" blind signature scheme. In *ICICS'97*, pages 386–397, 1997.

A Proofs of Security

A.1 Tools

Forking Lemma. We use the Forking Lemma [35] in our proof, to prove that an adversary $\mathcal{A}$ is not able to produce a new valid signature unless he knows all the underlying secrets (A, x, x_u, s, r, m).

Using the notation of [35], if an adversary is able to produce a signature (σ_1, h, σ_2), where $\sigma_1 = (h_1^s, \Delta_2, T, T_1, T_2, T_3, T_4)$, $h = \mathcal{H}(h_1^s, \Delta_2, T, T_1, T_2, T_3, T_4)$ and $\sigma_2 = (s_x, s_{x_u}, s_r, s_z, s_a, s_b)$, then it can produce two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1', h', \sigma_2')$ such that $h \neq h'$.

To prove the security of our scheme, we use the unforgeability property of the block messages signature based on the join protocol of [8]. We first recall the signature scheme of [8] (noted *BBS*) and turn it into a block messages signature (noted BBS_L^{ext}) as presented in [11]. We then prove the unforgeability of BBS_L^{ext} using a reduction to the *BBS* scheme.

***BBS*$_L^{ext}$ for *BBS* Signature Scheme.** In the original group signature scheme of Boneh *et al.* [8], the parameters are: g_2 a generator of $\mathbb{G}_2$, $g_1 = \psi(g_2) \in \mathbb{G}_1$, $\gamma \in \mathbb{Z}_p$,the secret key of the signer, and g_2^γ, the public key of the signer. The user's group membership is of the form (A, x) where $A = g_1^{1/(\gamma+x)}$ and x is his secret. To obtain a signature, the user proves the knowledge of his certificate. Boneh *et al.* proved under the q-SDH and the decision Linear assumption that their scheme is secure using the Bellare *et al.* security model [5].

For a block messages signature, the secret key is γ and the public key is $(g_1, g_2^\gamma, h_1, \ldots, h_L)$ with $(h_1, \ldots, h_L)$ generators of $\mathbb{G}_1$. Let $(m_1, \ldots, m_L)$ be a block of messages, then the signature is (A, x) such that $A^{\gamma+x} = g_1 h_1^{m_1} \ldots h_L^{m_L}$. Our scheme is an extension of this signature. The public key is $(g_1, g_2, h_1, h_2, h_3)$ and the so called certificate is (A, x) such that $A^{\gamma+x} = g_1 h_1^s h_2^{x_u} h_3^m h_4^r$, that is a signature of (s, x_u, m, r). We then turn it into a blind signature using a signature of knowledge which is a modification of the one used in *BBS*.

Reduction of *BBS*$_L^{ext}$ to *BBS* Signature Scheme. Let $\mathcal{A}$ be an adversary who breaks the unforgeability of BBS_L^{ext} with non negligible probability. Using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ against the unforgeability of *BBS*. $\mathcal{A}$ asks for signatures on blocks of messages $M_1 = (m_1^1, \ldots, m_L^1), M_2 = (m_1^2, \ldots, m_L^2), \ldots, M_{q_s} = (m_1^{q_s}, \ldots, m_L^{q_s})$ and receives the corresponding signatures (A_i, x_i) for $i \in \{1, \ldots, q_s\}$. Eventually, $\mathcal{A}$ outputs his forgery $(A, x, M = (m'_1, \ldots, m'_L))$. We differentiate two types of forgers:

- Type-1 Forger: a forger that outputs a forgery where $(A, x) \neq (A_i, x_i)$ for $i \in \{1, \ldots, q_s\}$.
- Type-2 Forger: a forger that outputs a forgery where $(A, x) = (A_i, x_i)$ for some $i \in \{1, \ldots, q_s\}$ and $(m'_1, \ldots, m'_L) \neq (m_1^i, \ldots, m_L^i)$.

We show that any Forger can be used to forge *BBS* signatures. However, the reduction works differently for each Forger type. Therefore, initially $\mathcal{B}$ chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will emulate.
If $c_{mode} = 1$:

- let $(g_1, g_2, \Gamma = g_2^\gamma, h)$ the public key of the *BBS* scheme sent by $\mathcal{B}$'s challenger. $\mathcal{B}$ constructs the public keys for $\mathcal{A}$ in the following way. He puts $h_1 = h$ and for $i \in [2, L], h_i = h^{r_i}$ with $r_i \in \mathbb{Z}_q^*$;
- when $\mathcal{A}$ sends a request to $\mathcal{B}$ on $(m_1, \ldots, m_L)$, $\mathcal{B}$ asks the *BBS* oracle on $M = m_1 + r_2 m_2 + \ldots + m_L r_L$ and gets the signature (A, x). It sends back (A, x) to $\mathcal{A}$ which is a valid signature for $\mathcal{A}$;
- eventually, $\mathcal{A}$ outputs his forgery (A', x') on a message $(m'_1, \ldots, m'_L)$;

- using those values, $\mathcal{B}$ directly outputs its *BBS* forgery on $M' = m'_1 + r_2 m'_2 + \ldots + r_L m'_L$s
- and so $\mathcal{B}$ breaks the unforgeability of *BBS* with the same advantage as $\mathcal{A}$.

If $c_{mode} = 2$:

- $\mathcal{B}$ chooses $\gamma \in \mathbb{Z}_p$ and fixes $\Gamma = g_2^\gamma$. It chooses $I \in \{1, L\}$ and puts $h_I = h$, and for $i \neq I, h_i = g_1^{r_i}$;
- when $\mathcal{A}$ sends a request on a message $(m_1, \ldots, m_L)$, $\mathcal{B}$ plays the signer using γ as private key;
- eventually, $\mathcal{A}$ outputs his forgery (A, x) on a message $(m'_1, \ldots, m'_L)$. By assumption, (A, x) his equal to one of the $\mathcal{A}$'s requests, let say (A_i, x_i), but it is a forgery on a new message, so $(m^i_1, \ldots, m^i_L) \neq (m'_1, \ldots, m'_L)$. There are at least two differences[8] in the two block messages.
- with probability $1/L$, $m^i_I \neq m'_I$.
- using this inequality and the verification equation, $\mathcal{B}$ is able to find the discrete logarithm of h_I, $\log_{g_1} h_I = g_1^{(\Sigma^L_{j \neq I} r_j (m'_j - m^i_j))/(m^i_I - m'_I)}$ with probability ε/L where ε is the probability of $\mathcal{A}$ to break the unforgeability of the BBS^{ext}_L. If $\mathcal{B}$ can break the discrete logarithm, then he can break the *BBS* signature scheme.

We can guess which of the two forgeries a particular Forger is with probability 1/2. So assuming the more pessimistic scenario (case 2), $\mathcal{B}$ can break the unforgeability of *BBS* with probability $\varepsilon/2L$.

A.2 Proof of Theorem 2 - Traceability

Identity Traceability. Sketch of proof
The aim of the adversary is to produce a signature such that:

- Case 1: the Revocation Authority gives a correct decryption but the Judge cannot find a match in his database. In this case, we can show that the adversary breaks the unforgeability of the BBS^{ext}_L signature. Suppose we have an adversary $\mathcal{A}$ who breaks the identity tracing of our FBSS scheme with non negligible probability. We can construct an algorithm $\mathcal{B}$ using $\mathcal{A}$ as an oracle, which breaks the unforgeability of BBS^{ext}_L. $\mathcal{B}$ receives on input from its challenger $(g_1, g_2, \Gamma = g_2^\gamma, h_1, \ldots, h_L)$ the public key of the BBS^{ext}_L scheme, and has oracle access to a BBS^{ext}_L signature oracle. It also chooses the keys for the Paillier encryption scheme. $\mathcal{B}$ sends its public key to $\mathcal{A}$ and the public key of Paillier scheme. The keys of the Revocation Authority can be chosen by $\mathcal{A}$ or $\mathcal{B}$.

  $\mathcal{B}$ simulates the oracle requests of $\mathcal{A}$. For the Sign requests, $\mathcal{B}$ plays as follows: $\mathcal{B}$ receives a Paillier encryption of x_u, m, r, s' and is able to extract them ($\mathcal{B}$ chose the private key for the Paillier encryption scheme and so it can decrypt the extractable commitments). It asks to the BBS^{ext}_L oracle a

[8] It is easy to show that it is impossible for only one block to be different.

signature on (m, r, s, x_u) where $s = s' + s''$ and s'' is chosen by $\mathcal{B}$. The oracle sends back a couple (A, x) and $\mathcal{B}$ transmits them to $\mathcal{A}$ with the value s''. So $\mathcal{B}$ is a perfect simulation of the Sign oracle for $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs a non-identity traceable signature $(\tilde{m}, h_1^{\tilde{s}}, \tilde{\Delta}_2, \tilde{P})$ with non-negligible probability ε. This signature opens on an unknown identity and so contains a value $\tilde{x}_u$ which was not used previously. Using the Forking Lemma and the soundness property of the proof, $\mathcal{B}$ is able to extract with non-negligible probability the underlying secrets $(\tilde{A}, \tilde{s}, \tilde{x}, \tilde{x_u}, \tilde{r})$ and $(\tilde{A}, \tilde{x})$ is a signature on $(\tilde{s}, \tilde{x_u}, \tilde{r}, \tilde{m})$. This is a forgery for the block messages signature scheme because the value $\tilde{x_u}$ has (by definition) never been a part of a previous block messages.

– Case 2: the Identity Revocation gives a decryption of Δ_2 (a double ElGamal ciphertext) but the two underlying plaintexts are different and so the Judge cannot conclude. The only way to do so is for the adversary to produce a false proof, in the double ElGamal encryption, that the two plaintexts are equal (although it is not the case). However he can produce such a false proof with only negligible probability assuming the soundness of the double ElGamal encryption's proof [19].

Signature Traceability. Sketch of Proof

The aim of the adversary is to produce a non traceable signature or two signatures with the same identifier. Using an adversary $\mathcal{A}$ against the signature traceability of our scheme, we can construct an adversary $\mathcal{B}$ against the BBS_L^{ext} signature scheme. $\mathcal{A}$ can output one or two signatures from his attack. $\mathcal{B}$ starts by guessing the output of $\mathcal{A}$.

– $\mathcal{B}$ guesses that $\mathcal{A}$ will output one signature and simulates all the oracle queries of $\mathcal{A}$. As for the identity traceability, $\mathcal{B}$ is able to use $\mathcal{A}$ to produce a new signature (A, x) on values (x, x_u, s, r, m). By assumption, $\mathcal{A}$ is successful in his attack and only outputs one signature. By definition, the h_1^s value of this signature is different of all the $h_1^{s_i}$ values of the signatures issued from the Sign queries. Then, the message (x, x_u, r, s, m) that $\mathcal{B}$ presents as its forgery has never been asked to the BBS_L^{ext} oracle.
– $\mathcal{B}$ guesses that $\mathcal{A}$ will output two signatures with the same value h_1^s.
  - First case: all the underlying secrets are the same, so $\mathcal{A}$ did not break the signature traceability
  - Second case: one of the secret is different. As previously, $\mathcal{B}$ uses $\mathcal{A}$ to break the unforgeability of the BBS_L^{ext} scheme. It chooses the Paillier and ElGamal keys for $\mathcal{A}$ and sends his public key to $\mathcal{A}$. When it receives the Sign queries of $\mathcal{A}$, it extracts all the values and randomly chooses a value s". It then computes the value $h_1^{s'+s''}$. If it is equal to a previously computed value it aborts the procedure. At the end of the game, $\mathcal{A}$ outputs two couples message/signature, unless $\mathcal{B}$ aborted. If $\mathcal{B}$ never aborted during the Sign queries, it knows that one of the signature was not a response to these queries. It makes a guess on this signature, and as previously, using the Forking Lemma and the soundness of the proof,

it is able to forge a new BBS_L^{ext} signature. If we let q_Σ be the number of Sign queries that $\mathcal{A}$ made, then the probability of $\mathcal{B}$ to abort is less than q_Σ^2/q which is negligible.

So, in both cases, $\mathcal{B}$ can break the unforgeability of the BBS_L^{ext} scheme.

A.3 Proof of Theorem 3 - Non Frameability

Identity Frameability. Sketch of Proof
We assume that the challenger $\mathcal{C}$ receives a random instance $(h_2^{x_1}, h_2^{x_2}, ..., h_2^{x_l})$ of the one-more DL problem. $\mathcal{C}$ will run the adversary $\mathcal{A}$ has a subroutine and act as $\mathcal{A}$'s challenger in the non-frameability attack. Because $\mathcal{C}$ is against the one-more DL assumption, he has access to a DL oracle. $\mathcal{A}$ chooses the key γ for the signer, and $\mathcal{C}$ chooses the keys p, q for the Paillier encryption and the keys (ξ_1, ξ_2) for the revocation authority. He is able to construct public parameters for $\mathcal{A}$ and he answers to the requests in the following way:

- AddU requests: $\mathcal{C}$ answers using his input of the one-more DL problem. $\mathcal{C}$ puts $Id_{U_i} = h_2^{x_i}$
- CrptU requests: $\mathcal{C}$ doest not need to do anything.
- USK requests: $\mathcal{C}$ uses is DL oracle to give the corresponding x_i to the adversary.
- User requests: in the random oracle model, $\mathcal{C}$ is able to simulate perfectly the proofs.

Now, we assume that the adversary $\mathcal{A}$ manages to produce a valid signature $(h_1^s, EG(h_2^{x'_u}), Pok(m, r, s, x, x'_u, A))$ such that he breaks the frameability of our scheme and he made k requests to the USK oracle. This means, this signature has not been obtained on a User query and the identity opening gives an $Id_{U'}$ which is in the list of honest users and so in the one-more DL problem input. We know by the definition of the experiment that no DL oracle query has been made on this identity. It follows from the Forking Lemma that if $\mathcal{A}$ is sufficiently efficient to produce such a signature, then there exists an algorithm $\mathcal{A}'$ which can produce two signatures $((h_1^s, EG(h_2^{x'_u}), Pok(m, r, s, x, x'_u, A)$ and $(h_1^s, EG(h_2^{x'_u}), \tilde{P}ok(m, r, s, x, x'_u, A)$ with non negligible probability. Using the techniques of replay and soundness, one is able to extract all the values (A', s', r', x'_u). $\mathcal{C}$ outputs the k values x_i that comes from the requests to the DL oracle plus the value x'_u and so breaks the one-more DL assumption.

Signature Frameability. Sketch of Proof
Let $\mathcal{C}$ be a challenger against the Discrete Logarithm, using our adversary $\mathcal{A}$ against the signature non-frameability. $\mathcal{C}$ is given a challenge (I, h_1) and must find $\log_{h_1} I$. $\mathcal{A}$ chooses the key γ for the signer, and $\mathcal{C}$ chooses the keys p, q for the Paillier encryption and the keys (ξ_1, ξ_2) for the Revocation Authority. He is able to construct public parameters for $\mathcal{A}$ and he answers to the requests in the following way:

- AddU requests: $\mathcal{C}$ chooses a value x_{u_i} and computes $Id_{U_i} = h_2^{x_{u_i}}$
- CrptU requests: $\mathcal{C}$ does not need to do anything.

- USK requests: $\mathcal{C}$ answers the value x_{u_i} he chose during the AddU request.
- User requests: $\mathcal{C}$ is playing the role of a user whom he knows the secret x_u. He chooses random values $\alpha, \beta, m, r \in \mathbb{Z}_p$ and computes $C = I^{\alpha} h_1^{\beta} h_3^{m} h_4^{r}$. The others values are computed following the protocol ($\mathcal{C}$ does not need to simulate them). In the random oracle model, $\mathcal{C}$ is able to perfectly simulates the proof of knowledge Pok_1 and $\mathcal{A}$ is not able to distinguish between a real proof of C and a simulated one. $\mathcal{C}$ sends these values to $\mathcal{A}$ and receives (A, s'', x). He checks the validity of the signature and puts in a I-list the values $(I^{\alpha} h_1^{s'+\beta} = I_{sig}, r, s'')$.

Finally, $\mathcal{A}$ outputs with non negligible probability his signature (m, σ) which breaks the non signature frameability of our scheme. By definition, this signature was not issued during a User request and so using the Forking Lemma, one can extract the underlying secrets (A, x, x_u, s, r, m). $\mathcal{C}$ knows that this value s matches to an $I_{sig} = h_1^s$ which is in the I-list and so $\tilde{s} = \frac{s-s'-\beta}{\alpha} = \log_{h_1} I$, unless $\alpha = 0$ which happens with probability $1/p$ which is negligible.
So $\mathcal{C}$ breaks the Discrete Logarithm assumption with non negligible probability.

A.4 Proof of Theorem 1 - Blindness

Using sequences of game, we can prove that:

$$|\Pr[S_0] - \frac{1}{2}| \leq 2^{(\varepsilon-1)(l_p+k)} + 6\mathsf{Adv}_{\mathrm{2EG}}^{\mathrm{ind-cca2}}(\mathcal{D}) + 2\mathsf{Adv}_{DCR}(\mathcal{D})$$

where S_0 is the game corresponding to the real attack and $\mathcal{D}$ is an adversary against the *IND-CCA2* property of the double ElGamal encryption scheme and against the semantic security of the Paillier encryption scheme.

Supersingular Elliptic Curves in Cryptography

Alfred Menezes

Department of Combinatorics & Optimization
University of Waterloo
ajmeneze@uwaterloo.ca

Abstract. I will survey the checkered history of supersingular elliptic curves in cryptography, from their first consideration in the seminal papers of Koblitz and Miller, to their rejection after the discovery of the Weil and Tate pairing attacks on the discrete logarithm problem for these curves, and concluding with their resurrection alongside the discovery of pairing-based cryptography.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, p. 293, 2007.

On the Minimal Embedding Field

Laura Hitt

Department of Mathematics
The University of Texas at Austin
Austin, TX 78712
lhitt@math.utexas.edu

Abstract. Let C be a curve of genus g, defined over a finite field $\mathbb{F}_q$, where $q = p^m$ for a prime p. Let N be a large integer coprime to p, dividing the order of the Jacobian variety associated to C. Pairings can transport the discrete logarithm problem (DLP) from the curve to a finite field where there are more efficient methods for solving the discrete logarithm. The embedding degree is defined to be the smallest positive integer k such that N divides $q^k - 1$. We show that the minimal embedding field is not necessarily $\mathbb{F}_{q^k}$, as is traditionally understood, but rather is $\mathbb{F}_{p^{\mathrm{ord}_N p}} = \mathbb{F}_{q^{\mathrm{ord}_N p/m}}$, which can be a field of significantly smaller size. This fact reveals that attacks on the DLP can be dramatically faster than otherwise expected, so a parameter separate from the embedding degree k needs to be used to indicate security.

Keywords: pairing-based cryptosystems, embedding degree, discrete logarithm, elliptic curve cryptography, security.

1 Introduction

The use of elliptic curves over finite fields in public-key cryptography provides greater security and more efficient performance than first generation public key techniques, such as RSA and Diffie-Hellman. Hyperelliptic curves of small genus (that is, the associated Jacobian abelian varieties with low dimension) are also believed to offer the benefits of having comparable levels of security with smaller key sizes than other finite abelian groups. Pairings on groups have been used constructively to design cryptographic protocols and to solve problems that have been open for many years, such as identity-based encryption, one-round three-party key agreement, and short signatures. On the other hand, pairings have been used destructively to attack cryptographic security. For example, the Frey-Rück attack (or MOV attack) uses the Tate pairing (or Weil pairing) to map the discrete logarithm problem (DLP) on the Jacobian of a curve to the discrete logarithm in the finite field $\mathbb{F}^*_{q^k}$, where there are more efficient methods for solving the DLP. So for pairing-based cryptosystems, it is important to find curves where the embedding degree k is small enough that the pairing is efficiently computable, but large enough that the DLP in $\mathbb{F}^*_{q^k}$ is hard.

This leads to the understanding of a *pairing-friendly* curve over $\mathbb{F}_q$ as one that satisfies the following two conditions: (1) $\#J_C(\mathbb{F}_q)$ should be divisible by a

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 294–301, 2007.

sufficiently large prime N so that the DLP in the order-N subgroup of $J_C(\mathbb{F}_q)$ is resistant to Pollard's rho attack (and other known attacks), and (2) The embedding degree k should be sufficiently large so that the DLP in $\mathbb{F}_{q^k}^*$ withstands index-calculus attacks, but small enough that the arithmetic in $\mathbb{F}_{q^k}$ can be efficiently implemented. It is important to note that while k must be small enough to enable pairings in the group, if it is too small, then the embedding field $\mathbb{F}_{q^k}$ is small enough to warrant the curve insecure for DL systems.

We show that pairing can embed into the field $\mathbb{F}_{p^{\mathrm{ord}_N p}}$, not merely into $\mathbb{F}_{q^k}$, which can dramatically speed up attacks on the DLP. The possible difference in the size of the fields has the implication in theory that there could be curves used in DL systems that are presently regarded as secure against pairing-based attacks, but are in fact insecure. That is, there could be "pairing-friendly" curves that may not be as secure as previously believed.

Published literature does not properly recognize and discuss this difference between the minimal embedding field and the field indicated by the embedding degree k. In fact, the following quote in [1] is an example of the misleading portrayal that appears in the literature.

> Let q be a prime power, and let $E/\mathbb{F}_q$ be an elliptic curve with m points in $E(\mathbb{F}_q)$. Let P in $E(\mathbb{F}_q)$ be a point of prime order p where $p^2 \nmid m$. We say that the subgroup $< P >$ has a security multiplier α, for some integer $\alpha > 0$, if the order of q in $\mathbb{F}_p^*$ is α.
>
> The MOV method can, at best, reduce the discrete log problem in $< P >$ to a discrete log problem in a subgroup of $\mathbb{F}_{q^\alpha}^*$. Therefore, to ensure that discrete log is hard in $< P >$ we want curves where α is sufficiently large to make discrete log in $\mathbb{F}_{q^\alpha}^*$ intractable.

Our paper shows that one needs not a positive *integer* k with $q^k - 1$ divisible by the size of the large prime-order subgroup; rather, it suffices to have a positive *rational* number k with $q^k - 1$ divisible by the prime. In particular, if $q = p^m$, then this rational number will be $\frac{\mathrm{ord}_N p}{m}$.

Rubin and Silverberg in [8] recognize that there may be a difference between the size of the field $\mathbb{F}_{q^k}$ and the actual embedding field for supersingular abelian varieties. They show that for supersingular abelian varieties, the difference in the size of the exponent can be at most a factor of two. Our observation is not limited to the supersingular case and explains that the difference in the fields is related to the order of the characteristic modulo the prime N, not merely on the dimension of the variety. We see that for curves of any genus, the difference in the size of the exponent can be unbounded, though it only impacts non-prime fields of small characteristic.

In section 2, we give a preliminary framework and examine the bounds on k for pairing-based attacks to be sub-exponential in q. In section 3, we discuss the underlying mathematics that causes the embedding degree of a curve to not necessarily correspond to the minimal embedding field, and hence why it may fail to capture the security of a pairing-based cryptosystem. We show that for a

curve of any genus defined over $\mathbb{F}_q$, the pairing in a group of order N embeds in a field that is not necessarily an extension of $\mathbb{F}_q$, but merely of $\mathbb{F}_p$ (where $q = p^m$). In particular, the embedding field is $\mathbb{F}_{p^{\mathrm{ord}_N p}}$. We measure the difference in size of the field exponents, finding that it grows with m. We advocate the use of two separate parameters: an embedding degree to indicate the field one must work over to compute the pairing, and a security parameter, such as $k' = \frac{\mathrm{ord}_N p}{mg}$, to indicate the minimal field containing the embedding. We then examine the bounds for attacks to be sub-exponential in the group size of the curve in light of this understanding of the minimal embedding field. Finally, in section 4, we give examples of curves that demonstrate when the embedding degree k does not correspond to the minimal embedding field and hence is a poor assessment of security. Although these curves have not been chosen for practical implementation, we hope that, for mathematical completeness, subsequent literature will acknowledge the possible difference between the field suggested by embedding degree and the actual minimal embedding field.

2 Preliminaries

Let $\mathbb{F}_q$ be a finite field with $q = p^m$ for some prime p and positive integer m, and let C be a curve over $\mathbb{F}_q$. The Jacobian of the curve is an abelian variety, J_C, of dimension g defined over $\mathbb{F}_q$. Let N be an integer dividing the order of $J_C(\mathbb{F}_q)$. A subgroup of $J_C(\mathbb{F}_q)$ with order N is said to have *embedding degree* k if N divides $q^k - 1$, but does not divide $q^i - 1$ for all $0 < i < k$.

The Tate pairing is a (bilinear, non-degenerate) function

$$J_C(\mathbb{F}_{q^k})[N] \times J_C(\mathbb{F}_{q^k})/NJ_C(\mathbb{F}_{q^k}) \longrightarrow \mathbb{F}^*_{q^k}/\mathbb{F}^{*N}_{q^k}.$$

One can then map $\mathbb{F}^*_{q^k}/\mathbb{F}^{*N}_{q^k}$ isomorphically into the set of Nth roots of unity, μ_N, by raising the image to the power $\frac{q}{N}$ not $\frac{q-1}{N}$.

Pairing-based attacks can transport the discrete logarithm problem in $J_C(\mathbb{F}_q)$ to the discrete logarithm in the finite field $\mathbb{F}^*_{q^k}$, where there are sub-exponential methods for solving the DLP. So for pairing-based cryptosystems, one would like to find curves where the embedding degree k is small enough for computations to be feasible, but large enough for the DLP in the embedding field to be difficult. We know that $k \leq 6$ for supersingular elliptic curves, as first shown in [6], and [3] gives an upper bound of 12 on k for supersingular genus 2 curves. However, for most non-supersingular curves, k is enormous since $k = \mathrm{ord}_N q$.

The best generic algorithm known for solving the DLP is Pollard's rho method, which has a fully-exponential expected running time. For particular groups, pairing-based attacks with index-calculus methods yield sub-exponential-time algorithms. The latest results, in [5], give an algorithm for computing discrete logarithms in finite fields $\mathbb{F}_{q^k}$ with heuristic complexity $L_{q^k}(1/3) = \exp((1 + o(1))(\log q^k)^{1/3}(\log\log q^k)^{2/3})$. So in order for an attack to be sub-exponential

in q, one needs $k \in o((\frac{\log q}{\log\log q})^2)$. Galbraith in [3] noted that the size of the group $J_C(\mathbb{F}_q)$ is approximately q^g, so to determine the applicability of the sub-exponential algorithms for solving the DLP in finite fields, one should actually consider k/g.

3 An Examination of the Embedding Degree

Given a subgroup of order N of a curve over $\mathbb{F}_q$, the standard definition of the embedding degree k is that k is the smallest integer such that $N \mid q^k - 1$. Since the MOV attack first used pairings to transport the discrete log problem on the curve to the discrete log problem in $\mathbb{F}_{q^k}^*$, where one can perform index calculus, the security of a cryptosystem has been assumed to be related to the size of this parameter k.

However, we point out that to properly determine the security level of a pairing-based cryptosystem, it is important to know the minimal field containing the Nth roots of unity and to incorporate this exponent as a security parameter. If $q = p^m$, then the pairings embed into μ_N which lies in $\mathbb{F}_{p^{\mathrm{ord}_N p}}^*$, not merely in $\mathbb{F}_{q^k}^*$. That is, the embedding is into the multiplicative group of an extension of $\mathbb{F}_p$, which is not necessarily an extension of $\mathbb{F}_q$. This difference in the size of the groups can be quite large, by as much as a factor of m.

Let us examine the present definition of embedding degree with respect to a general prime N over $\mathbb{F}_q$. We let $\mathrm{ord}_N p$ be the smallest positive integer x such that $p^x \equiv 1 \bmod N$.

Lemma 1. *Let $q = p^m$ for some prime p and positive integer m, N be a prime not equal to p, and k be the smallest integer such that $q^k \equiv 1 \bmod N$. Then*

$$k = \frac{\mathrm{ord}_N p}{\gcd(\mathrm{ord}_N p, m)}.$$

Proof. Clearly $k \mid \frac{\mathrm{ord}_N p}{\gcd(\mathrm{ord}_N p, m)}$, since

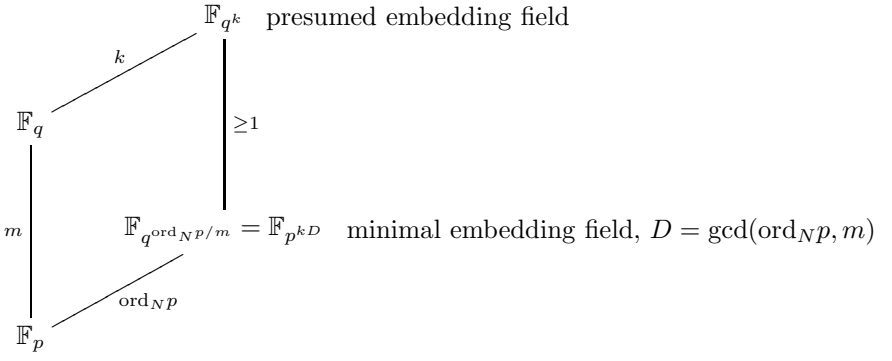
$$1 \equiv p^{\mathrm{ord}_N p} \equiv (p^{\mathrm{ord}_N p})^{m/\gcd(\mathrm{ord}_N p, m)} \equiv (p^m)^{\mathrm{ord}_N p/\gcd(\mathrm{ord}_N p, m)} \bmod N.$$

Now let $D = \gcd(\mathrm{ord}_N p, m)$. So we have $k \mid \frac{\mathrm{ord}_N p}{D}$.

We also know that $\mathrm{ord}_N p \mid mk$, and this implies $\frac{\mathrm{ord}_N p}{D} \mid \frac{m}{D}k$. But $\gcd(\frac{\mathrm{ord}_N p}{D}, \frac{m}{D}) = 1$, therefore it must be that $\frac{\mathrm{ord}_N p}{D} \mid k$. Thus we have $k = \frac{\mathrm{ord}_N p}{D}$ and the proof is complete. □

Since μ_N lies in $\mathbb{F}_{p^{\mathrm{ord}_N p}}^*$, it is apparent that the embedding field is not $\mathbb{F}_{q^k} = \mathbb{F}_{p^{km}}$, but $\mathbb{F}_{p^{\mathrm{ord}_N p}} = \mathbb{F}_{p^{kD}}$, where $D = \gcd(\mathrm{ord}_N p, m)$. So it is conceivable for the size of the actual and presumed embedding fields to differ by a factor of m.

The following field diagram helps to illustrate the difference in these fields of discussion.



We note that it is possible for this gap to be as large as one dictates, simply by increasing the exponent m prime to $\mathrm{ord}_N p$.

We see that the term "embedding degree" is a bit of a misnomer, as the minimal embedding field is not necessarily the one indicated by the embedding degree. We suggest a separate parameter be used to indicate security against solving the DLP in the finite field. In security analysis, one wants to compare the difficulty of solving the DLP in the minimal embedding field with solving the DLP in the curve group, as both should be computationally intractable. The size of the Jacobian of a genus g curve over $\mathbb{F}_q$ is approximately q^g, so [3] suggests k/g should be considered for security, as this represents the logarithmic ratio between the size of the finite field $\mathbb{F}_{q^k}$ and the size of the curve group. In light of our observation, we now suggest $k' = \frac{\mathrm{ord}_N p}{mg}$ is a more accurate indicator, as it takes into account the minimal embedding field. Whenever q is prime, then there is no difference between presumed and actual minimal embedding field sizes, so in that case we have $k = mgk'$.

Roughly speaking, for cryptographic security one needs the size of the minimal embedding field to be of approximately 1024 bits to avoid index calculus computations, and the prime N should be approximately 160 bits so that the DLP in the curve group is suitably hard. Pairing-based systems become much slower as q^k increases, so one normally chooses q^k just barely large enough to have hard discrete logarithms in the field $\mathbb{F}_{q^k}$. Suppose an elliptic curve is defined over $q = p^2$, such that q^k is larger than 2^{1024}, as is the conventional custom. If $q^{5/2} - 1$ is divisible by N, then the Tate pairing in $\mathbb{F}_{q^5}$ actually produces a result in $\mathbb{F}_{q^{5/2}}$. Computing the DLP in this field is dramatically faster than in $\mathbb{F}_{q^5}$. It is likely that the original curve was not chosen to have $q^{5/2}$ larger than 2^{1024}, so we now realize the curve was insecure. Thus the standard practice of checking traditional embedding degrees needs to be modified in light of this observation of the minimal embedding field.

To examine the potential difference between the size of the minimal field that contains the embedding and the one under the conventional notion of embedding degree, let $q = p^m$ with $m \neq 1$, and let us consider $[\mathbb{F}_{q^k} : \mathbb{F}_{p^{\mathrm{ord}_N p}}]$. That is, set $\Delta = \frac{m}{\gcd(\mathrm{ord}_N p, m)}$, so the size of Δ reveals the relative change in field size. We see that $\Delta = 1$ if and only if $\gcd(\mathrm{ord}_N p, m) = m$, which corresponds to k being

an accurate indicator of the minimal embedding field. However, it is not unusual to have $\gcd(\mathrm{ord}_N p, m) = 1$, hence $\Delta = m$, showing k to be the least accurate indicator of the minimal embedding field.

Since the minimal embedding field is $\mathbb{F}_{p^{\mathrm{ord}_N p}} = \mathbb{F}_{p^{kD}}$, where $D = \gcd(\mathrm{ord}_N p, m)$, we see that an attack will now be sub-exponential in q if $k \in o(\frac{m(\log q)^2}{D(\log\log p^{\mathrm{ord}_N p})^2})$; that is, if $k \in o(\Delta\frac{(\log q)^2}{(\log\log p^{\mathrm{ord}_N p})^2})$. So clearly more curves will be susceptible to pairing attacks than previously anticipated.

4 Examples

Let us look at some examples of genus 1 and genus 2 curves that clearly emphasize this difference between the size of the minimal embedding field and the field suggested by the conventional notion of embedding degree. Since cryptographic applications usually focus on prime fields and binary fields, and this difference in the minimal embedding field is only visible in the extension field case, we will give examples in characteristic 2. Although these curves may not be chosen for practical implementation, we hope subsequent literature will acknowledge the possibility of having a smaller embedding field in certain situations.

Example 1. Let $N = 2^p - 1$, and $q = 2^{p+s}$, for $1 \leq s \leq p+1$, $s \neq p$. We know from [9] that for each s, there exists at least one non-supersingular elliptic curve over $\mathbb{F}_q$ with $|E(\mathbb{F}_q)| = 2^s N$. We emphasize that this allows for the extension degree to be prime, so Weil descent attacks do not apply. These curves have embedding degree $k = p$, which suggests that the embedding field is $\mathbb{F}_{q^k} = \mathbb{F}_{2^{p(p+s)}}$. But in fact, $\gcd(\mathrm{ord}_N 2, p+s) = 1$ not $\gcd(\mathrm{ord}_N 2, p+1) = 1$, so the embedding field is $\mathbb{F}_{2^p}$, and these sizes differ by a factor of $\Delta = p + s$. We see that in this case the "presumed" embedding field grows quadratically in p, but the actual minimal embedding field grows only linearly in p.

We note that Appendix A of [7], which develops standard specifications for public-key cryptography, states a condition that one needs only to test whether the embedding degree is larger than some small integer B, and the largest B stated is 28. So the curves in Example 1 could have been considered as secure for DL systems, but in light of this paper's observations, we see that the resulting field size is smaller than q, with embedding degree 1, so the DLP is easy to break.

Example 2. We can consider the Mersenne prime $N = 2^p - 1$ for genus 2 curves as well, as in [4]. We note that these examples have an absolutely simple Jacobian, so these curves are not merely the product of an elliptic curve from Example 1 and another elliptic curve. For each $\lceil \frac{2p}{3} \rceil \leq m \leq p-1$, there exists a genus 2 curve over $\mathbb{F}_{2^m}$ with $\#J_C(\mathbb{F}_{2^m}) = 2^{2m-p}N$. Each curve is given by the Weil polynomial with coefficients $(a_1, a_2) = (-1, 2^m - 2^{2m-p})$. These curves have embedding degree $k = p$, which suggests that the embedding field is $\mathbb{F}_{q^k} = \mathbb{F}_{2^{pm}}$, but in fact the embedding field is $\mathbb{F}_{2^p}$, since $\gcd(\mathrm{ord}_N 2, m) = 1$. One might previously have considered these curves as secure for DL systems, but we now see the DLP is easy to break.

This observation of the misleading notion of embedding degree has motivated us to check the accuracy of k as a security parameter in curve examples in the published literature. The following is an actually proposed system in [2] which is insecure due to the observations we have mentioned above.

Example 3 (from published literature). The authors of [2] propose a family of genus 2 curves over $\mathbb{F}_q$ where $q(l) = l^2$ for any prime (power) l. The associated Jacobian variety has size $n(l) = l^4 \pm l^3 + l^2 \pm l + 1$. A prime N dividing $n(l)$ clearly divides $q^5 - 1$, but cannot divide $q^k - 1$ for $k < 5$ except in the absurdly small case of $N = 5$. So every curve of this form has embedding degree $k = 5$, as shown in [2]. However, if $n(l) = l^4 + l^3 + l^2 + l + 1$, then N divides $l^5 - 1 = q^{5/2} - 1$, so in fact the minimal embedding field cannot be larger than $\mathbb{F}_{q^{5/2}}$. This makes a dramatic difference in how large l has to be chosen for the curves to remain secure against pairing-based attacks. However no such security warning is present in [2].

As we have mentioned, whenever working over $\mathbb{F}_q$, for q a prime, there is no discrepancy between the mathematical and cryptographic notions of embedding degree, but when q is a prime power there may be a significant difference. The techniques given in [2] are presented in general for prime powers q, although most of the curves examples they list are over a prime field, and hence escape the discrepancy. One should be cautious when using these techniques to generate curves, as certain parameters may yield a prime power q, and hence the curves could be insecure in light of our observation.

We now give two numerical examples taken from [4]. Though these curves are not used in practice, they serve to illustrate our observation.

Example 4. Consider the genus 2 curve over $\mathbb{F}_{2^{267}}$ given by the Weil polynomial with coefficients $(a_1, a_2) = (-1, 2^{267} + 2^{178})$. Then $\#J_C(\mathbb{F}_{2^{267}}) = 2^{178} \cdot 17 \cdot N$, where $N = \frac{2^{4(89)}+1}{17}$ is prime, and the embedding degree is $k = 8$. So we have a 351-bit DLP on the curve, and a 2136-bit DLP in $\mathbb{F}^*_{q^k}$, which is considered hard. However, in the minimal embedding field, we have only a 712-bit DLP, which is considered easy.

Example 5. Consider the genus 2 curve over $\mathbb{F}_{2^{136}}$ given by the Weil polynomial with coefficients $(a_1, a_2) = (-1, 2^{136} + 2^{124})$. Then $\#J_C(\mathbb{F}_{2^{136}}) = 2^{124} \cdot 17 \cdot N$, where $N = \frac{2^{4(37)}+1}{17}$ is prime, and the embedding degree is $k = 37$. So we have a 5032-bit DLP in $\mathbb{F}^*_{q^k}$, which is considered hard. However, in the minimal embedding field, we have only a 296-bit DLP, which is considered easy.

5 Conclusion

We have shown that the minimal embedding field is not $\mathbb{F}_{q^k}$, but $\mathbb{F}_{q^{\mathrm{ord}_N p/m}} = \mathbb{F}_{p^{kD}}$, where $D = \gcd(\mathrm{ord}_N p, m)$. It is of critical importance to check when working over fields of small characteristic, though the observation of this paper does not affect the case of prime fields. We advocate the use of two separate

parameters: the traditional embedding degree[1] k to indicate the field one must work over to compute the pairing, and a security parameter, $k' = \frac{\mathrm{ord}_{Np}p}{mg}$, to indicate the difficulty of solving the DLP in the minimal finite field containing the embedding.

Acknowledgments

I am grateful to Felipe Voloch for his supervision, to Tanja Lange for her valuable suggestions and encouragement in this work, and to the anonymous referees.

References

1. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of Cryptology 17(4), 297–319 (2004)
2. Galbraith, S., McKee, J., Valença, P.: Ordinary abelian varieties having small embedding degree. Cryptology ePrint Archive, Report 2004/365 (2004)
3. Galbraith, S.D.: Supersingular curves in cryptography. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 495–513. Springer, Heidelberg (2001)
4. Hitt, L.: Families of genus 2 curves with small embedding degree. Cryptology ePrint Archive, Report 2007/001 (2007)
5. Joux, A., Lercier, R., Smart, N., Vercauteren, F.: The number field sieve in the medium prime case. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 326–344. Springer, Heidelberg (2006)
6. Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions Information Theory 39(5), 1639–1646 (1993)
7. IEEE P1363. Standard Specifications for Public Key Cryptography. IEEE (2000)
8. Rubin, K., Silverberg, A.: Supersingular abelian varieties in cryptology. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 336–353. Springer, Heidelberg (2002)
9. Waterhouse, W.C., Milne, J.S.: Abelian varieties over finite fields. In: 1969 Number Theory Institute (Proc. Sympos. Pure Math., State Univ. New York, Stony Brook, N.Y., 1969) Amer. Math. Soc., Providence, R.I., vol. XX, pp. 53–64 (1971)

[1] Although this can be a misleading name, we do not attempt to change it in order to avoid causing unnecessary confusion simply for the sake of theoretical accuracy.

Remarks on Cheon's Algorithms for Pairing-Related Problems

Shunji Kozaki[1], Taketeru Kutsuma[1], and Kazuto Matsuo[1,2]

[1] Institute of Information Security,
2-14-1, Tsuruya-cho, Kanagawa-ku, Yokohama 221-0835, Japan
http://lab.iisec.ac.jp/~matsuo_lab/
[2] RDI, Chuo Univ., 1-13-27, Kasuga Bunkyo-ku, Tokyo 112-8551, Japan

Abstract. In EUROCRYPT 2006, Cheon proposed breakthrough algorithms for pairing-related problems such as the q-weak/strong Diffie-Hellman problem. Using that the exponents of an element in an abelian group G of prime order p form the ring $\mathbf{Z}/p\mathbf{Z}$ structure even if G is a generic group, Cheon's algorithms reduce their complexity by Pohlig-Hellman like method over $(\mathbf{Z}/p\mathbf{Z})^*$ or its extension. The algorithms are more efficient than solving the relative discrete logarithm problems in certain cases. This paper shows that Cheon's algorithms are faster than the result obtained by the complexity analysis in Cheon's paper, i.e. the algorithms can be done within $O(\sqrt{p/d} + \sqrt{d})$ group operations, where d is a positive divisor of $p-1$ with $d \leq q$ or a positive divisor of $p+1$ with $2d \leq q$, instead of $O(\log p(\sqrt{p/d}+\sqrt{d}))$ group operations shown by Cheon. This paper also shows an improvement of one of the algorithms for q-weak Diffie-Hellman problem. The improvement can be done within $O(\epsilon\sqrt{p/d})$ group operations, where $\epsilon = \min(2/(1-\log_p d), \log p)$. Moreover, this paper discusses how to choose the group order so that the algorithms are inefficient and also shows a condition for the group order and the probability that an order satisfies the condition.

1 Introduction

The Weil and Tate pairings have been used to solve the discrete logarithm problems on (hyper-)elliptic curves [MOV91, FR94]. However, around the end of the past century, positive usages of the Weil/Tate pairing for cryptography have been proposed by Ohgishi, Sakai, and Kasahara [OSK99] and Joux [Jou00] independently. They used the pairing to construct cryptographic protocols with nice properties. Then, Boneh and Franklin [BF01] proposed an IND-ID-CCA secure identity-based cryptography under the Weil Diffie-Hellman assumption in the random oracle model. Its provable security is fundamentally obtained from the properties of the pairing. Afterwards, many provable secure protocols have been proposed using the pairings. Moreover, many kind of protocols have been carried out by using the pairings. See [Pat05] for the literature.

In 2002, Mitsunari, Sakai, and Kasahara [MSK02] proposed a traitor tracing protocol. It is based on the q-weak Diffie-Hellman problem that takes a long

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 302–316, 2007.

series of group elements as its input. Then many protocols without random oracles have been proposed based on q-weak Diffie-Hellman-like problems, e.g. [BB04b] ,[BB04a], [BBG05], [Gen06], [Oka06]. We call such kind of problems the "pairing-related problems" in this paper. The number of the inputs for a pairing-related problem is proportionate to the parameter q.

It is known that any pairing-related problems can be reduced to a discrete logarithm problem. Moreover, no algorithm which is more efficient than solving the relative discrete logarithm problems had been known for the pairing-related problems.

In 2006, Cheon [Che06] proposed breakthrough algorithms for pairing-related problems. He used that the exponents of an element in an abelian group G of prime order p form the ring $\mathbf{Z}/p\mathbf{Z}$ structure even if G is a generic group [Sho97], and the algorithms were constructed by using a subgroup of $(\mathbf{Z}/p\mathbf{Z})^*$ or its generalization. The algorithms are more efficient than solving the relative discrete logarithm problems in certain cases.

This paper shows that Cheon's algorithms are faster than the result obtained by the complexity analysis in [Che06], i.e. those can be done within $O\left(\sqrt{p/d}+\sqrt{d}\right)$ group operations, where d is a positive divisor of $p-1$ with $d \leq q$ or a positive divisor of $p+1$ with $2d \leq q$, instead of $O\left(\log p\left(\sqrt{p/d}+\sqrt{d}\right)\right)$ group operations shown in [Che06]. This paper also shows an improvement of one of the algorithms for the q-weak Diffie-Hellman problem. The problem can be solved within $O\left(\epsilon\sqrt{p/d}\right)$ group operations, where $\epsilon = \min\left(2/(1-\log_p d), \log p\right)$, by the improvement. These results lead a simple discussion about the way to choose the group order so that the algorithms are inefficient. Accordingly, we discuss a condition for the group order and shows the probability that an order satisfies the condition.

The organization of this paper is as follows: Section 2 defines the discrete logarithm problem and the pairing-related problems dealt with in this paper, and summarizes the known facts about those problems. Section 3 briefly reviews Cheon's algorithms and their complexity shown in [Che06]. Then, Section 4 shows a better complexity analysis for the algorithms and Section 5 an improvement for the q-weak Diffie-Hellman problem. Furthermore, Section 6 discusses the group order so that the algorithms are inefficient and shows the probability that an order satisfies the condition. Finally, Section 7 concludes this paper.

In this paper, we estimate for the time complexity by group operations and for the space complexity by group elements respectively according to [Che06].

2 DLP and Pairing-Related Problems

In this section, we recall the discrete logarithm problem and the pairing-related problems which are dealt with in this paper.

Let G be an abelian group whose order is a large prime number p, $g \in G$, and $\alpha \in (\mathbf{Z}/p\mathbf{Z})^*$.

In this paper, we assume that G is a generic group [Sho97] or a (generic) bilinear group [BB04a] according to the problem.

Definition 1 (Discrete Logarithm Problem). *The discrete logarithm problem (hereafter DLP) asks α for a pair $(g, [\alpha]g)$.*

DLP can be solved within $O\left(\sqrt{p}\right)$ group operations by using Shanks's baby-step giant-step algorithm [Sha71] (hereafter BSGS) or Pollard's algorithm [Pol78]. Those algorithms are usually called "square-root algorithms." For the details of the algorithms, see [Tes01] for example.

This paper deals with the pairing-related problems shown below.

Definition 2 (q-Weak Diffie-Hellman Problem [MSK02]). *The q-weak Diffie-Hellman problem (hereafter q-WDHP) asks $[1/\alpha]g$ for a $(q+1)$-tuple*

$$(g, [\alpha]g, [\alpha^2]g, \ldots, [\alpha^q]g).$$

q-WDHP is also called the "q-Diffie-Hellman inversion problem" [BB04a].

Definition 3 (q-Strong Diffie-Hellman Problem [BB04b]). *The q-strong Diffie-Hellman problem (hereafter q-SDHP) asks $([1/(\alpha+a)]h, a)$, where a is any element in $(\mathbf{Z}/p\mathbf{Z})^*$, for a $(q+2)$-tuple*

$$(h \in G_h, g, [\alpha]g, [\alpha^2]g, \ldots, [\alpha^q]g),$$

where G_h is an abelian group of order p.

If $h = g$ (and $G_h = G$), the problem is called the "q-simplified strong Diffie-Hellman problem (hereafter q-SSDHP)" [BBG05, Appendix A.4].

The blind and partially blind signatures proposed by [Oka06] are based on a 2 variable variant of q-SDHP.

Definition 4 (q-Bilinear Diffie-Hellman Inversion Problem [BB04a]). *The q-bilinear Diffie-Hellman inversion problem (hereafter q-BDHIP) asks $e(g,g)^{1/\alpha} \in G_m$ for a $(q+1)$-tuple*

$$(g, [\alpha]g, [\alpha^2]g, \ldots, [\alpha^q]g),$$

where G_m is a (multiplicative) abelian group of order p and

$$e : G \times G \to G_m$$

a non-degenerate bilinear map.

Definition 5 ($(q+1)$-Bilinear Diffie-Hellman Exponent Problem [BBG05]). *The $(q+1)$-bilinear Diffie-Hellman exponent problem (hereafter $(q+1)$-BDHEP) asks $e(h,g)^{\alpha^{q+1}} \in G_m$ for a $(2q+3)$-tuple*

$$(h \in G_h, g, [\alpha]g, \ldots, [\alpha^q]g, [\alpha^{q+2}]g, \ldots, [\alpha^{2(q+1)}]g), \tag{1}$$

where G_h is an abelian group of order p, G_m a (multiplicative) abelian group of order p, and

$$e : G_h \times G \to G_m$$

a non-degenerate bilinear map.

The "$(q+1)$-augmented bilinear Diffie-Hellman exponent problem," which takes

$$(h \in G_h, [\alpha^{q+2}]h, g, [\alpha]g, \ldots, [\alpha^q]g, [\alpha^{q+2}]g, \ldots, [\alpha^{2(q+1)}]g)$$

as its input instead of the tuple (1), is introduced by [Gen06].

Some reductions are known between the above problems:

- Both of q-WDHP and q-SDHP can be reduced to DLP in polynomial time.
- q-SSDHP can be reduced to both of q-WDHP and q-SDHP in polynomial time.
- Each of q-BDHIP and $(q+1)$-BDHEP can be reduced to q-WDHP in polynomial time.

See [BBG05, Wei05] for the details.

Until recently, no other efficient algorithm had been known for the pairing-related problems than to solve DLP obtained by the above reductions.

3 Cheon's Algorithms

In EUROCRYPT 2006, Cheon [Che06] proposed breakthrough algorithms for the pairing-related problems. Using that the exponents of an element in an abelian group G of prime order p form the ring $\mathbf{Z}/p\mathbf{Z}$ structure even if G is a generic group, Cheon's algorithms reduce their complexity by Pohlig-Hellman [PH78] like method over $(\mathbf{Z}/p\mathbf{Z})^*$ or its extension. The algorithms are more efficient than solving the relative discrete logarithm problems in certain cases.

Cheon's basic algorithm finds the discrete logarithm α for given $(g, [\alpha]g, [\alpha^d]g)$, where d is a positive divisor of $p-1$. Therefore, if there exists $d \mid p-1$ with $d \leq q$, the algorithm can be applied to the pairing-related problems.

Algorithm 1 shows an outline of the algorithm. For the correctness of the algorithm, see [Che06].

Algorithm 1. Cheon's Algorithm

Input: G: an abelian group of prime order p, g, g_1, $g_d \in G$, and $d \in \mathbf{N}$ such that $d \mid p-1$

Output: $\alpha \in (\mathbf{Z}/p\mathbf{Z})^*$ such that $g_1 = [\alpha]g$ and $g_d = [\alpha^d]g$

1: Find a generator $\zeta_0 \in (\mathbf{Z}/p\mathbf{Z})^*$
2: $\zeta \leftarrow \zeta_0^d$
3: $d_1 \leftarrow \left\lceil \sqrt{(p-1)/d} \right\rceil$
4: Find $0 \leq u_1, v_1 < d_1$ such that $[\zeta^{-u_1}]g_d = [\zeta^{d_1 v_1}]g$ by BSGS
5: $k_0 \leftarrow d_1 v_1 + u_1$ /*Note that $\alpha^d = \zeta^{k_0}$*/
6: $d_2 \leftarrow \left\lceil \sqrt{d} \right\rceil$
7: Find $0 \leq u_2, v_2 < d_2$ such that $[\zeta_0^{-u_2(p-1)/d}]g_1 = [\zeta_0^{k_0 + d_2 v_2 (p-1)/d}]g$ by BSGS
8: **return** $\zeta_0^{k_0 + (d_2 v_2 + u_2)(p-1)/d}$

Cheon [Che06] showed Theorem 1 below for the complexity of Algorithm 1.

Theorem 1 (Cheon). *Let g be an element of prime order p in an abelian group. Suppose that d is a positive divisor of $p-1$. If g, $[\alpha]g$ and $[\alpha^d]g$ are given, α can be computed within $O\left(\log p\left(\sqrt{p/d}+\sqrt{d}\right)\right)$ group operations using space for $O\left(\max\left(\sqrt{p/d},\sqrt{d}\right)\right)$ group elements.*

Furthermore, [Che06] showed a $p+1$ variant of Algorithm 1, i.e. an algorithm using a positive divisor d of $p+1$ instead of $p-1$, by applying the similar manner in Algorithm 1 on $(\mathbf{F}_{p^2})^*/(\mathbf{Z}/p\mathbf{Z})^*$ instead of $(\mathbf{Z}/p\mathbf{Z})^*$. [Che06] showed Theorem 2 below for the $p+1$ variant.

Theorem 2 (Cheon). *Let g be an element of prime order p in an abelian group. Suppose that d is a positive divisor of $p+1$ and $[\alpha^i]g$ for $i=1,2,\ldots,2d$ are given. Then α can be computed within $O\left(\log p\left(\sqrt{p/d}+d\right)\right)$ group operations using space for $O\left(\max\left(\sqrt{p/d},\sqrt{d}\right)\right)$ group elements.*

4 Better Complexity Analysis

This section shows that the upper bound of group operations required in Algorithm 1 is lower than the bound in Theorem 1, i.e. Algorithm 1 is faster than the result shown in Theorem 1.

The complexity of Algorithm 1 is dominated by BSGS in Steps 4 and 7. In Step 4, one needs to compute

$$[\zeta^{-u_1}]g_d \text{ for } u_1 = 0,1,\ldots,d_1-1,$$

where $d_1 = \left\lceil\sqrt{(p-1)/d}\right\rceil$, from given $\zeta^{-1} \in (\mathbf{Z}/p\mathbf{Z})^*$ and g_d, and also to compute

$$[\zeta^{d_1 v_1}]g \text{ for } v_1 = 0,1,\ldots,d_1-1$$

from given $\zeta^{d_1} \in (\mathbf{Z}/p\mathbf{Z})^*$ and g. Similarly, in Step 7, one needs to compute

$$[\zeta_0^{-u_2(p-1)/d}]g_1 \text{ for } u_2 = 0,1,\ldots,d_2-1,$$

where $d_2 = \left\lceil\sqrt{d}\right\rceil$, from given $\zeta_0^{-(p-1)/d} \in (\mathbf{Z}/p\mathbf{Z})^*$ and g_1, and also to compute

$$[\zeta_0^{k_0+d_2 v_2(p-1)/d}]g \text{ for } v_2 = 0,1,\ldots,d_2-1$$

from given $\zeta_0^{d_2(p-1)/d} \in (\mathbf{Z}/p\mathbf{Z})^*$ and $[\zeta_0^{k_0}]g$.

Now, we let

$$(g_t,\beta,n) \in \{(g_d,\zeta^{-1},d_1),(g,\zeta^{d_1},d_1),(g_1,\zeta_0^{-(p-1)/d},d_2),([\zeta_0^{k_0}]g,\zeta_0^{d_2(p-1)/d},d_2)\},$$

then computing the repeated integer multiplications of an element in G appeared in Steps 4 and 7 can be generalized to a problem given as follows:

Problem 1. Find

$$(g_t, [\beta]g_t, [\beta^2]g_t, \ldots, [\beta^{n-1}]g_t)$$

for given

$$g_t \in G, \beta \in (\mathbf{Z}/p\mathbf{Z})^*, \text{and } n \in \mathbf{Z}.$$

Problem 1 corresponds to finding

$$(g_t, [2]g_t, [3]g_t, \ldots, [n-1]g_t)$$

for given $g_t \in G$ and $n \in \mathbf{Z}$ in ordinary BSGS. The problem in ordinary BSGS can be solved by using the recurrent sequence

$$[i]g_t = [i-1]g_t + g_t, i = 1, \ldots, n-1.$$

In this computation, $[i]g_t$ can be obtained by a group operation from $[i-1]g_t$. Thus, the problem can be solved within $O(n)$ group operations.

In the analysis in [Che06], using the similar manner in ordinary BSGS, one computes $[\beta^i]g_t$ from $[\beta^{i-1}]g_t$ by the recurrence

$$[\beta^i]g_t = [\beta][\beta^{i-1}]g_t.$$

It needs $O(\log p)$ group operations to compute $[\beta^i]g_t$ from $[\beta^{i-1}]g_t$ by the binary method. Therefore, one needs $O(n \log p)$ group operations to solve Problem 1 by the manner in the analysis.

In the following, we discuss to use an on-line precomputation table in the above computation. While on-line precomputation tables are usually used for practical implementation of integer multiplication, this section uses a table in order to reduce the asymptotic complexity.

A scenario of the computation with a table is to use a table spanned just $\{g_t, [2]g_t, [3]g_t, \ldots, [n-1]g_t\}$. However, a table in this scenario seems difficult to construct because the elements in $\{g_t, [2]g_t, [3]g_t, \ldots, [n-1]g_t\}$ are discretely distributed in $\mathbf{Z}/p\mathbf{Z}$ in general. Another scenario is to use a table spanned all elements in $\mathbf{Z}/p\mathbf{Z}$. However, the efficiency of this scenario is not so clear because the number of elements asked by the problem is smaller than p in general. Below shows that the latter scenario is actually efficient.

Let c be a positive integer that is independent of p and $b = \lceil p^{1/c} \rceil$. Then, any $\delta \in \mathbf{Z}/p\mathbf{Z}$ can be represented by a b-adic expansion as

$$\delta \equiv \delta_0 + \delta_1 b + \delta_2 b^2 + \cdots + \delta_{c-1} b^{c-1} \bmod p, \tag{2}$$

where $0 \leq \delta_i < b$ for $i = 0, \ldots, c-1$. We consider the computation with a precomputation table by using this representation.

One can construct the on-line precomputation table

$$T = \{T_i | i = 0, \ldots, c-1\}$$

as

$$\begin{aligned} T_i &= (T_{(i,0)}, T_{(i,1)}, \ldots, T_{(i,b-1)}) \\ &= (1, [b^i]g_t, [2b^i]g_t, \ldots, [(b-1)b^i]g_t). \end{aligned}$$

In fact, T can be obtained by the following manner.

First, one puts $T_{(0,1)} = g_t$, then computes $T_{(i,1)}$ for $i = 1, \ldots, c-1$ by the recurrence

$$T_{(i,1)} = [b]T_{(i-1,1)}.$$

Each recurrence can be computed within $O(\log p)$ group operations by the binary method. Thus, it needs $(c-1)O(\log p)$ group operations to obtain $T_{(i,1)}$ for $i = 0, \ldots, c-1$.

Second, one computes T_i for each $i = 0, \ldots, c-1$. It can be done by the recurrence

$$T_{(i,j)} = T_{(i,j-1)} + T_{(i,1)}.$$

Each recurrence can be computed a group operation, so that T_i can be obtained by $b-2$ group operations for each i.

Consequently, the table T can be obtained within

$$\begin{aligned} O\left((c-1)\log p + c(b-2)\right) &= O\left(c\left(\log p + b\right)\right) \\ &= O\left(c\left(\log p + p^{1/c}\right)\right) \\ &= O\left(cp^{1/c}\right) \end{aligned}$$

group operations. In addition, T needs the space for

$$O(cb) = O\left(cp^{1/c}\right) \tag{3}$$

group elements.

For δ given by Eq. (2), $[\delta]g_t$ can be computed as follows:

$$\begin{aligned} [\delta]g_t &= [\delta_0 + \delta_1 b + \delta_2 b^2 + \cdots + \delta_{c-1}b^{c-1}]g_t \\ &= [\delta_0]g_t + [\delta_1 b]g_t + [\delta_2 b^2]g_t + \cdots + [\delta_{c-1}b^{c-1}]g_t \\ &= T_{(0,\delta_0)} + T_{(1,\delta_1)} + T_{(2,\delta_2)} + \cdots + T_{(c-1,\delta_{c-1})}. \end{aligned}$$

Therefore, it can be obtained by $c-1$ group operations by using the table T. Computing $[\delta]g_t$ for all

$$\delta \in \{\beta^i \mid i = 0, \ldots, n-1\},$$

one can obtain

$$(g_t, [\beta]g_t, [\beta^2]g_t, \ldots, [\beta^{n-1}]g_t).$$

It can be done within $(n-1)(c-1) = O(cn)$ group operations.

Summarizing, we see that Problem 1 can be solved within $O\left(c\left(p^{1/c} + n\right)\right)$ group operations using the table with $O\left(cp^{1/c}\right)$ group elements.

Step 4 in Algorithm 1 needs to solve Problem 1 two times for $n = d_1$, so that it needs

$$O\left(c\left(p^{1/c} + d_1\right)\right) = O\left(c\left(p^{1/c} + \sqrt{p/d}\right)\right) \tag{4}$$

group operations. Similarly, Step 7 in the algorithm needs to solve the problem two times for $n = d_2$, so that it needs

$$O\left(c\left(p^{1/c} + d_2\right)\right) = O\left(c\left(p^{1/c} + \sqrt{d}\right)\right)$$

group operations. Therefore, Algorithm 1 needs

$$O\left(c\left(p^{1/c} + \sqrt{p/d} + \sqrt{d}\right)\right)$$

group operations. Now, if $c \geq 4$ then

$$p^{1/c} \leq \sqrt{p/d} + \sqrt{d}.$$

Therefore, fixing c as a constant which is greater than or equal to 4, we see that Algorithm 1 can be done within

$$O\left(\sqrt{p/d} + \sqrt{d}\right)$$

group operations. Similarly, Algorithm 1 needs the space for

$$O\left(\max\left(p^{1/c} + \sqrt{p/d}, p^{1/c} + \sqrt{d}\right)\right) = O\left(\max\left(\sqrt{p/d}, \sqrt{d}\right)\right)$$

group elements if c is fixed as a constant which is greater than or equal to 4.

From Theorem 1 and the above discussion, we have Theorem 1' below.

Theorem 1'. *Let g be an element of prime order p in an abelian group. Suppose that d is a positive divisor of $p-1$. If g, $[\alpha]g$ and $[\alpha^d]g$ are given, α can be computed within $O\left(\sqrt{p/d} + \sqrt{d}\right)$ group operations using space for $O\left(\max\left(\sqrt{p/d}, \sqrt{d}\right)\right)$ group elements.*

Theorem 1' shows that Algorithm 1 is $O(\log p)$ times faster than the result shown by [Che06].

By the similar discussion for the $p+1$ variant, Theorem 2' shown below can be obtained from Theorem 2.

Theorem 2'. *Let g be an element of prime order p in an abelian group. Suppose that d is a positive divisor of $p+1$ and $[\alpha^i]g$ for $i = 1, 2, \ldots, 2d$ are given. Then α can be computed within $O\left(\sqrt{p/d} + d\right)$ group operations using space for $O\left(\max\left(\sqrt{p/d}, \sqrt{d}\right)\right)$ group elements.*

Note that the number of group operations shown in Theorem 2' can be obtained if $c \geq 3$. However, the space needs for $O\left(\max\left(\sqrt{p/d}, d\right)\right)$ group elements in the case of $c = 3$. Therefore, $c \geq 4$ is also necessary for the $p+1$ variant as well as Algorithm 1.

Remark 1. Cheon [Che06] pointed out that a reduced memory variant of Algorithm 1 can be obtained by the similar manner for Pollard's lambda algorithm with distinguished points [Pol78, QD90, Tes98]. The on-line precomputation table introduced in this section can be also applied to this variant. However, $O\left(cp^{1/c}\right)$ amount of memory is needed for the table from Eq. (3). Therefore, careful discussion on setting of a small positive integer c should be needed according to the situation when one applies the table to the variant.

5 Improvement for q-WDHP

This section proposes an improvement of Algorithm 1 for q-WDHP. It is more efficient than Algorithm 1 if $O(d) > O\left(\sqrt{p}\right)$. The complexity of Algorithm 1 achieves its lowest as $O\left(p^{1/4}\right)$ group operations when $d = O\left(\sqrt{p}\right)$. Besides, it increases monotonically as d increases for $O(d) > O\left(\sqrt{p}\right)$. Unlike Algorithm 1, the complexity of the improvement decreases monotonically as d increases. The improvement can be obviously applied to the problems which is reducible to q-WDHP.

This section deals with Problem 2 shown below.

Problem 2. Find $[\alpha^k]g$ for given: $k \in \mathbf{Z}$, a positive divisor d of $p-1$, $g \in G$, $[\alpha^d]g$, and $[\alpha^r]g$ with $r \in \mathbf{Z}$ such that $0 \leq r < d$ and

$$k \equiv r \bmod d, \tag{5}$$

where $\alpha \in (\mathbf{Z}/p\mathbf{Z})^*$ is not given.

Setting $k = p-2$, we see that q-WDHP can be reduced to Problem 2, and moreover, q-SSDHP can be directly reduced to the problem by setting $k = q+1$.

On the complexity to solve Problem 2, we have:

Theorem 3. *Let g be an element of prime order p in an abelian group and k a positive integer. Suppose that g, $[\alpha^d]g$, and $[\alpha^r]g$ are given, where d is a positive divisor of $p-1$, r is a integer such that $r \equiv k \bmod d$ and $0 \leq r < d$. Then $[\alpha^k]g$ can be computed within $O\left(\epsilon\sqrt{p/d}\right)$ group operations, where $\epsilon = \min\left(2/(1-\log_p d), \log p\right)$, using space for $O\left(\epsilon\sqrt{p/d}\right)$ group elements in the case of $\epsilon = 2/(1-\log_p d)$ or $O\left(\sqrt{p/d}\right)$ group elements in the case of $\epsilon = \log p$.*

Proof. First, ζ and k_0 in Algorithm 1 can be obtained by executing Steps 1-5 in the algorithm. They need $O\left(\log p\sqrt{p/d}\right)$ group operations and space for $O\left(\sqrt{p/d}\right)$ group elements due to the proof of Theorem 1 in [Che06] if the on-line precomputation table introduced in Section 4 is not used. On the other hand, from Eqs. (3), (4) and that

$$p^{1/c} = \sqrt{p/d} \quad \text{if } c = \frac{2}{1-\log_p d},$$

we see that they can be done within $O\left(\frac{2}{1-\log_p d}\sqrt{p/d}\right)$ group operations with space for $O\left(\frac{2}{1-\log_p d}\sqrt{p/d}\right)$ group elements by using the table. From these ζ and k_0, α^d can be obtained as

$$\alpha^d = \zeta^{k_0}.$$

Next,

$$s = \frac{k-r}{d}$$

can be obtained as an integer since Eq. (5). Then $[\alpha^k]g$ can be computed as follows:

$$[\alpha^k]g = [\alpha^{sd+r}]g = [(\alpha^d)^s\alpha^r]g = [(\alpha^d)^s][\alpha^r]g$$

from α^d, s, and given $[\alpha^r]g$. In fact, one can compute an integer $\tau \in [0, p-1]$ such that $\tau \equiv (\alpha^d)^s \bmod p$. Therefore, $[\alpha^k]g$ can be computed by using τ as follows:

$$[\alpha^k]g = [\tau][\alpha^r]g.$$

It can be done within $O(\log p)$ group operations by the binary method, so that its cost is negligible. □

Theorem 3 implies Corollary 1 below immediately.

Corollary 1. *If there exists a positive divisor $d \le q$ of $p-1$, q-WDHP can be solved within $O\left(\epsilon\sqrt{p/d}\right)$ group operations, where $\epsilon = \min\left(2/(1-\log_p d), \log p\right)$, using space for $O\left(\epsilon\sqrt{p/d}\right)$ group elements in the case of $\epsilon = 2/(1-\log_p d)$ or $O\left(\sqrt{p/d}\right)$ group elements in the case of $\epsilon = \log p$.*

Algorithm 2 shows the improvement given by Corollary 1.

Algorithm 2. Improved Algorithm for q-WDHP

Input: G: an abelian group of prime order p, g, g_r, $g_d \in G$, $d \in \mathbf{N}$ such that $d \mid p-1$
Output: $[1/\alpha]g$ for (unknown) $\alpha \in (\mathbf{Z}/p\mathbf{Z})^*$ that satisfies $g_d = [\alpha^d]g$ and $g_r = [\alpha^r]g$, where $r \in \mathbf{Z}$ such that $0 \le r < d$ and $r \equiv p-2 \bmod d$

1: **if** $2/(1-\log_p d) < \log p$ **then**
2: Find α^d from g, g_d, and d by executing Steps 1-5 in Algorithm 1 with the on-line precomputation table introduced in Section 4 at $c = 2/(1-\log_p d)$
3: **else**
4: Find α^d from g, g_d, and d by executing Steps 1-5 in Algorithm 1 without the table
5: **end if**
6: Find s, $r \in \mathbf{Z}$ such that $p-2 = sd+r$, $0 \le r < d$
7: Find $\tau \in \mathbf{Z}$ such that $0 \le \tau < p$, $\tau \equiv (\alpha^d)^s \bmod p$
8: **return** $[\tau]g_r$

Unlike Algorithm 1, the complexity of the improvement decreases monotonically as d increases. Therefore, the improvement is asymptotically faster than Algorithm 1 if $d > \sqrt{p}$. Moreover, even if the upper bound of d is fixed for moderately large value, ϵ in Theorem 3 can be regarded as a constant. For example, if the upper bound of d is fixed as $p^{7/8}$, the value $2/(1 - \log_p d)$ is bounded by 16. Therefore, in such case, ϵ can be regarded as a constant for cryptographically interesting sized p, i.e. the complexity of the algorithm for both time and space shown in Theorem 3 can be regarded as $O\left(\sqrt{p/d}\right)$ for many cases in our interesting situations.

Remark 2. In present protocols based on the pairing-related problem, the problem size is $O(q \log p)$. Therefore, q should be a polynomial of $\log p$. Then, d should also be a polynomial of $\log p$ because $d \leq q$. Consequently, the effect of the improvement shown in this section is not so large on the protocols, whereas the improvement is slightly faster than Algorithm 1 even for such small d. However, we need to pay attention to the improvement in order to make a new problem for coming protocols. Because, as well as Algorithm 1, the improvement needs only constant numbers of group elements as its inputs.

6 Discussion

This section discusses how one chooses the group order so that both of Algorithm 1 and the $p+1$ variant are inefficient for the pairing-related problems. Moreover, this section shows a condition for the group order and the probability to satisfy the condition.

In this section, we assume that d is small enough, i.e. a polynomial of $\log p$.

From Theorem 1 (Theorem 2), it seems that Algorithm 1 (resp. the $p+1$ variant) is more efficient than the ordinary square-root algorithms if

$$d = \Omega\left(\log^2 p\right), \tag{6}$$

which can be obtained from

$$O\left(\log p \sqrt{p/d}\right) \leq O\left(\sqrt{p}\right).$$

In order to avoid the ability of Algorithm 1 and the $p+1$ variant, [Che06] recommended to increase the key size or to choose p so that both of $p+1$ and $p-1$ have no small divisor greater than $\log^2 p$ according to (6).

However, from Theorem 1' (Theorem 2') newly obtained in Section 4, we see that Algorithm 1 (resp. the $p+1$ variant) is more efficient than the ordinary square-root algorithms if

$$d = \Omega(1),$$

i.e. there is asymptotically no lower bound of d for the efficiency of the algorithms.

On the other hand, Algorithm 1 needs $[\alpha^d]g$ as its input, and the $p+1$ variant needs $[\alpha^i]g$ for $i = 1, \ldots, 2d$. Therefore, Algorithm 1 (the $p+1$ variant) has the

same complexity as ordinary square-root algorithms if $p-1$ (resp. $p+1$) has no divisor $1 < d \leq q$ (resp. $1 < d \leq q/2$).

Unfortunately, $p \pm 1$ always has small divisors if p is a cryptographically interesting size, i.e. $2 \mid p \pm 1$,

$$4 \mid p-1 \text{ if } p \equiv 1 \bmod 4,$$
$$4 \mid p+1 \text{ if } p \equiv 3 \bmod 4,$$

and moreover,

$$3 \mid p-1 \text{ if } p \equiv 1 \bmod 3,$$
$$3 \mid p+1 \text{ if } p \equiv 2 \bmod 3.$$

However, the effect of these divisors on the algorithms seems to be small. Accordingly, the simplest strategy to avoid the ability of the algorithms is to choose p according to q so that $d > q$ for any prime $d \mid p_-$ and $d > q/2$ for any prime $d \mid p_+$, where $p_\pm$ is given by the following table.

$p \bmod 12$	1	5	7	11
p_-	$\frac{p-1}{12}$	$\frac{p-1}{4}$	$\frac{p-1}{6}$	$\frac{p-1}{2}$
p_+	$\frac{p+1}{2}$	$\frac{p+1}{6}$	$\frac{p+1}{4}$	$\frac{p+1}{12}$

For example, if $p \equiv 1 \bmod 12$ then $p_- = (p-1)/12$.

Now, suppose that p_- is a random positive integer. Then the probability P_- that p_- has no prime divisor less than or equal to q is given as follows:

$$P_- = \prod_{l \leq q} \left(1 - \frac{1}{l}\right),$$

where l is in prime numbers. Therefore, we see directly from Mertens's theorem [HW79, Theorem 429] that

$$P_- \approx \frac{e^{-\gamma}}{\log_e q} = \Theta\left(\frac{1}{\log q}\right),$$

where γ denotes Euler's constant.

By the similar discussion, the probability P_+ that p_+ has no prime divisor less than or equal to $q/2$ can be obtained as

$$P_+ = \Theta\left(\frac{1}{\log q}\right).$$

Assuming p_- and p_+ are independent of each other, we see that there exists the suitable p, i.e. a prime number p such that $d > q$ for any prime $d \mid p_-$ and $d > q/2$ for any prime $d \mid p_+$, with the probability $\Theta\left(1/\log^2 q\right)$. Consequently,

a randomly chosen prime order p generally does not satisfy the condition in cryptographically settings, even though the order can be obtained by cut-and-try in many time.

On the other hand, the studies on constructing a "pairing-friendly curve" are still ongoing, e.g. [MNT01], [BLS04b], [BLS04a], [GMV07], [BW05], [BN06], [SB06], [Fre06], [CKT06], [FST06]. See also [Gal05] and [DL05]. Therefore, construction of a pairing-friendly curve whose order satisfies the condition given in this section is an interesting further research subject.

Remark 3. The condition shown in this section is not sufficient in certain cases, i.e. there are the cases that Algorithm 1 (the $p+1$ variant) efficiently works even if any prime divisor of p_- (resp. p_+) is greater than q (resp. $q/2$). Because the efficiency of the algorithms is higher in the cases shown below, we should take care of the cases.

$(q+1)$-BDHEP takes $[\alpha^i]g$ for $i = 0, \ldots, q, q+2, 2q+2$ as its input. Therefore, in order to avoid the ability of Algorithm 1 on $(q+1)$-BDHEP, we should be choose p so that p_- has no prime divisor $d \leq 2q+2$ except $d = q+1$.

q-BDHIP takes $[\alpha^i]g$ for $i = 0, \ldots, q$ as its input, and moreover, it takes

$$e : G \times G \to G_m,$$

so that, for $k = 0, \ldots, 2q$, $e(g,g)^{\alpha^k}$ can be obtained by

$$e(g,g)^{\alpha^k} = e([\alpha^i]g, [\alpha^j]g), 0 \leq i, j \leq q, k = i + j.$$

Executing the algorithms on G_m with the selected elements in $\{e(g,g)^{\alpha^k} \mid k = 0, \ldots, 2q\}$ as their input, one can obtain $e(g,g)^{1/\alpha}$. Therefore, in order to avoid the ability of Algorithm 1 (the $p+1$ variant) on q-BDHIP, we should be choose p so that p_- (resp. p_+) has no prime divisor $d \leq 2q$ (resp. $d \leq q$).

7 Conclusion

This paper showed that Cheon's algorithms [Che06] are faster than the complexity shown by the analysis in [Che06]. The paper also showed an improvement of the basic algorithm. It is faster than the Cheon's algorithms for the q-weak Diffie-Hellman problem if $p-1$ has a large positive divisor d for the group order p. Based on the above results, this paper discussed how one chooses the group order so that the algorithms are inefficient, and moreover, showed a condition for the group order and the probability of the order satisfied the condition.

Acknowledgment

We would like to thank Kazumaro Aoki, Seigo Arita, and Katsuyuki Takashima for helpful discussions and the anonymous referees for their valuable comments. This research was partially supported by "The Research on Security and Reliability in Electronic Society," Chuo University 21st Century COE Program.

References

[BB04a] Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[BB04b] Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.): EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

[BBG05] Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/015, IACR, 2005. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)

[BF01] 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)

[BLS04a] Barreto, P.S.L.M., Lynn, B., Scott, M.: Efficient implementation of pairing-based cryptosystems. J. Cryptology 17, 321–334 (2004)

[BLS04b] 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)

[BN06] 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)

[BW05] Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Design, Codes and Cryptography 37, 133–141 (2005)

[Che06] Cheon, J.H.: Security analysis of strong Diffie-Hellman problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)

[CKT06] Comuta, A., Kawazoe, M., Takahashi, T.: How to construct pairing-friendly curves for the embedding degree $k = 2n$, n is an odd prime Cryptology ePrint Archive, Report 2006/427, IACR (2006)

[DL05] Duquesne, S., Lange, T.: Pairing-based cryptography. In: Cohen, H., Frey, G., Doche, C. (eds.) Handbook of elliptic and hyperelliptic curve cryptography, pp. 573–590. Chapman & Hall/CRC, Sydney (2005)

[FR94] 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)

[Fre06] Freeman, D.: Constructing pairing-friendly elliptic curves with embedding degree 10. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. LNCS, vol. 4076, pp. 452–465. Springer, Heidelberg (2006)

[FST06] Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves, Cryptology ePrint Archive, Report 2006/372, IACR (2006)

[Gal05] Galbraith, S.D.: Pairings, Advances in Elliptic Curves Cryptography. In: Blake, I., Seroussi, G., Smart, N. (eds.) LMS 317, Cambridge U. P, pp. 183–213 (2005)

[Gen06] Gentry, C.: Practical identity-based encription without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)

[GMV07] Galbraith, S.D., McKee, J., Valença, P.: Ordinary abelian varieties having small embedding degree. In: Finite Fields and Their Applications (to appear, 2007)

[HW79] Hardy, G.H., Wright, E.M.: An introduction to the theory of numbers, 5th edn. Oxford U. P., Oxford (1979)

[Jou00] Joux, A.: One round protocol for tripartite Diffie-Hellman. In: Bosma, W. (ed.) Algorithmic Number Theory. LNCS, vol. 1838, pp. 385–393. Springer, Heidelberg (2000)

[KM07] Kutsuma, T., Matsuo, K.: Remarks on Cheon's algorithms for pairing-related problems. In: Proc. of SCIS2007, no. 4A1-2, IEICE (2007)

[MNT01] Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Trans. Fundamentals E84-A(5), 1234–1243 (2001)

[MOV91] Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite fields. In: Proc. of STOC, pp. 80–89 (1991)

[MSK02] Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Trans. Fundamentals E85-A(2), 481–484 (2002)

[Oka06] Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)

[OSK99] Ohgishi, K., Sakai, R., Kasahara, M.: Notes on ID-based key sharing systems over elliptic curve (in Japanese). Tech. Report ISEC99-57, IEICE (1999)

[Pat05] Paterson, K.G.: Cryptography from pairings. In: Blake, I., Seroussi, G., Smart, N. (eds.) Advances in Elliptic Curves Cryptography. LMS 317, pp. 215–251. Cambridge U. P., Cambridge (2005)

[PH78] Pohlig, G.C., Hellman, M.E.: An improved algorithm for comuting logarithms over GF(p) and its cryptographic significance. IEEE Trans. on Info. Theory IT- 24, 106–110 (1978)

[Pol78] Pollard, J.M.: Monte Carlo methods for index computation (mod p). Math. Comp. 32, 918–924 (1978)

[QD90] Quisquater, J.-J., Delescaille, J.-P.: How easy is collision search. New results and applications to DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 408–413. Springer, Heidelberg (1990)

[SB06] Scott, M., Barreto, P.S.L.M.: Generating more MNT elliptic curves. Designs, Codes and Cryptography 38, 209–217 (2006)

[Sha71] Shanks, D.: Class number, a theory of factrization, and genera. In: Proc. of Symp. Math. Soc., vol. 20, pp. 415–440 (1971)

[Sho97] Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)

[Tes98] Teske, E.: Speeding up pollard's rho method for computing discrete logarithms. In: Buhler, J.P. (ed.) Algorithmic Number Theory. LNCS, vol. 1423, pp. 541–553. Springer, Heidelberg (1998)

[Tes01] Square-root algorithms for the discrete logarithm problem (A survey), Public-Key Cryptography and Computational Number Theory, pp. 283–301, Walter de Gruyter, Berlin-New York (2001)

[Wei05] Wei, V.K.: Tight reductions among strong Diffie-Hellman assumptions, Cryptology ePrint Archive, Report 2005/057, IACR (2005)

On Pairing Inversion Problems

Takakazu Satoh*

Department of Mathematics,
Tokyo Institute of Technology, Tokyo, 152-8551, Japan
satopairing07@mathpc-satoh.math.titech.ac.jp

Abstract. In many aspects, cryptanalyses of pairing based cryptography consider protocol level security and take difficulties of primitives for granted. In this survey, we consider pairing inversion. At the time this manuscript was written(April 2007), to the best of the author's knowledge, there are neither known feasible algorithms for pairing inversions nor published proofs that the problem is unfeasible.

Keywords: elliptic curves, pairing based cryptography, complexity.

1 Introduction

Pairing based cryptography added one more very important "raison d'être" to elliptic curve cryptography. Before Joux's one round tripartite key sharing protocol[16], what can be done with elliptic curve cryptography can be done with cryptosystems based on the discrete logarithm problem over multiplicative groups of finite fields (with acceptable increase of computational cost). Compared to finite fields, understanding elliptic curve cryptography needs much more prerequisites. They are enough for some cryptographers to adhere to finite field based cryptosystems. However, in order to realize pairings used in pairing based cryptography, understanding elliptic curves or other algebraic geometrical objects is indispensable.[1]

In designing cryptographic protocols, pairings are usually regarded as a black box for which the bilinear Diffie-Hellman problem is difficult. However, elliptic curves used in pairing based cryptography are so called pairing friendly curves which are very rare (Balasubramanian and Koblitz[1], Luca and Shparlinski[23]). Although it is widely believed that the generic elliptic curve discrete logarithm problem(ECDLP) is unfeasible, hardness of that for pairing friendly curve is not well studied. Not only this problem, but also many other problems on pairing based cryptography still remain open (cf. Lange[20]). In this article, we focus our attention on pairing inversion problems.

The paper is organized as follows. In Section 2, we recall some basic definitions on pairings. Section 3 reviews the work of Verheul. Section 4 is a survey of

* The author would like to thank Steven Galbraith, Florian Hess and Loren Olson for discussion and/or comments.

[1] One can use the technique of Cocks[6] but this method needs so many arithmetic operations which is a problem in application.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 317–328, 2007.

attempts to pairing inversion problems. In Section 5, we compare difficulties of problems related to pairing inversions.

Notation. For the abelian group G and a non-zero integer n, we denote the subgroup of n-torsion elements of G by $G[n]$. The point at infinity of an elliptic curve given by the Weierstrass model is denoted by $\mathcal{O}$.

2 Basic Definitions

Let G_1 and G_2 be abelian groups whose group operations are written additively. Let H be the abelian groups whose group operations are written multiplicatively. A map $e : G_1 \times G_2 \to H$ is called a **bilinear map** if it satisfies the following two conditions:

(1) $e(a+b,c) = e(a,c)e(b,c)$ for all $a, b \in G_1$ and $c \in G_2$.
(2) $e(a,b+c) = e(a,b)e(a,c)$ for all $a \in G_1$ and $b, c \in G_2$.

Moreover, if the next condition holds, e is said to be **non-degenerate**.

(3) For any non-zero $a \in G_1$, there exists $b \in G_2$ satisfying $e(a,b) \neq 1$ *and* for any non-zero $b \in G_2$ there exists $a \in G_1$ satisfying $e(a,b) \neq 1$.

In case that $G_1 = G_2$ and that $e(a,b) = e(b,a)$ (resp. $e(a,b) = e(b,a)^{-1}$) for all $a, b \in G$, the pairing e is said to be **symmetric** (resp. **alternate**). One important property of an alternate pairing is that $e(a,a) = 1$ provided that order of a is odd.[2]

We formulate the following two problems:

Computational bilinear Diffie-Hellman Problem (CBDHP): Let $a \in G_1$ and $b \in G_2$ and assume $e(a,b) \neq 1$. Given ma and $na \in G_1$, compute $e(mna, b)$.

Decision bilinear Diffie-Hellman Problem (DBDHP): Let $a \in G_1$ and $b \in G_2$ and assume $e(a,b) \neq 1$. Given $ma, na \in G_1$ and $h \in H$, determine whether $e(mna, b) = h$ or not.

They are bilinear analogues of the following classical problems:

Computational Diffie-Hellman Problem (CDHP): Let $a \in G_1$ and assume $a \neq 0$. Given ma and $na \in G_1$, compute mna.

Decision Diffie-Hellman Problem (DDHP): Let $a \in G_1$ and assume $a \neq 0$. Given $ma, na, b \in G_1$, determine whether $mna = b$.

In the case that $G_1 = G_2$ and that G_1 is a cyclic group of prime order, existence of a non-degenerate pairing implies that DDHP for G_1 is easy. Note that $e(a,a)$ is a generator of H (otherwise, e is degenerate). Hence we can solve DDHP by testing $e(ma, na) = e(b, a)$.

[2] The assumption on the order cannot be dropped. Consider the bilinear pairing $e : \mathbf{Z}/2\mathbf{Z} \times \mathbf{Z}/2\mathbf{Z} \to \{\pm 1\}$ defined by $e(a,b) = 1$ for $ab = 0$ and $e(1,1) = -1$.

So far, we did not exploit the underlying structure of G_1, G_2 and H. In reality, we use (variants of) the Tate pairing or the Weil pairing on suitable (Jacobians of hyper-)elliptic curves defined over finite fields. Let p be a prime and let q be a power of p. For simplicity, we only consider elliptic curves and their points whose order is an odd prime. Let E be an elliptic curve over $\mathbf{F}_q$ and let l ($\neq p$) be an odd (and large) prime dividing $\#E(\mathbf{F}_q)$. Denote the group of the l-th roots of unity by μ_l. The **embedding degree** for l is defined to be the minimal positive integer r such that $q^r \equiv 1 (\mathrm{mod}\ l)$, or equivalently the field extension degree $[\mathbf{F}_q(\mu_l) : \mathbf{F}_q]$. A similar but definitely different notion is the **security parameter** for l which is the minimal positive integer s such that $p^s \equiv 1 (\mathrm{mod}\ l)$, or equivalently the field extension degree $[\mathbf{F}_p(\mu_l) : \mathbf{F}_p]$. In order that a pairing based protocol efficiently works, it is *necessary* that the following two conditions must simultaneously hold:

(1) The embedding degree must not be so large. Otherwise computing values of pairings is unfeasible.
(2) The security parameter must be large. Otherwise we can solve DLP on μ_l by applying index calculus algorithm to $\mathbf{F}_p(\mu_l)$.

By the minimality of s, it is obvious that $s \leq [\mathbf{F}_q : \mathbf{F}_p]r$. But r being large does not necessarily imply large s.[3] An elliptic curve satisfying these two condition is called as a **pairing friendly curve**. They are rather rare (Balasubramanian and Koblitz[1], Luca and Shparlinski[23]). The probability that a randomly generated curve being pairing friendly is negligible. In practice, there are two methods to find out pairing friendly curves.

(1) Supersingular curves. In this case, the possible r is either 1, 2, 3, 4 or 6 (Menezes, Okamoto and Vanstone[26], [27]).
(2) Generate pairing friendly curve using complex multiplication theory: This topic alone amounts to an another survey article. See e.g. Freeman, Scott and Teske[10]. Here we refer only few of known results. Miyaji, Nakabayashi and Takano[29] for $r = 3, 4, 6$. Freeman[9] for $r = 10$. Barreto and Naehrig[4] for $r = 12$. In their unpublished paper,[4] Cocks and Pinch give an algorithm for arbitrary r but the characteristics of the ground fields of the resulting curves by the Cocks and Pinch algorithm are of order r^2 rather than r. See also Brezing and Weng[5].

In many applications, G_1 and G_2 are identical cyclic subgroups of $E(\mathbf{F}_q)[l]$. However, the Weil pairing is always trivial on $G_1 \times G_1$ due to its alternating property (note that l is odd). Less obvious is that the Tate pairing is always trivial in case that $r > 1$. (See e.g. Galbraith[11, Lemma IX.13].) A **distortion map** for the group $\langle P \rangle$ where $P \in E[l]$ is an effectively computable group

[3] To the best knowledge of the author, the importance of distinguishing r and s is completely overlooked until the initial preprint version of Hitt[15] was posted to the IACR e-print server 2006/245.
[4] A description of their algorithm can be found in Galbraith[11].

homomorphism $\delta : \langle P \rangle \to E[l]$ satisfying $\delta(P) \notin \langle P \rangle$. It is used to construct a non-trivial self pairing. For any non-degenerate pairing e on $E[l]$, either $e(P, P) \neq 1$ or $e(P, \delta(P)) \neq 1$ since P and $\delta(P)$ form a base of $E[l]$.[5]

3 The Work of Verheul

In this section, assume that G_1, G_2, H are cyclic groups of an equal order of known factorization (e.g. prime order). To state Verheul's results[34], we introduce one more problem related to CDHP.

Weak Diffie-Hellman Problem (WDHP): Let $g \in H$ be a generator. Find a generator $\omega \in H$ such that ω^{mn} is feasibly computed from g^m and g^n for all g^m, $g^n \in H$.

Clearly, CDHP implies WDHP. We consider the situation where there exist efficient algorithms to compute the following three maps.

(1) $e : G_1 \times G_2 \to H$, a non-degenerate bilinear pairing
(2) $\delta : G_1 \to G_2$, an isomorphism s.t. $e(x, \delta(x)) \neq 1$.
(3) $v : H \to G_1$, an isomorphism

Verheul observed that WDHP for any generator g of H is feasible in the above setting. Indeed, put $\omega := e(v(g), \delta(v(g)))$. Then, bilinearlity of e yields $e(v(g^m), \delta(v(g^n))) = \omega^{mn}$. Moreover, he proved that WDHP for H implies CDHP for H or even stronger variants of CDHP. (His proof of this part uses WDHP as an oracle. Pairings and distortions do not appear explicitly.) He actually constructed the maps δ for the Weil pairing on certain supersingular elliptic curves. He concluded that v is unlikely to be feasibly computational for that curve because CDHP for H in his example is believed to be difficult.

Following Koblitz and Menezes[19], we call $v : H \to G_1$ the Verheul map. In general, the problem of constructing a map from H to $G_1 \times G_2$ which has "nice" properties is called the **pairing inversion problem**. Here, we introduce two more well known pairing inversion problems:

Fixed Pairing Inversion (FPI): Let $a \in G_1$ be fixed. For given $z \in H$, find $b \in G_2$ s.t. $e(a, b) = z$.
Generalized Pairing Inversion (GPI): For given $z \in H$, find $a \in G_1$ and $b \in G_2$ s.t. $e(a, b) = z$.

They definitely appear in Joux[17], but their origins are unknown.

[5] In some of the literature, a distortion map is defined as a non-$\mathbf{F}_q$ rational endomorphism. Since the endomorphism ring of an ordinary elliptic curve is commutative, a non-$\mathbf{F}_q$ rational endomorphism exists only for supersingular curves. However, one can construct a distortion map in the sense of our definition for ordinary curves and it plays the required role.

We emphasize that all cryptographic protocols based on CDHP[6] on H are broken if we can find one 5-tuple (G_1, G_2, δ, e, v) for which v, δ, and e are effectively computable. That is, the effect of pairing inversion problems is not limited to pairing based cryptography. For some values of l, Maurer and Wolf[24], [25] proved that CDHP and DLP for μ_l are equivalent in some sense. Although characterization of l for which CDHP and DLP is equivalent is yet still an open problem, for those l, pairing inversion problems affects a wider class of cryptographic systems.

4 Hardness of Pairing Inversions

Surprisingly, hardness of pairing inversion is not well studied despite of its importance. Historically, the problem is considered in conjunction with reduction of ECDLP to DLP in multiplicative groups of finite fields(FFDLP). Let $E/\mathbf{F}_q$ be an elliptic curve. Let l $(\neq p)$ be a prime dividing $\#E(\mathbf{F}_q)$ and let r be its embedding degree. Menezes, Okamoto, Vanstone[26], [27] (with the Miller algorithm[28]) and Semaev[32] independently discovered that ECDLP is reduced to FFDLP in $\mathbf{F}_{q^r}^{\times}$ by a probabilistic polynomial time algorithm. Roughly speaking, ECDLP in $E(\mathbf{F}_q)[l]$ is not more difficult than FFDLP in $\mathbf{F}_{q^r}^{\times}$. However, "generic" ECDLP is conjectured to be a more difficult problem than FFDLP. In order prove this, it suffices to construct an isomorphism, which is now called the Verheul map. Verheul's pioneering results showed that this plan probably does not work, at least in general. But this does not imply that a pairing inversion is hard for a particular elliptic curve. There might exist a family of elliptic curves for which a pairing inversion is easy.

Satoh[30] considered polynomial interpolations of the X-coordinate of the Verheul map. Assume that E is given by the affine Weierstrass model. We denote the X-coordinate function and the Y-coordinate function by X and Y, respectively. Choose (and fix) a generator $B \in E[l]$ and a generator $\zeta \in \mu_l$. There exists V_X, $V_Y \in \mathbf{F}_{q^r}[z]$ satisfying

$$\deg V_X \leq l-2, \quad V_X(\zeta^n) = X(nB) \text{ for } 0 < n < l-1,$$
$$\deg V_Y \leq l-2, \quad V_Y(\zeta^n) = Y(nB) \text{ for } 0 < n < l-1.$$

If V_X and V_Y can be evaluated quickly, we obtain one of the feasible Verheul maps. Assuming l to be odd, Satoh[30, Theorem 3] obtained

$$\deg V_X \geq \begin{cases} (l-1)/5 & (p \neq 2), \\ (l-1)/2 & (p = 2). \end{cases} \tag{4.1}$$

[6] An important example is the ElGamal cryptosystem. The fact that CDHP is enough to break the ElGamal cryptosystem was already pointed out in the original paper by ElGamal[8].

The proof uses division polynomials and technique developed by Lange and Winterhof[21], [22]. The inequality (4.1) is too weak to insist hardness of the Verheul map. A polynomial interpolation is just one of many possible methods to compute the Verheul maps. Moreover some large degree polynomials (such as monomials) can be evaluated quickly. Kiltz and Winterhof[18] obtained estimates of weights for polynomial interpolations for CDHP over multiplicative groups of finite fields. However, the method does not seem to be applicable for elliptic curves, at least in a straightforward way. In the case $q = p \geq 5$ and $\sqrt{12p} < l < 2p/\sqrt{3}$, Satoh[31] proved the following assertion: None of the coefficients of V_X vanishes (in particular $\deg V_X = l - 2$) for about 58and a cyclic subgroup of order l. The proof utilizes that over the complex number filed, the coefficients of V_X are certain modular forms of weight two, of which we can consider "reduction mod p". The theory of modular forms and modular functions gives a number of zeros of modular forms. By taking mod p, we can estimate how frequently at least one coefficient of V_X vanishes. Unfortunately, it seems very difficult to say something about a particular curve with this "moduli" approach.

Recently, Galbraith, ÓhÉigearaigh and Sheedy[13] proposed another method. The idea is that η-pairing on supersingular curves can be expressed in a simple form so that we have a non-trivial way to compute the Verheul map.

Assume that E is a supersingular elliptic curve. As before, l is an odd prime dividing $\#E(\mathbf{F}_q)$ and r is its embedding degree. For simplicity, we restrict ourselves to the case that E is defined over $\mathbf{F}_p$. Assume that r is even and that $\gcd(r, m) = 1$ where $m := [\mathbf{F}_q : \mathbf{F}_p]$. We also assume that there is a non-$\mathbf{F}_q$ rational endomorphism ψ satisfying

$$\pi^{r/2}(\psi(P)) = -\psi(P)$$

for $P \in E(\mathbf{F}_q)$ where π is the q-th power Frobenius. Put $M := q^{r/2} - 1$. For $n \in \mathbf{Z}$ and $P \in E(\mathbf{F}_q)$, there exists $f_{n,P} \in \mathbf{F}_q(E)$ such that $\mathrm{div}(f_{n,P}) = n[P] - [nP] - (n-1)[\mathcal{O}]$ and that the leading coefficient of $f_{n,P}$ with respect to a local parameter defined over $\mathbf{F}_q$ at $\mathcal{O}$ is 1. We define the η-pairing[7] by

$$\eta(P, Q) = f_{q,P}(\psi(Q)) \tag{4.2}$$

for P, $Q \in E(\mathbf{F}_q)$.[8] In general, the η pairing itself may not be a bilinear pairing. However $\eta^{Mqr/2}$ coincides with the reduced Tate pairing by Barreto, Galbraith, Ó'hÉigeartaigh and Scott[3, Theorem 1]. Usually, $\eta(P, Q)$ is evaluated by the Miller algorithm with several optimizations followed by raising to the $(Mqr/2)$-th power. A key observation in Galbraith, ÓhÉigeartaigh and Sheedy[13] is

[7] For generalities on the η-pairing, see Barreto, Galbraith, Ó'hÉigeartaigh and Scott[3].

[8] Note that $f_{n,P}$ and $\eta(P, Q)$ depend on the choice of a local parameter defined over $\mathbf{F}_q$. However, since r is even, $c^M = 1$ for all $c \in \mathbf{F}_q^\times$. Hence $\eta(P, Q)^M$ is independent of the choice of a local parameter defined over $\mathbf{F}_q$. See [2, Lemma 1 and Theorem 1]. If E is given by the affine Weierstrass model in the XY-plane, one can always use $-Y/X$ as a local parameter defined over $\mathbf{F}_q$ (or even any prime field) at $\mathcal{O}$.

that computing (4.2) without optimizations (in particular, so-called denominator elimination), the order of resulting value is divisible by $q^{r/2}+1$. Therefore, $\eta(P,Q)^M = \eta(P,Q)^{-2}$. This technique is also applicable to the Duursma-Lee[7] curves $Y^2 = X^p - X \pm 1$. For simplicity, we consider the case $p=2$ in what follows. In this case, $\eta(P,Q)$ itself is a non-degenerate bilinear pairing. It can be evaluated as

$$\eta(P,Q) = \prod_{i=0}^{m-1} \left(\frac{g_{p^i P}(\psi(Q))}{v_{p^{i+1}P}(\psi(Q))} \right)^{p^{m-1-i}}$$

where g_A and v_A are suitably normalized rational functions with $\mathrm{div}(g_A) = p[A] + [-pA] - (p+1)[\mathcal{O}]$ and $\mathrm{div}(v_A) = [A] + [-A] - 2[\mathcal{O}]$, respectively. Take (and fix) an $\mathbf{F}_2$ basis $\{\theta_0, \ldots, \theta_{m-1}\}$ of $\mathbf{F}_q$ and $P \in E(\mathbf{F}_q)[l]$. For a given $z \in \mu_l$ satisfying $z \neq 1$, we want to find a unique point $Q \in E(\mathbf{F}_q)[l]$ satisfying $\eta(P,Q) = z$. We introduce $2m$ variables $x_0, \ldots, x_{m-1}$ and $y_0, \ldots, y_{m-1}$ over $\mathbf{F}_2$ and put $Q = \left(\sum_{i=0}^{m-1} x_i \theta_i, \sum_{i=0}^{m-1} y_i \theta_i \right)$. Note that the squaring map is an $\mathbf{F}_2$-linear map. Thus, there exist constants $a_{i,j}$, $b_{i,j}$, $c_{i,j}$, $u_{i,j}$, $v_{i,j}$, $w_{i,j} \in \mathbf{F}_{q^r}$ (depending on P) such that

$$\eta(P,Q) = \prod_{j=0}^{m-1} \frac{\sum_{i=0}^{m-1} (u_{i,j}x_i + v_{i,j}y_i + w_{i,j})}{\sum_{i=0}^{m-1} (a_{i,j}x_i + b_{i,j}y_i + c_{i,j})}.$$

Therefore, in order to find Q satisfying $\eta(P,Q) = z$, it is suffice to solve a multivariate polynomial equation

$$z \prod_{j=0}^{m-1} \sum_{i=0}^{m-1} (a_{i,j}x_i + b_{i,j}y_i + c_{i,j}) - \prod_{j=0}^{m-1} \sum_{i=0}^{m-1} (u_{i,j}x_i + v_{i,j}y_i + w_{i,j}) = 0.$$

Solving such a equation is laborious and infeasible for practical values of m.[9]

Ordinary curves are another story. The following results are obtained by Galbraith, Hess and Vercauteren[12]. Let E be an ordinary elliptic curve defined over $\mathbf{F}_q$. Denote the trace of the q-th Frobenius by t. In what follows, assume $t > 1$. Recall that the elliptic ate pairing (see Hess, Smart and Vercauteren[14]) is defined as $f_{t-1,P}(Q)^d$ with some positive integer d. We decompose a pairing inversion problem into two steps:

Final Exponentiation Inversion: Finding correct d-th root of a given value.
Miller Inversion: Given $z \in \mu_l$, find Q satisfying

$$f_{t-1,P}(Q) = z. \tag{4.3}$$

[9] Nevertheless, [13] gives implementation results for small m, which is suggestive. The possible use of Duursma-Lee type higher genus curves to reduce computational complexity for solving pairing inversion is observed there too.

The smaller t is, the smaller the degree (as a rational function) of $f_{t-1,P}$ is, which means a fewer number of arithmetic operations to solve (4.3). In [12], it is shown that there exist families of pairing friendly elliptic curves with small t so that the Miller inversion is feasible. As to final exponentiation inversion, [12] proved that a random d-th root of a pairing value can be passed to the Miller inversion phase for some elliptic curves. An open problem is whether the two families of elliptic curves have non-empty intersection.

5 Relations Among Pairing Inversion Problems

It is obvious that FPI implies GPI. However, GPI is believed to be weaker than FPI. What else can we say about relations between two pairing inversion problems? In this section, we consider relations of the Weil pairing inversion on supersingular elliptic curves. Although $E[l]$ is not cyclic, FPI and GPI straightforwardly generalize to pairings on non-cyclic groups. In what follows, E denotes a supersingular elliptic curve defined over $\mathbf{F}_q$. Let l be an odd prime dividing $\#E(\mathbf{F}_q)$ with embedding degree $r > 1$. Let e be the l-th Weil pairing over $E[l] \times E[l]$. We denote the q-th power Frobenius endomorphism by π. For a field extension $\mathbf{F}_{q^m}/\mathbf{F}_q$, we define the trace map $\mathrm{Tr}_{E(\mathbf{F}_{q^m})/E(\mathbf{F}_q)}$ by

$$\mathrm{Tr}_{E(\mathbf{F}_{q^m})/E(\mathbf{F}_q)} := \sum_{i=0}^{m-1} \pi^i.$$

By definition, $\mathrm{Tr}_{E(\mathbf{F}_{q^m})/E(\mathbf{F}_q)} \in \mathrm{End}(E)$ and it sends $E(\mathbf{F}_{q^m})$ to $E(\mathbf{F}_q)$. Put $G_l := E(\mathbf{F}_q) \cap E[l]$ and $G'_l := \mathrm{Ker}\,\mathrm{Tr}_{E(\mathbf{F}_{q^r})/E(\mathbf{F}_q)} \cap E[l]$.[10] The assumption $r > 1$ implies that G_l and G'_l are isomorphic to $\mathbf{Z}/l\mathbf{Z}$. Put $T := r^{-1}\,\mathrm{Tr}_{E(\mathbf{F}_{q^r})/E(\mathbf{F}_q)}$ (r^{-1} is mod l inverse), $T' := 1 - T$. Note that T and T' are projections to G_l and G'_l, respectively. We begin with a technical lemma.

Lemma 1. *Let δ be a non-$\mathbf{F}_q$-rational map and define $\tilde{\delta} \in \mathrm{End}(E)$ by $\pi\delta = \tilde{\delta}\pi$.*[11] *Assume*

$$l \nmid r \text{ and } l \nmid \deg_{\mathrm{sep}}(\delta - \tilde{\delta}). \tag{5.1}$$

Take a generator B' of G'_l. Put $B := T\delta B'$. Then, $B \neq \mathcal{O}$.

Proof. Set $t := \mathrm{Tr}\,\pi$, the trace of the q-th power Frobenius endomorphism.[12] Put $C_rX + D_r := \mathrm{rem}(X^{r-1} + X^{r-2} + \cdots + 1, X^2 - tX + q)$ where rem stands for the remainder of polynomial division in $\mathbf{Z}[X]$. Now, assume $B = \mathcal{O}$. Since $\pi^{r-1} + \cdots + \pi + 1 = C_r\pi + D_r$ in $\mathrm{End}(E)$, we have

$$C_r\pi B' + D_rB' = \mathcal{O}, \tag{5.2}$$

$$C_r\pi\delta B' + D_r\delta B' = \mathcal{O}. \tag{5.3}$$

[10] The group G'_l is known as the trace zero subgroup of $E[l]$.

[11] The existence of $\tilde{\delta}$ follows from, for example, Silverman[33, Cor. II.2.12].

[12] Do not confuse a trace of an endomorphism with a trace endomorphism.

Recall that $\tilde{\delta} \in \mathrm{End}(E)$ satisfies $\pi\delta = \tilde{\delta}\pi$. Therefore $\tilde{\delta}C_r\pi B' + D_r\delta B' = \mathcal{O}$ by (5.3) and $-D_r\tilde{\delta}B + D_r\delta B' = D_r(\delta - \tilde{\delta})B' = \mathcal{O}$ by (5.2). Therefore either $l|D_r$ or $l|\deg_{\mathrm{sep}}(\delta - \tilde{\delta})$.

However, l does not divide D_r. By $l|\#E(\mathbf{F}_q) = 1 + q - t$, we see that $X^2 - tX + q = (X-1)(X-q)$ in $\mathbf{F}_l[X]$. There exists $U(X) \in \mathbf{F}_l[X]$ such that

$$X^{r-1} + \cdots + X + 1 = (X-1)(X-q)U(X) + C_rX + D_r. \tag{5.4}$$

By the assumption $r > 1$, we have $l|q^r - 1$ but $l \nmid q - 1$. Thus $q^{r-1} + q^{r-2} + \cdots + q + 1 = \frac{q^r-1}{q-1} \equiv 0 \bmod l$. Now, assume $l|D_r$. Evaluating (3) at $X = q$, we see that $qC_r = 0$ in $\mathbf{F}_l$. But E is supersingular, which means $l \nmid q$. Thus $C_r = 0$. Then, evaluating (5.4) at $X = 1$, we have $r = 0$ in $\mathbf{F}_l$, which contradicts the assumption $l \nmid r$. □

Remark 1. For the elliptic curves listed in Galbraith[11, Table IX.1] (so called "popular supersingular elliptic curves"), it is easy to verify that $\deg(\delta - \tilde{\delta})$ is either 1, 2 or 3.

We introduce the following problem:

Character-valued Pairing Inversion (CPI): Let G and H be cyclic groups of order l. Denote the H-valued character group of G by $\widehat{G}$. Find a group isomorphism $\varphi : H \to \widehat{G}$, the H-valued character of G such that for given $h \in H$ and $g \in G$, the computation of $\varphi(h)(g)$ is feasible.

At first glance, this problem looks awkward. Literally speaking, it is not inversion of e. But it maps H to something closely related to the domain of e. The CPI is equivalent to find a non-degenerate feasibly computable pairing $\mathcal{E} : \mu_l \times G \to \mu_l$.[13] Obviously, CPI is weaker than finding a feasible Verheul map whereas CPI seems not weaker than existence of feasible pairing.

Theorem 1. *Assume that l satisfies (5.1). Let B and B' be as in Lemma 1. Put $\zeta := e(B, B')$. If CPI and GPI are easy, WDHP on μ_l for generator ζ is easy.*

Proof. We use the same notation as in Lemma 1. Assume we have an feasibly computable non-degenerate pairing $\mathcal{E} : \mu_l \times G_l \to \mu_l$. Define $\tilde{\mathcal{E}} : \mu_l \times G_l \times G_l' \to \mu_l$ by $\tilde{\mathcal{E}}(z, U, V) := \mathcal{E}(\mathcal{E}(z, U), T\delta V)$ Put $\omega := \tilde{\mathcal{E}}(\zeta, B, B')$. There exists $u \in \mathbf{Z}$ satisfying $\mathcal{E}(z, B) = z^u$. We may not know the value of u but it holds that $\omega = \mathcal{E}(\mathcal{E}(\zeta, B), B) = \zeta^{u^2}$. Then for any $a, c \in \mathbf{Z}$ we have

$$\begin{aligned}\tilde{\mathcal{E}}(z, aB, cB') =& \mathcal{E}(\mathcal{E}(z, aB), T\delta(cB')) \\ =& \mathcal{E}(z^{au}, cB) = z^{acu^2}.\end{aligned}$$

For a given $\zeta \in \mu_l$, we can find P, $Q \in E[l]$ satisfying $e(P, Q) = \zeta^n$ by using GPI.

[13] In some sense, φ can be regarded a "bidirectional" map. The author hopes that CPI acts as a path to the Verheul map from a pairing.

Since B and B' form a base of $E[l]$, there are a, b, c, $d \in \mathbf{F}_l$ satisfying $P = aB + bB'$, $Q = cB + dB'$. Observe that

$$\zeta^n = e(P,Q) = e(B,B')^{ad-bc} = \zeta^{ad-bc},$$

$$\tilde{\mathcal{E}}(\zeta^m, TP, T'Q)\tilde{\mathcal{E}}(\zeta^m, TQ, T'P)^{-1} = \zeta^{m(ad-bc)u^2} = \omega^{mn}.$$

Hence, WDHP for the generator ζ is easy. □

Since $E[l]$ is non-cyclic, FPI does not immediately give the Verheul map in our situation. A key point of the proof of the next Theorem is that P is given in terms of B and B'.

Theorem 2. *Assume that l satisfies (5.1). Let a, $b \in \mathbf{F}_l$. Assume that either a or b is non-zero and put $P := aB + bB'$. If FPI for P is easy, then there exists a feasible Verheul map. In particular, WHDP for μ_l is easy.*

Proof. We keep the notation above. We can find $Q_z \in E[l]$ satisfying $e(P, Q_z) = z$. In case of $a = 0$, a map defined by $z \to TQ_z$ is the Verheul map because $e(P,R) = e(P,TR)$ for any $R \in E[l]$. Similarly, $z \to T'Q_z$ is the Verheul map in case of $b = 0$. Assume $ab \neq 0$. Put $\nu := (\delta - \tilde{\delta})^2 \in \mathrm{End}(E)$. Then ν is in fact a multiplication by a non-zero integer, say n. By an argument similar to the proof of Lemma 1, we see $D_r \equiv r \pmod{l}$. Observe

$$\begin{aligned} T\delta =& \tilde{\delta}T + r^{-1}(\delta - \tilde{\delta})D_r, \\ T'\delta =& \tilde{\delta}(1-T) + (\delta - \tilde{\delta})(1 - r^{-1}D_r). \end{aligned}$$

Therefore

$$T'\delta T\delta = (\delta - \tilde{\delta})(1 - r^{-1}D_r)(\tilde{\delta}T + r^{-1}(\delta - \tilde{\delta})D_r).$$

Since B' is a generator of G'_l, we see that $TB' = \mathcal{O}$. Thus

$$T'\delta T\delta B' = (\delta - \tilde{\delta})^2 r^{-2}(r - D_r)D_r B'.$$

Define $\psi \in \mathrm{End}(E)$ by $\psi := n^{-2}r^2(r - D_r)^{-1}D_r^{-1}T'\delta T\delta$. With this definition, $\psi(B) = B'$.

Now there exists c, $d \in \mathbf{F}_l$ satisfying $Q_z = cB + dB'$. By the alternating property of the Weil pairing,

$$\begin{aligned} z =& e(TP, T'Q)e(T'P, TQ) = e(aB, dB')e(bB', cB) \\ =& e(bB, ab^{-1}T'Q)e(bB, \psi(TQ))^{-1}. \end{aligned}$$

Thus, $z \to ab^{-1}T'Q - \psi(TQ)$ is a group homomorphism, which is the desired one. □

References

1. Balasubramanian, R., Koblitz, N.: The improbability that an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm. J. Cryptology 11, 141–145 (1998)
2. Barreto, P., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithm for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
3. Barreto, P.S.L.M., Galbraith, S.D., Ó'hÉigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular Abelian varieties. Des. Codes Crypt. 42, 239–271 (2007)
4. 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)
5. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Des. Codes Crypt. 37, 133–141 (2005)
6. Cocks, C.: Identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
7. Duursma, I., Lee, H.: Tate pairing implementation for hyperellipitc curves $y^2 = x^p - x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
8. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory 31, 469–472 (1985)
9. Freeman, D.: Constructing pairing-friendly elliptic curves with embedding degree 10. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory. LNCS, vol. 4076, pp. 452–465. Springer, Heidelberg (2006)
10. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. IACR e-print 2006/372 (2006)
11. Galbraith, S.: Pairings. Advances in elliptic curve cryptography, Chap. 9. In: Blake, I.F., Seroussi, G., Smart, N.P. (eds.) London math. soc. lect. note series, vol. 317, Cambridge University Press, Cambridge (2005)
12. Galbraith, S., Hess, F., Vercautern, F.: Aspects of Pairing inversion, manuscript (2007)
13. Galbraith, S.D.: ÓhÉigeartaigh, C., Sheedy, C.: Simplified pairing computation and security implications. J. Math. Crypt. (to appear)
14. Hess, F., Smart, N.P., Vercauteren, F.: The eta pairing revisited. IEEE trans. on IT 52, 4995–4602 (2006)
15. Hitt, L.: On the minimal embedding field. In: This volume (2007)
16. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: Bosma, W. (ed.) Algorithmic Number Theory 4. LNCS, vol. 1838, pp. 385–393. Springer, Heidelberg (2000)
17. Joux, A.: The Weil and Tate pairings as building blocks for public key cryptosystems (survey). In: Fieker, C., Kohel, D.R. (eds.) Algorithmic Number Theory. LNCS, vol. 2369, pp. 20–32. Springer, Heidelberg (2002)
18. Kiltz, E., Winterhof, A.: On the interpolation of bivariate polynomials related to the Diffie-Hellman mapping. Bull. Austral. Math. Soc. 69, 305–315 (2004)
19. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)

20. Lange, T.: Open problems in pairing. In: Slides at IPAM workshop Number theory and cryptography – open problems (2006) available from https://www.ipam.ucla.edu/programs/scws1/
21. Lange, T., Winterhof, A.: Polynomial interpolation of the elliptic curve and XTR discrete logarithm. In: Ibarra, O.H., Zhang, L. (eds.) COCOON 2002. LNCS, vol. 2387, pp. 137–143. Springer, Heidelberg (2002)
22. Lange, T., Winterhof, A.: Interpolation of the discrete logarithm in F_q by boolean functions and by polynomials in several variables modulo a divisor of $q-1$. Discrete Appl. Math. 128, 193–206 (2003)
23. Luca, F., Shparlinski, I.: Elliptic curves with low embedding degree. J. Cryptology 19, 553–562 (2006)
24. Maurer, U.M., Wolf, S.: The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. SIAM J. Comput. 28, 1689–1721 (1999)
25. Maurer, U.M., Wolf, S.: The Diffie-Hellman protocol. Des. Codes Cryptogr. 19, 147–171 (2000)
26. Menezes, A., Vanstone, S., Okamoto, T.: Reducing elliptic curve logarithms to logarithms in a finite field. In: Proc. 23rd annual ACM symp. on theory of computing, pp. 80–89. ACM press, New York (1991)
27. Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. Info. Theory 39, 1639–1646 (1993)
28. Miller, V.: Short programs for functions on curves (1986) preprint available at http://crypto.stanford.edu/miller/miller.pdf
29. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Trans. Fundamentals E84, 1234–1243 (2002)
30. Satoh, T.: On degrees of polynomial interpolations releated to elliptic curves. In: Ytrehus, Ø. (ed.) WCC 2005. LNCS, vol. 3969, pp. 155–163. Springer, Heidelberg (2006)
31. Satoh, T.: On polynomial interpolations related to Verheul homomorphisms. LMS J. Comput. Math. 9, 135–158 (2006)
32. Semaev, I.A.: Bystryĭ algorytm bychisleniya sparivaniya A. Veĭlya na èllipticheskikh kribykh (A fast algorithm for computing the Weil pairing on elliptic curves). Mezhdu narodnaya konferentsiya Sovremennye problemy teorii chisel, Rossiya, Tula, 20 sentyabrya – 25 sentyabrya 1993 goda. In: International conference Modern Problems in Number Theory, Russia, Tula, September 20 – September 25, 1993, 142 (abstract, in russian) (1993)
33. Silverman, J.H.: The arithmetic of elliptic curves. GTM, p. 106. Springer, Heidelberg (1985)
34. Verheul, E.R.: Evidence that XTR is more secure than supersingular elliptic curve cryptosystem. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 195–210. Springer, Heidelberg (2001)

The Tate Pairing Via Elliptic Nets

Katherine E. Stange

Brown University, Providence, RI 02912, USA

Abstract. We derive a new algorithm for computing the Tate pairing on an elliptic curve over a finite field. The algorithm uses a generalisation of elliptic divisibility sequences known as elliptic nets, which are maps from $\mathbb{Z}^n$ to a ring that satisfy a certain recurrence relation. We explain how an elliptic net is associated to an elliptic curve and reflects its group structure. Then we give a formula for the Tate pairing in terms of values of the net. Using the recurrence relation we can calculate these values in linear time. Computing the Tate pairing is the bottleneck to efficient pairing-based cryptography. The new algorithm has time complexity comparable to Miller's algorithm, and should yield to further optimisation.

Keywords: Tate pairing, elliptic curve, elliptic divisibility sequence, elliptic net, Miller's algorithm, pairing-based cryptography.

1 Introduction

The use of pairings in elliptic curve cryptography was originally suggested as a means of reducing the discrete logarithm problem on an elliptic curve to the discrete logarithm problem on a finite field [1, 2], but considerable excitement and research has since been generated by public-key cryptographic applications such as Sakai, Ohgishi and Kasahara's key agreement and signature schemes [3], Joux's tri-partite Diffie-Hellman key exchange [4], and Boneh and Franklin's identity-based encryption scheme [5]. Good overviews of the research include [6, 7], while a very up-to-date research bibliography can be found at [8].

The bottleneck to pairing-based cryptographic implementations is the costly computation of the pairing, which is most frequently the Tate or Weil pairing, the former usually being more efficient. Currently, the only polynomial time algorithm is due to Victor Miller [9] (for an overview of implementions, see [10, 11]).

In this paper, we propose a new method of computing of the Tate pairing, arising from the theory of elliptic nets.

Elliptic nets are a generalisation of elliptic divisibility sequences, which were first studied by Morgan Ward in 1948 [12]. These are integer sequences h_0, h_1, h_2, ..., h_n, ... satisfying the following two properties:

1. For all positive integers $m > n$,

$$h_{m+n}h_{m-n} = h_{m+1}h_{m-1}h_n^2 - h_{n+1}h_{n-1}h_m^2 \ . \tag{1}$$

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 329–348, 2007.

2. h_n divides h_m whenever n divides m.

Ward demonstrates that an elliptic divisibility sequence arises from any choice of elliptic curve and rational point on that curve.

Theorem 1 (M. Ward, 1948, [12]). *Suppose E is an elliptic curve defined over $\mathbb{Q}$, $\sigma : \mathbb{C} \to \mathbb{C}$ is its Weierstrass sigma function, and $u \in \mathbb{C}$ corresponds to a rational point on E. Then there exists an integer k such that the sequence*

$$h_n := k^{n^2-1} \frac{\sigma(nu)}{\sigma(u)^{n^2}}$$

forms an elliptic divisibility sequence.

For an overview of research on elliptic divisibility sequences, see [13].

Given an integral domain R and a finitely generated free abelian group A, an *elliptic net* is a map $W : A \to R$ satisfying the following recurrence relation for $p, q, r, s \in A$:

$$\begin{aligned} &W(p+q+s)W(p-q)W(r+s)W(r) \\ &\quad + W(q+r+s)W(q-r)W(p+s)W(p) \\ &\quad + W(r+p+s)W(r-p)W(q+s)W(q) = 0 \ . \end{aligned}$$

When $A = R = \mathbb{Z}$ and $W(1) = 1$, the positive terms of an elliptic net satisfy Ward's equation (1) above. Under the further conditions that $W(2)|W(4)$ and $W(0) = 0$, these terms form an elliptic divisibility sequence.

Theorem 2 in Sect. 2 relates elliptic nets over $R = \mathbb{C}$ to elliptic curves, generalising Theorem 1. However, for cryptographic applications it is desired to work over finite fields: Theorem 3 allows results over $\mathbb{C}$ to be carried over to the finite field case. Theorem 4 is the statement of the curve-net relationship over finite fields.

According to these results, we can associate to any choice of curve E defined over a finite field K and n points $P_i \in E(K)$ an elliptic net

$$W_{E,P_1,\ldots,P_n} : \mathbb{Z}^n \to K \ .$$

This net can then be used to compute the Tate pairing: the main result can be stated as follows.

Theorem (Introductory Version of Theorem 6). *Fix a positive $m \in \mathbb{Z}$. Let E be an elliptic curve defined over a finite field K containing the m-th roots of unity. Let P, $Q \in E(K)$, with $[m]P = \mathcal{O}$. Choose $S \in E(K)$ such that $S \notin \{\mathcal{O}, -Q\}$. Then there exists an elliptic net $W : \mathbb{Z}^n \to K$ and $\mathbf{p}, \mathbf{q}, \mathbf{s} \in \mathbb{Z}^n$ such that the quantity*

$$T_m(P, Q) = \frac{W(\mathbf{s} + m\mathbf{p} + \mathbf{q})W(\mathbf{s})}{W(\mathbf{s} + m\mathbf{p})W(\mathbf{s} + \mathbf{q})}$$

is exactly the Tate pairing $T_m = \tau_m : E(K)[m] \times E(K)/mE(K) \to K^*/(K^*)^m$.

From Theorem 6, to calculate the Tate pairing efficiently only requires an efficient method of calculating the terms of an elliptic net. Rachel Shipsey's thesis provides a double-and-add method of calculating the n-th term of an elliptic divisibility sequence in $\log n$ time [14]. We generalise her algorithm to elliptic nets in Sect. 4.

This application is an example of doing arithmetic on elliptic curves via the arithmetic of elliptic nets. Rachel Shipsey's work made use of this approach to solve the elliptic curve discrete logarithm problem in certain cases. Her paradigm may have many other fruitful applications.

The Elliptic Net Algorithm and Miller's algorithm are both $log(n)$ algorithms; the difference is in the constants. In this nascent form, the Elliptic Net Algorithm is only somewhat slower than an optimised Miller's, especially at higher embedding degrees. This note should be considered a call to further research.

Guide to the Reader. I give substantial mathematical background in Sect. 2, which is currently unavailable elsewhere, and will be necessary for any improvements and new applications. The entirely theory-averse can skip the preliminaries. For most, a suggested path is Sect. 2.2 with reference to Definition 1, followed by Sect. 2.4 and 2.5. The proof of Theorem 3 is omitted for lack of space: for this and more details, see [15].

In Sect. 3, we prove Theorem 6 and a corollary relating elliptic nets and the Tate pairing. In Sect. 4, we describe the algorithms necessary to compute elliptic nets, and therefore the Tate pairing, efficiently. In Sect. 5, we make some brief remarks on optimisation of the algorithms and the efficiency as compared with Miller's algorithm. Finally, we make some concluding remarks in Sect. 6.

2 Mathematical Preliminaries

2.1 Elliptic Functions $\Psi_{\mathbf{v}}$

Elliptic Curves Over $\mathbb{C}$. We begin with some complex function theory which will be necessary for the definition of elliptic nets over $\mathbb{C}$. This material is covered in, for example, [16, 17]. For a complex lattice Λ, define the Weierstrass sigma function $\sigma : \mathbb{C} \to \mathbb{C}$ by

$$\sigma(z;\Lambda) = z \prod_{\substack{\omega\in\Lambda \\ \omega\neq 0}} \left(1 - \frac{z}{\omega}\right) e^{z/\omega + (1/2)(z/\omega)^2} \quad ,$$

and the Weierstrass zeta function $\zeta : \mathbb{C} \to \mathbb{C}$ by

$$\zeta(z;\Lambda) = \frac{1}{z^2} + \sum_{\substack{\omega\in\Lambda \\ \omega\neq 0}} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2}\right) \quad .$$

Recall that the quantity

$$\zeta(z+\omega;\Lambda)-\zeta(z;\Lambda)$$

is independent of z, and we call this $\eta(\omega)$. The map $\eta:\Lambda\to\mathbb{C}$ is called the quasi-period homomorphism. Define $\lambda:\Lambda\to\{\pm 1\}$ by

$$\lambda(\omega)=\begin{cases}1 & \text{if } \omega\in 2\Lambda\ ,\\ -1 & \text{if } \omega\notin 2\Lambda\ .\end{cases}$$

Recall that the Weierstrass sigma function $\sigma:\mathbb{C}\to\mathbb{C}$ satisfies the following transformation formula for all $z\in\mathbb{C}$ and $\omega\in\Lambda$:

$$\sigma(z+\omega;\Lambda)=\lambda(\omega)e^{\eta(\omega)(z+\frac{1}{2}\omega)}\sigma(z;\Lambda)\ . \tag{2}$$

Functions $\Psi_{\mathbf{v}}$. We now define the functions which will be used to obtain an elliptic net from an elliptic curve, and collect a few basic results for later reference.

Definition 1. *Fix a lattice $\Lambda\in\mathbb{C}$ corresponding to an elliptic curve E. For $\mathbf{v}=(v_1,\ldots,v_n)\in\mathbb{Z}^n$, define a function $\Psi_{\mathbf{v}}$ on $\mathbb{C}^n$ in variables $\mathbf{z}=(z_1,\ldots,z_n)$ as follows:*

$$\Psi_{\mathbf{v}}(\mathbf{z};\Lambda)=\frac{\sigma(v_1z_1+\ldots+v_nz_n;\Lambda)}{\displaystyle\prod_{i=1}^{n}\sigma(z_i;\Lambda)^{2v_i^2-\sum_{j=1}^{n}v_iv_j}\prod_{1\le i<j\le n}\sigma(z_i+z_j;\Lambda)^{v_iv_j}}\ .$$

In particular, we have for each $n\in\mathbb{Z}$, a function Ψ_n on $\mathbb{C}$ in the variable z:

$$\Psi_n(z;\Lambda)=\frac{\sigma(nz;\Lambda)}{\sigma(z;\Lambda)^{n^2}}\ ,$$

and for each pair $(m,n)\in\mathbb{Z}\times\mathbb{Z}$, a function $\Psi_{m,n}$ on $\mathbb{C}\times\mathbb{C}$ in variables z and w:

$$\Psi_{m,n}(z,w;\Lambda)=\frac{\sigma(mz+nw;\Lambda)}{\sigma(z;\Lambda)^{m^2-mn}\sigma(z+w;\Lambda)^{mn}\sigma(w;\Lambda)^{n^2-mn}}\ .$$

From the general theory of elliptic functions, the divisor of $\Psi_{\mathbf{v}}$ as a function of z_1 is

$$\left(\sum_{j=2}^{n}[-v_j]z_j\right)-\sum_{j=2}^{n}v_1v_j(-z_j)-\left(v_1^2-\sum_{j=2}^{n}v_1v_j\right)(0)\ . \tag{3}$$

Proposition 1. *The functions $\Psi_{\mathbf{v}}$ are elliptic functions in each variable.*

Proof. Let $\omega\in\Lambda$. We show the function is elliptic in the first variable. Let $\mathbf{v}=(v_1,\ldots,v_n)\in\mathbb{Z}^n$ and $\mathbf{z}=(z_1,\ldots,z_n),\mathbf{w}=(\omega,0,\ldots,0)\in\mathbb{C}^n$. Using (2), we calculate

$$F=\frac{\Psi_{\mathbf{v}}(\mathbf{z}+\mathbf{w};\Lambda)}{\Psi_{\mathbf{v}}(\mathbf{z};\Lambda)}=\frac{\lambda(v_1\omega)}{\lambda(\omega)^{v_1^2}}\ .$$

If $\omega, v_1\omega \notin 2\Lambda$, then v_1 is odd, and $F = 1$. If $\omega \notin 2\Lambda$ but $v_1\omega \in 2\Lambda$, then v_1 must be even, and so $F = 1$ again. Finally, if $\omega \in 2\Lambda$, then $v_1\omega \in 2\Lambda$, and $F = 1$. Thus $\Psi_{\mathbf{v}}$ is invariant under adding a period to the variable z_1. Similarly $\Psi_{\mathbf{v}}$ is elliptic in each variable on $\mathbb{C}^n$. □

In view of this proposition, we will use the same notation $\Psi_{\mathbf{v}}$ for the associated map $E^n \to \mathbb{C}$, and write, for example, $\Psi_{m,n}(P_1, P_2; E)$.

Proposition 2. *Fix a lattice $\Lambda \subset \mathbb{C}$ corresponding to an elliptic curve. Let $\mathbf{v} \in \mathbb{Z}^n$ and $\mathbf{z} \in \mathbb{C}^n$. Let T be an $n \times n$ matrix with entries in $\mathbb{Z}$ and transpose T^{tr}. Then*

$$\Psi_{\mathbf{v}}(T^{tr}(\mathbf{z}); \Lambda) = \frac{\Psi_{T(\mathbf{v})}(\mathbf{z}; \Lambda)}{\displaystyle\prod_{i=1}^{n} \Psi_{T(\mathbf{e}_i)}(\mathbf{z}; \Lambda)^{2v_i^2 - \sum_{j=1}^{n} v_i v_j} \prod_{1 \le i < j \le n} \Psi_{T(\mathbf{e}_i + \mathbf{e}_j)}(\mathbf{z}; \Lambda)^{v_i v_j}} \quad . \tag{4}$$

Proof. A straightforward calculation using (2). □

2.2 Elliptic Nets from Elliptic Curves

Definition 2. *Let A be a finitely generated free abelian group, and R be an integral domain. An* elliptic net *is any map $W : A \to R$ such that the following recurrence holds for all p, q, r, $s \in A$:*

$$\begin{aligned} &W(p+q+s)W(p-q)W(r+s)W(r) \\ &\quad + W(q+r+s)W(q-r)W(p+s)W(p) \\ &\quad + W(r+p+s)W(r-p)W(q+s)W(q) = 0 \quad . \end{aligned} \tag{5}$$

The set of such nets is denoted $\mathcal{EN}(A, R)$. If B is a subgroup of A, then W restricted to B is also an elliptic net and is called an elliptic subnet of A.

Proposition 3. *Let $W : A \to R$ be an elliptic net. Then $W(-z) = -W(z)$ for any $z \in A$. In particular $W(0) = 0$.*

Proof. If $W(-z) = W(z) = 0$, we are done. If not, then without loss of generality, assume $W(z) \neq 0$. Then setting $p = q = z, r = s = 0$ in (5), we obtain $0 + W(z)^4 + W(z)^3 W(-z) = 0$, whence $W(-z) = -W(z)$. □

Work of Christine Swart [18] and van der Poorten [19] on translated elliptic divisibility sequences provided the clues that the theory of elliptic nets existed. It has recently come to my attention that the possibility of such a definition was briefly discussed in correspondence by Noam Elkies, James Propp and Michael Somos in 2001 [20].

We will now see that the $\Psi_{\mathbf{v}}$ form an elliptic net as a function of $\mathbf{v} \in \mathbb{Z}^n$ when the curve E and points $P_1, \ldots, P_n$ are fixed. Let the standard basis of $\mathbb{Z}^n$ be denoted $\mathbf{e}_1, \ldots, \mathbf{e}_n$. As a means of fixing n points P_i, we specify a homomorphism $\phi : \mathbb{Z}^n \to E$.

Definition 3. *Fix an elliptic curve E. Suppose $\phi : \mathbb{Z}^n \to E$ is a homomorphism such that the images of $\pm\mathbf{e}_i$ under ϕ are all distinct and nonzero. Define $W_\phi : \mathbb{Z}^n \to \mathbb{C}$ by*

$$W_\phi(\mathbf{v}) = \Psi_{\mathbf{v}}(\phi(\mathbf{e}_1), \phi(\mathbf{e}_2), \ldots, \phi(\mathbf{e}_n); E) \ .$$

Theorem 2. *W_ϕ is an elliptic net.*

We will prove Theorem 2 in the next section.

Suppose we choose n points P_i of an elliptic curve E, such that the $\pm P_i$ are all distinct and nonzero. Define $\phi : \mathbb{Z}^n \to E$ by $\phi(\mathbf{e}_i) = P_i$. We call $W_\phi \in \mathcal{EN}(\mathbb{Z}^n, \mathbb{C})$ the *elliptic net associated to* $E, P_1, \ldots, P_n$. In fact, $W_\phi \in \mathcal{EN}(\mathbb{Z}^n, L)$ where L is the field of definition of the P_i. Part of the first quadrant of such an example net is shown in Fig. 1 at left. In this example, $E : y^2 + y = x^3 + x^2 - 2x$, $P = (0, 0)$, $Q = (1, 0)$, and $L = \mathbb{Q}$. For example, $W(3, 2) = -13$.

	4335	5959	12016	-55287	23921	1587077	-7159461
	94	479	919	-2591	13751	68428	424345
	-31	53	-33	-350	493	6627	48191
	-5	8	-19	-41	-151	989	-1466
	1	3	-1	-13	-36	181	-1535
	1	1	2	-5	7	89	-149
$\uparrow Q$	0	1	1	-3	11	38	249
	$P \rightarrow$				over $\mathbb{Q}$		

	0	4	4	3	1	2	4
	4	4	4	4	1	3	0
	4	3	2	0	3	2	1
	0	3	1	4	4	4	4
	1	3	4	2	4	1	0
	1	1	2	0	2	4	1
$\uparrow Q$	0	1	1	2	1	3	4
	$P \rightarrow$				over $\mathbb{F}_5$		

Fig. 1. Portion of the elliptic net of $E : y^2 + y = x^3 + x^2 - 2x$, $P = (0, 0)$, $Q = (1, 0)$

Let E be an elliptic curve defined over $\mathbb{Q}$, and $P \in E(\mathbb{Q})$. Then if the positive terms of the elliptic net associated to E, P are integers, they form an elliptic divisibility sequence as described by Ward. In particular, the recurrence relation (5) implies Ward's relation (1). For example, in Fig. 1, the bottom row is the elliptic divisibility sequence associated to P: 0, 1, 1, -3, 11, 38, 249, ...

A word of caution: it is not appropriate to think of elliptic nets as maps on the points of the curve. This can lead to two misconceptions. First, although it is tempting in this example to think of -13 as the "number associated to $3P + 2Q$", this depends on the choice of "basis" P, Q of the net. That is to say, if we consider instead the net W' associated to $E, P + Q, P$, then $W'(2, 1)$ is **not** equal to $W(3, 2)$. The relationship between the nets relative to different bases on a single curve is the content of Proposition 2. Further, even having restricted our attention to exactly one net we may be surprised. Suppose W is an elliptic net associated to E, P where P is an m-torsion point. We cannot expect $W(m + k)$ to equal $W(k)$ in general. We will address these crucial issues in Sect. 2.5.

2.3 Proof of Theorem 2

Proof. We will make use of the well-known elliptic function identity

$$\wp(a) - \wp(b) = -\frac{\sigma(a+b)\sigma(a-b)}{\sigma(a)^2\sigma(b)^2} \quad . \tag{6}$$

First, we show that

$$\zeta(x+a) - \zeta(a) - \zeta(x+b) + \zeta(b) = \frac{\sigma(x+a+b)\sigma(x)\sigma(a-b)}{\sigma(x+a)\sigma(x+b)\sigma(a)\sigma(b)} \quad . \tag{7}$$

Denote by f and g the left and right side of (7) respectively. Suppose that $a, b \notin \Lambda$. The functions f and g are elliptic in x. Both f and g have single poles at $-a$ and $-b$ only. The zeroes of g are at $-a-b$ and 0. These are also zeroes of f, since ζ is an odd function. Hence we have $f = cg$ for some c not depending on x. Now define instead

$$\begin{aligned} F &= (\zeta(x+a) - \zeta(a) - \zeta(x+b) + \zeta(b))\,\sigma(x+a)\sigma(x+b) \quad , \\ G &= \sigma(x+a+b)\sigma(x) \quad . \end{aligned}$$

We have $F = c'G$ for some constant c' independent of x. Taking derivatives and evaluating at $x = 0$, we have

$$(\wp(a) - \wp(b))\,\sigma(a)\sigma(b) = c'\sigma(a+b)\sigma'(0) \quad .$$

We have $\sigma'(0) = 1$. By (6),

$$c' = -\frac{\sigma(a-b)}{\sigma(a)\sigma(b)} \quad .$$

which proves (7).

Fix $\mathbf{z} \in \mathbb{C}^n$. We will show that the values $\Psi_{\mathbf{v}}(\mathbf{z}; \Lambda)$ for $\mathbf{v} \in \mathbb{Z}^n$ form an elliptic net. For notational simplicity, we drop the arguments $(\mathbf{z}; \Lambda)$ and also write $\sigma(\mathbf{v})$, $\wp(\mathbf{v})$ and $\zeta(\mathbf{v})$ for $\sigma(\mathbf{v} \cdot \mathbf{z})$, $\wp(\mathbf{v} \cdot \mathbf{z})$ and $\zeta(\mathbf{v} \cdot \mathbf{z})$. We observe that $\mathbf{v} = \mathbf{0}$ if and only if $\Psi_{\mathbf{v}} \equiv 0$.

If any of $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$, $\mathbf{p}+\mathbf{s}$, $\mathbf{q}+\mathbf{s}$, or $\mathbf{r}+\mathbf{s}$ are zero, then the recurrence relation (5) holds trivially. So we may assume none of $\Psi_{\mathbf{p}}$, $\Psi_{\mathbf{q}}$, $\Psi_{\mathbf{r}}$, $\Psi_{\mathbf{p}+\mathbf{s}}$, $\Psi_{\mathbf{q}+\mathbf{s}}$, or $\Psi_{\mathbf{r}+\mathbf{s}}$ is identically zero.

For any quadratic form f defined on $\mathbb{Z}^n$, we have the following relation for all $\mathbf{p}, \mathbf{q}, \mathbf{s} \in \mathbb{Z}^n$:

$$f(\mathbf{p}+\mathbf{q}+\mathbf{s}) + f(\mathbf{p}-\mathbf{q}) + f(\mathbf{s}) - f(\mathbf{p}+\mathbf{s}) - f(\mathbf{p}) - f(\mathbf{q}+\mathbf{s}) - f(\mathbf{q}) = 0 \quad . \tag{8}$$

Suppose that $\mathbf{s} \neq \mathbf{0}$ and so $\Psi_{\mathbf{s}} \not\equiv 0$. By (8) and (7),

$$\frac{\Psi_{\mathbf{p}+\mathbf{q}+\mathbf{s}}\Psi_{\mathbf{p}-\mathbf{q}}\Psi_{\mathbf{s}}}{\Psi_{\mathbf{p}+\mathbf{s}}\Psi_{\mathbf{p}}\Psi_{\mathbf{q}+\mathbf{s}}\Psi_{\mathbf{q}}} = \frac{\sigma(\mathbf{p}+\mathbf{q}+\mathbf{s})\sigma(\mathbf{p}-\mathbf{q})\sigma(\mathbf{s})}{\sigma(\mathbf{p}+\mathbf{s})\sigma(\mathbf{p})\sigma(\mathbf{q}+\mathbf{s})\sigma(\mathbf{q})} = \zeta(\mathbf{p}+\mathbf{s}) - \zeta(\mathbf{p}) - \zeta(\mathbf{q}+\mathbf{s}) + \zeta(\mathbf{q}) \quad .$$

Therefore, we have

$$\frac{\Psi_{\mathbf{p+q+s}}\Psi_{\mathbf{p-q}}\Psi_{\mathbf{s}}}{\Psi_{\mathbf{p+s}}\Psi_{\mathbf{p}}\Psi_{\mathbf{q+s}}\Psi_{\mathbf{q}}} + \frac{\Psi_{\mathbf{q+r+s}}\Psi_{\mathbf{q-r}}\Psi_{\mathbf{s}}}{\Psi_{\mathbf{q+s}}\Psi_{\mathbf{q}}\Psi_{\mathbf{r+s}}\Psi_{\mathbf{r}}} + \frac{\Psi_{\mathbf{r+p+s}}\Psi_{\mathbf{r-p}}\Psi_{\mathbf{s}}}{\Psi_{\mathbf{r+s}}\Psi_{\mathbf{r}}\Psi_{\mathbf{p+s}}\Psi_{\mathbf{p}}} = 0 \ ,$$

or, more simply,

$$\Psi_{\mathbf{p+q+s}}\Psi_{\mathbf{p-q}}\Psi_{\mathbf{r+s}}\Psi_{\mathbf{r}} + \Psi_{\mathbf{q+r+s}}\Psi_{\mathbf{q-r}}\Psi_{\mathbf{p+s}}\Psi_{\mathbf{p}} + \Psi_{\mathbf{r+p+s}}\Psi_{\mathbf{r-p}}\Psi_{\mathbf{q+s}}\Psi_{\mathbf{q}} = 0 \ ,$$

which is what was required to prove.

The case $\mathbf{s} = 0$ is done similarly, using

$$\frac{\Psi_{\mathbf{p+q}}\Psi_{\mathbf{p-q}}}{\Psi_{\mathbf{p}}^2\Psi_{\mathbf{q}}^2} = \frac{\sigma(\mathbf{p}+\mathbf{q})\sigma(\mathbf{p}-\mathbf{q})}{\sigma(\mathbf{p})^2\sigma(\mathbf{q})^2} = \wp(\mathbf{q}) - \wp(\mathbf{p}) \ . \qquad \square$$

2.4 Moving to Finite Fields

Some Notation. Now is a good moment to collect the relevant notation for the next section and the remainder of the paper.

L	number field contained in $\mathbb{C}$
E_L	elliptic curve defined over L
R	ring of integers of L
$\mathfrak{p}$	prime of R of good reduction for E_L
$k_\mathfrak{p}$	residue field of $\mathfrak{p}$
$E_{k_\mathfrak{p}}$	E_L reduced modulo $\mathfrak{p}$
$\delta : E_L(L) \to E_{k_\mathfrak{p}}(k_\mathfrak{p})$	reduction map modulo $\mathfrak{p}$
$\delta : \mathbb{P}^1(L) \to \mathbb{P}^1(k_\mathfrak{p})$	reduction map modulo $\mathfrak{p}$

Reduction Modulo $\mathfrak{p}$. We wish to extend the relationship between nets and curves to finite fields, but we can no longer use Weierstrass' sigma function to define appropriate functions. The following theorem allows us to push results on number fields L over to residue fields $k_\mathfrak{p}$. It says that we can find the appropriate functions $\Omega_\mathbf{v}$ for $E_{k_\mathfrak{p}}$ by simply considering the net $\Psi_\mathbf{v}$ modulo $\mathfrak{p}$. These $\Omega_\mathbf{v}$ will also form an elliptic net.

Theorem 3. *Consider points $P_1, \ldots, P_n \in E_L(L)$ such that the reductions modulo $\mathfrak{p}$ of the $\pm P_i$ are all distinct and nonzero. Then for each $\mathbf{v} \in \mathbb{Z}^n$ there exists a function $\Omega_\mathbf{v}$ such that the following diagram commutes:*

$$\begin{array}{ccc} E_L^n(L) & \xrightarrow{\Psi_\mathbf{v}} & \mathbb{P}^1(L) \\ \downarrow{\scriptstyle \delta} & & \downarrow{\scriptstyle \delta} \\ E_{k_\mathfrak{p}}^n(k_\mathfrak{p}) & \xrightarrow{\Omega_\mathbf{v}} & \mathbb{P}^1(k_\mathfrak{p}) \end{array}$$

Furthermore $\operatorname{div}(\Omega_\mathbf{v}) = \delta^* \operatorname{div}(\Psi_\mathbf{v})$.

Proof (Sketch). Consider E^n as a scheme over $\operatorname{Spec} R$. The proof requires extending the function on the generic fibre to the special fibres. The difficulty lies in showing that the resulting function does not have any vertical fibres in the support of its divisor. This reduces to a statement about the form of the $\Psi_{\mathbf{v}}$ as polynomials in the structure sheaf. It relies on a number of nested and complicated inductive proofs. See [15]. □

In light of this, we extend Definition 3 and Theorem 2.

Definition 4. *Let $\phi : \mathbb{Z}^n \to E_{k_\mathfrak{p}}$ be a homomorphism such that the images of $\pm\mathbf{e}_i$ under ϕ are all distinct and nonzero. Let $\Omega_{\mathbf{v}}$ be defined according to Theorem 3. Define $W_\phi : \mathbb{Z}^n \to k_\mathfrak{p}$ by*

$$W_\phi(\mathbf{v}) = \Omega_{\mathbf{v}}(\phi(\mathbf{e}_1), \phi(\mathbf{e}_2), \ldots, \phi(\mathbf{e}_n)) \ .$$

Theorem 4. *Suppose K is either a number field or a finite field, and E is an elliptic curve defined over K. Let $\phi : \mathbb{Z}^n \to E(K)$ be a homomorphism. Then W_ϕ is an elliptic net.*

Proof. If K is a number field, this is Theorem 2. If K is a finite field, then this statement follows from Theorem 3: an elliptic net postcomposed with a homomorphism is still an elliptic net. □

Figure 1 illustrates the relationship between an example elliptic net associated to E, P, Q over $\mathbb{Q}$ and the elliptic net associated to their reductions modulo 5.

2.5 Equivalence of Nets

In this section, we restrict ourselves to finite fields.

Definition 5. *Let $W_1, W_2 \in \mathcal{EN}(A, K)$. Suppose $\alpha, \beta \in K^*$, and $f : A \to \mathbb{Z}$ is a quadratic form. If*

$$W_1(\mathbf{v}) = \alpha\beta^{f(\mathbf{v})} W_2(\mathbf{v})$$

for all $\mathbf{v}$, then we say W_1 is equivalent to *W_2 and write $W_1 \sim W_2$.*

Clearly this definition gives an equivalence relation, and it is easily verified that an equivalence applied to an elliptic net gives another elliptic net. We write

$$\mathcal{EN}_0(A, K) = \mathcal{EN}(A, K)/\sim \ .$$

If W_1 is a subnet of W_2, then we may, by abuse of language, say that the equivalence class $[W_1]$ is a subnet of the equivalence class $[W_2]$, since then any $W_1' \in [W_1]$ will be equivalent to some subnet of any $W_2' \in [W_2]$.

Recall the discussion at the end of Sect. 2.2. There, we encountered two reasons that we cannot consider an elliptic net W to be a map on the group $E(K)$ itself. The first is that a basis must be chosen, and the second is

that the net may take different values at two vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{Z}^n$ even when $\phi(\mathbf{v}_1) = \phi(\mathbf{v}_2) \in E(K)$.[1]

All is not lost, however. We can define elliptic nets on a free abelian cover of $E(K)$, and we shall see that, at least up to equivalence, we can do this in a canonical way.

Proposition 2 gives a "basis transformation formula" for elliptic nets. This formula holds in the finite field case by Theorem 3. We will see that it provides an equivalence of nets. This allows us to define a net on a free abelian cover of $E(K)$ whose equivalence class is unique.

For a finite field K, and elliptic curve E_K defined over K, there always exists a number field $L \subset \mathbb{C}$, prime $\mathfrak{p}$, and elliptic curve E_L defined over L such that $K = k_\mathfrak{p}$ and $E_K = \delta(E_L)$. Let $q : \mathbb{C} \to E_L$ be the complex uniformisation. Then we define

$$\hat{E}_K = q^{-1} \circ \delta^{-1}(E_K(K))$$

This is a free abelian group of finite rank with a quotient map

$$\pi : \hat{E}_K \to E_K(K) \ .$$

Let $\hat{\Gamma} \cong \mathbb{Z}^n$ be a subgroup of $\hat{E}_K$. Let $\Gamma = \pi(\hat{\Gamma})$. For any surjective homomorphism $\phi : \mathbb{Z}^n \to \Gamma$ there exists a lift $\hat{\phi} : \mathbb{Z}^n \to \hat{\Gamma}$ which is an isomorphism.

We define

$$V_\phi = W_\phi \circ \hat{\phi}^{-1} \ .$$

Theorem 5. *$V_\phi \in \mathcal{EN}(\hat{\Gamma}, K)$ and the equivalence class of V_ϕ is independent of the choice of the surjective map $\phi : \mathbb{Z}^n \to \Gamma$.*

Proof. The linearity of ϕ^{-1} shows that V_ϕ is an elliptic net.

Suppose $T : \mathbb{Z}^n \to \mathbb{Z}^n$ is a homomorphism. Then a restatement of Proposition 2 translated to finite fields via Theorem 3 is that $W_{\phi \circ T} \sim W_\phi \circ T$ (note that every finite field has a primitive element).

Now choose another surjective $\phi' : \mathbb{Z}^n \to \Gamma$. Then there exists an isomorphism $T : \mathbb{Z}^n \to \mathbb{Z}^n$ such that $\hat{\phi} \circ T = \hat{\phi}'$ and $\phi \circ T = \phi'$. Then

$$V_{\phi'} = W_{\phi'} \circ \hat{\phi}'^{-1} = W_{\phi \circ T} \circ T^{-1} \circ \hat{\phi}^{-1} \sim W_\phi \circ \hat{\phi}^{-1} = V_\phi \ .$$

The equivalence holds since $T^{-1} \circ \hat{\phi}^{-1}$ is linear. So we have defined a unique class $[V_\phi] \in \mathcal{EN}_0(\hat{\Gamma}, K)$. □

Definition 6. *Let $\mathcal{W}_{\hat{E}_K}$ denote the class $[V_\phi] \in \mathcal{EN}_0(\hat{E}_K, K)$ defined in Theorem 5.*

This equivalence class is in some sense the "abstract" elliptic net. Just as one writes an abstract linear transformation as a matrix with respect to a basis in order to do calculations, we must choose a basis in order to do calculations with

[1] An examination of the statement of Theorem 6 reveals that the difference in these values is in some sense what the Tate pairing measures.

nets. This choice of basis is for us the choice of homomorphism $\phi : \mathbb{Z}^n \to E(K)$. Theorem 6 gives a formula for the Tate pairing independent of the equivalence class chosen in $\mathcal{W}_{\hat{E}_K}$. Later, we will exploit this freedom to choose an appropriate ϕ for efficient calculations.

We note one useful proposition.

Proposition 4. *Let $W \in \mathcal{W}_{\hat{E}_K}$. Then $W(p) = 0$ implies $\pi(p) = \mathcal{O}$.*

Proof. If $W(p) = 0$ then by definition $\Omega_{\mathbf{v}}(\mathbf{P}) = 0$ for some $\mathbf{v}$ and $\mathbf{P}$ such that $\mathbf{v} \cdot \mathbf{P} = \pi(p)$. But the zeroes $\mathbf{P}$ of $\Psi_{\mathbf{v}}$ are exactly those $\mathbf{P}$ such that $\mathbf{v} \cdot \mathbf{P} = 0$. □

2.6 The Tate Pairing

Choose $m \in \mathbb{Z}^+$. Let E be an elliptic curve defined over a field K containing the m-th roots of unity. Suppose $P \in E(K)[m]$ and $Q \in E(K)/mE(K)$. Since P is an m-torsion point, $m(P) - m(\mathcal{O})$ is a principal divisor, say $\mathrm{div}(f_P)$. Choose another divisor D_Q defined over K such that $D_Q \sim (Q) - (\mathcal{O})$ and with support disjoint from $\mathrm{div}(f_P)$. Then, we may define the Tate pairing

$$\tau_m : E(K)[m] \times E(K)/mE(K) \to K^*/(K^*)^m$$

by

$$\tau_m(P, Q) = f_P(D_Q) \ .$$

This pairing is well-defined, bilinear and Galois invariant. For cryptographic applications, the Tate pairing is usually considered over finite fields, where it is non-degenerate. For details, see [21, 22].

3 Tate Pairing Using Elliptic Nets

Theorem 6. *Fix a positive $m \in \mathbb{Z}$. Let E be an elliptic curve defined over a finite field K containing the m-th roots of unity. Let P, $Q \in E(K)$, with $[m]P = \mathcal{O}$. Choose $S \in E(K)$ such that $S \notin \{\mathcal{O}, -Q\}$. Choose p, q, $s \in \hat{E}_K$ such that $\pi(p) = P$, $\pi(q) = Q$ and $\pi(s) = S$. Let $W \in \mathcal{W}_{\hat{E}_K}$. Then the quantity*

$$T_m(P, Q) = \frac{W(s + mp + q)W(s)}{W(s + mp)W(s + q)} \tag{9}$$

is a well-defined function $T_m : E(K)[m] \times E(K)/mE(K) \to K^*/(K^*)^m$. Further, $T_m(P, Q) = \tau_m(P, Q)$, the Tate pairing.

Proof. By Proposition 4 and the assumptions on the choice of S, any W in the equivalence class of $\mathcal{W}$ is non-vanishing at the four arguments in (9). To verify that T_m is independent of choice of representative of $\mathcal{W}$, suppose that W_1 and

W_2 are in the equivalence class of $\mathcal{W}$. Then $W_2(\mathbf{v}) = \alpha\beta^{f(\mathbf{v})}W_1(\mathbf{v})$ for some $\alpha, \beta \in K^*$ and quadratic form f. Then

$$\begin{aligned}&\frac{W_1(s+mp-q)W_1(s)W_2(s+mp)W_2(s-q)}{W_1(s+mp)W_1(s-q)W_2(s+mp-q)W_2(s)}\\ &= \beta^{f(s+mp)+f(s-q)-f(s+mp-q)-f(s)}\\ &= \beta^{f(mp+q)-f(mp)-f(q)} = \beta^{m[f(p+q)-f(p)-f(q)]} \in (K^*)^m \ .\end{aligned}$$

Let $\Gamma \subset E_K(K)$ be the subgroup generated by S, P, and Q. Let

$$f_P = \frac{\Omega_{1,0,0}(-S,P,Q)}{\Omega_{1,m,0}(-S,P,Q)} \ .$$

Therefore, we may compute the divisor of f_P as a function of S (by equation (3)):

$$(f_P) = -([m]P) + (1-m)(\mathcal{O}) + m(P) = m(P) - m(\mathcal{O}) \ .$$

Let D_Q be the divisor $(-S) - (-S-Q)$.

Then, using Proposition 2 and Theorem 3, in $K^*/(K^*)^m$,

$$\begin{aligned}f_P(D_Q) &= \frac{\Omega_{1,0,0}(S,P,Q)\Omega_{1,m,0}(S+Q,P,Q)}{\Omega_{1,m,0}(S,P,Q)\Omega_{1,0,0}(S+Q,P,Q)}\\ &= \frac{\Omega_{1,0,0}(S,P,Q)\Omega_{1,m,1}(S,P,Q)}{\Omega_{1,m,0}(S,P,Q)\Omega_{1,0,1}(S,P,Q)} \ .\end{aligned}$$

By a choice of $\phi : \mathbb{Z}^3 \to \Gamma$ such that $\phi(1,0,0) = S$, $\phi(0,1,0) = P$, and $\phi(0,0,1) = Q$, we have $W_\phi(\mathbf{v}) = \Omega_{\mathbf{v}}(S,P,Q) \in \mathcal{EN}(\mathbb{Z}^3, K)$. Therefore

$$\tau_m(P,Q) = f_P(D_Q) = \frac{V_\phi(s+mp+q)V_\phi(s)}{V_\phi(s+mp)V_\phi(s+q)} = T_m(P,Q) \ .$$

□

Corollary 1. *Let E be an elliptic curve defined over a finite field K, m a positive integer, $P \in E(K)[m]$ and $Q \in E(K)$. If W_P is the elliptic net associated to E, P, then we have*

$$\tau_m(P,P) = \frac{W_P(m+2)W_P(1)}{W_P(m+1)W_P(2)} \ . \tag{10}$$

Further, if $W_{P,Q}$ is the elliptic net associated to E, P, Q, then we have

$$\tau_m(P,Q) = \frac{W_{P,Q}(m+1,1)W_{P,Q}(1,0)}{W_{P,Q}(m+1,0)W_{P,Q}(1,1)} \ . \tag{11}$$

Proof. For the first formula, taking $q = p$ and $s = 2p$, we obtain

$$T_m(P,P) = \frac{W((m+2)p)W(p)}{W((m+1)p)W(2p)} \ .$$

For the second, take $s = p$, obtaining

$$T_m(P,Q) = \frac{W((m+1)p+q)W(p)}{W((m+1)p)W(p+q)} \ . \qquad \square$$

4 Tate Pairing Computation

4.1 Computing the Values of an Elliptic Net

Rachel Shipsey gives a double-and-add algorithm for computing terms of an elliptic divisibility sequence [14]. In the case of interest to us now, given the initial values of an elliptic divisibility sequence, the algorithm computes the n-th term of a sequence in $\log(n)$ time. Shipsey applied her more general algorithm (which allows beginning elsewhere in the sequence) to give a solution to the elliptic curve discrete logarithm problem in certain cases.

The algorithm described here is an adaptation and generalisation of Shipsey's algorithm to calculate terms $W(m,0)$ and $W(m,1)$ of an elliptic net. We define a *block centred on* k (shown in Fig. 2) to consist of a first vector of eight consecutive terms of the sequence $W(i,0)$ centred on terms $W(k,0)$ and $W(k+1,0)$ and a second vector of three consecutive terms $W(i,1)$ centred on the term $W(k,1)$. We define two functions:

1. **Double**(V)**:** Given a block V centred on k, returns the block centred on $2k$.
2. **DoubleAdd**(V)**:** Given a block V centred on k, returns the block centred on $2k+1$.

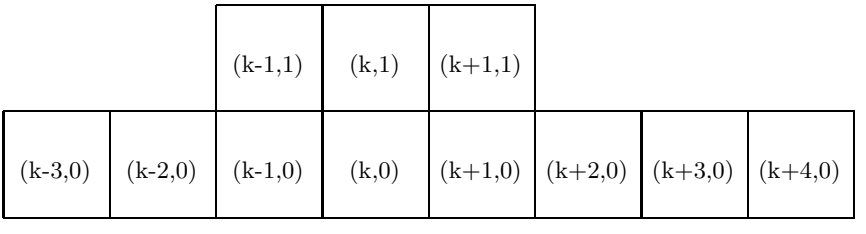


Fig. 2. A block centred on k

We assume the elliptic net satisfies $W(1,0) = W(0,1) = 1$. The first vectors of Double(V) and DoubleAdd(V) are calculated according to the following special cases of (5) (or (1)):

$$W(2i-1,0) = W(i+1,0)W(i-1,0)^3 - W(i-2,0)W(i,0)^3 \ , \qquad (12)$$

$$W(2i,0) = (W(i,0)W(i+2,0)W(i-1,0)^2 - W(i,0)W(i-2,0)W(i+1,0)^2)/W(2,0) \ . \qquad (13)$$

The formulæ needed for the computations of the second vectors are instances of (5).[2]

$$W(2k-1,1) = (W(k+1,1)W(k-1,1)W(k-1,0)^2 - W(k,0)W(k-2,0)W(k,1)^2)/W(1,1) \ , \tag{14}$$

$$W(2k,1) = W(k-1,1)W(k+1,1)W(k,0)^2 - W(k-1,0)W(k+1,0)W(k,1)^2 \ , \tag{15}$$

$$W(2k+1,1) = (W(k-1,1)W(k+1,1)W(k+1,0)^2 - W(k,0)W(k+2,0)W(k,1)^2)/W(-1,1) \ , \tag{16}$$

$$W(2k+2,1) = (W(k+1,0)W(k+3,0)W(k,1)^2 - W(k-1,1)W(k+1,1)W(k+2,0)^2)/W(2,-1) \ . \tag{17}$$

Equations (12) and (13), applied for $i = k-1, \ldots, k+3$, allow calculation of the first vectors of Double(V) and DoubleAdd(V) in terms of $W(2,0)$ and the terms of V. Equations (14)–(17) allow calculation of the second vectors in terms of $W(1,1)$, $W(-1,1)$, $W(2,-1)$ and the terms of V.

The algorithm to calculate $W(m,1)$ and $W(m,0)$ for any positive integer m is shown in Algorithm 1. The formula for the last term of the first vector of V in line 1 is from (11). Note that elliptic nets satisfy $W(-n,-m) = -W(n,m)$ by Proposition 3. In Sect. 5.1 we will consider possible optimisations.

Algorithm 1. Elliptic Net Algorithm

Input: Initial terms $a = W(2,0), b = W(3,0), c = W(4,0), d = W(2,1), e = W(-1,1), f = W(2,-1), g = W(1,1)$ of an elliptic net satisfying $W(1,0) = W(0,1) = 1$ and integer $m = (d_k d_{k-1} \ldots d_1)_2$ with $d_k = 1$
Output: Elliptic net elements $W(m,0)$ and $W(m,1)$

1: $V \leftarrow [[-a, -1, 0, 1, a, b, c, a^3c - b^3]; [1, g, d]]$
2: **for** $i = k-1$ down to 1 **do**
3: **if** $d_i = 0$ **then**
4: $V \leftarrow$ Double(V)
5: **else**
6: $V \leftarrow$ DoubleAdd(V)
7: **end if**
8: **end for**
9: **return** $V[0,3]$ and $V[1,1]$ // terms $W(m,0)$ and $W(m,1)$ respectively

4.2 Computation of the Tate Pairing

We can now compute the Tate pairing via Corollary 1. Consider an elliptic curve E over a finite field $\mathbb{F}_q$ of characteristic not 2 or 3, in Weierstrass form

$$y^2 = x^3 + Ax + B$$

[2] The values p, q, r, s substituted into (5) to obtain equations (14) - (17) are $[p,q,r,s] = [(k,0),(k-1,0),(1,0),(0,1)]$, $[(k+1,0),(k,0),(1,0),(-1,1)]$, $[(k+1,0),(k,0),(-1,0),(0,1)]$, and $[(k+2,0),(k,1),(1,0),(0,0)]$ respectively.

and points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ on $E(\mathbb{F}_q)$ with $Q \neq \pm P$. We must calculate the values a, b, c, d, e, f, g required as input for the Elliptic Net Algorithm. These are terms of the elliptic net associated to E, P, Q. The necessary formulæ are given by the functions $\Psi_{m,n}$. In the case that $m = 0$, these are called *division polynomials* (see [16, p.105] and [17, p.477]). We have

$$W(1,0) = 1 \ , \tag{18}$$

$$W(2,0) = 2y_1 \ , \tag{19}$$

$$W(3,0) = 3x_1^4 + 6Ax_1^2 + 12Bx_1 - A^2 \ , \tag{20}$$

$$W(4,0) = 4y_1(x_1^6 + 5Ax_1^4 + 20Bx_1^3 - 5A^2x_1^2 - 4ABx_1 - 8B^2 - A^3) \ . \tag{21}$$

For the formulæ in case of characteristic 2 or 3, or the more general Weierstrass form, see [23, p.80]. Also using classical formulæ (see for example [24]), we have

$$W(0,1) = W(1,1) = 1 \ , \tag{22}$$

$$W(2,1) = 2x_1 + x_2 - \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 \ , \tag{23}$$

$$W(-1,1) = x_1 - x_2 \ , \tag{24}$$

$$W(2,-1) = (y_1 + y_2)^2 - (2x_1 + x_2)(x_1 - x_2)^2 \ . \tag{25}$$

Suppose that P has order m. Then we use the Elliptic Net Algorithm, with input $m + 1$ and a, b, c, d, e, f, g given by (19)–(25).[3] The output is used to evaluate formula (11) of Corollary 1, giving the Tate pairing.

5 Analysis

5.1 Some Implementation Considerations

For an integer m and finite field $\mathbb{F}_q$, we define the *embedding degree* k to be the least integer such that $m|(q^k - 1)$, thus ensuring the m-th roots of unity are contained in $\mathbb{F}_{q^k}^*$. In cryptographic applications of the Tate pairing, it is usual to use a curve defined over $\mathbb{F}_q$ of embedding degree $k > 1$, and points $P \in E(\mathbb{F}_q)$, $Q \in E(\mathbb{F}_{q^k})$: throughout what follows we make this assumption.

First, note that no inversions are actually needed in equations (12)–(17), since the inverses of $W(2,0)$, $W(2,1)$, $W(-1,1)$ and $W(2,-1)$ may be precomputed before the double-and-add loop is begun. Therefore these inversions are replaced by multiplications.

Now we consider optimisations in the functions Double and DoubleAdd. The largest savings can be gained by first computing a number of products which appear frequently in the formulæ:

$$W(i,0)^2 \text{ and } W(i-1,0)W(i+1,0) \quad \text{for } i = k-2, \ldots, k+3 \ ,$$
$$W(k,1)^2 \text{ and } W(k-1,1)W(k+1,1) \ .$$

[3] In this case, $g = 1$. However, in Sect. 5.1 we will replace this elliptic net with an equivalent one for which $W(1,1) \neq 1$. For this reason, it is convenient to state Algorithm 1 in sufficient generality and include a variable g.

With these 14 computations, each term of the 11 to be calculated requires only two multiplications and an addition (plus multiplications by $W(2,0)^{-1}$, $W(2,-1)^{-1}$, $W(1,1)^{-1}$ and $W(-1,1)^{-1}$). The resulting Double and DoubleAdd algorithms are shown in Algorithm 2.

Algorithm 2. Double and DoubleAdd

Input: Block V centred at k of an elliptic net satisfying $W(1,0) = W(0,1) = 1$, values $A = W(2,0)^{-1}, E = W(-1,1)^{-1}, F = W(2,-1)^{-1}, G = W(1,1)^{-1}$ and boolean add

Output: Block centred at $2k$ if $add == 0$ and centred at $2k+1$ if $add == 1$

$S \leftarrow [0,0,0,0,0,0]$
$P \leftarrow [0,0,0,0,0,0]$
$S_0 \leftarrow V[1,1]^2$
$P_0 \leftarrow V[1,0]V[1,2]$
for $i = 0$ to 5 **do**
  $S[i] \leftarrow V[0,i+1]^2$
  $P[i] \leftarrow V[0,i]V[0,i+2]$
end for
if $add == 0$ **then**
  for $i = 1$ to 4 **do**
    $V[0,2i-2] \leftarrow S[i]P[i+1] - S[i+1]P[i]$
    $V[0,2i-1] \leftarrow (S[i]P[i+2] - S[i+2]P[i])A$
  end for
  $V[1,0] \leftarrow (S_0P[3] - S[3]P_0)G$
  $V[1,1] \leftarrow S[3]P_0 - S_0P[3]$
  $V[1,2] \leftarrow (S[4]P_0 - S_0P[4])E$
else
  for $i = 1$ to 4 **do**
    $V[0,2i-2] \leftarrow (S[i]P[i+2] - S[i+2]P[i])A$
    $V[0,2i-1] \leftarrow S[i+1]P[i+2] - S[i+2]P[i+1]$
  end for
  $V[1,0] \leftarrow S[3]P_0 - S_0P[3]$
  $V[1,1] \leftarrow (S[4]P_0 - S_0P[4])E$
  $V[1,2] \leftarrow (S_0P[5] - S[5]P_0)F$
end if
return V

Finally, we may try to avoid some of these extra multiplications by $W(2,0)^{-1}$, $W(1,1)^{-1}$, $W(2,1)^{-1}$ and $W(2,-1)^{-1}$ entirely. Recall that by Theorem 6, applying an equivalence to the net will not alter the Tate pairing result. Let $\eta = W(-1,1)$. Apply the equivalence given by $\alpha = 1$, $\beta = \eta$ and $f(n,m) = mn$. Clearly, this preserves the conditions[4] that $W(1,0) = W(0,1) = 1$ (and leaves terms $W(n,0)$ unchanged, so they are still in $\mathbb{F}_q$), but changes $W(-1,1)$ to 1, which saves one multiplication in $\mathbb{F}_{q^k}$ per iteration. If $W(2,0)$ has a cube root

[4] These were needed to derive formulæ (12)–(17).

ν in $\mathbb{F}_q$, then the equivalence $\alpha = \nu^{-1}, \beta = \nu$ and $f(n,m) = m^2 + n^2 + mn$ will change $W(2,0)$ to 1, while preserving $W(1,0) = W(0,1) = W(-1,1) = 1$, saving four $\mathbb{F}_q$ multiplications per iteration. Note that these equivalences may result in $W(1,1) \neq 1$.

Finally, we consider the applicability of some of the usual optimisations of Miller's algorithm. In Miller's algorithm, a final exponentiation is applied, in order to compute a unique value for the Tate pairing; the same exponentiation must be applied here. In the case of Miller's, this exponentiation eliminates multiplicative factors living in the base field $\mathbb{F}_q$. In our case, the $\mathbb{F}_q$ computations do not give rise to strictly multiplicative factors (the algorithm requires much addition and subtraction), and so we cannot use this final exponentiation as a justification for the saving of $\mathbb{F}_q$ computations. Windowing methods (as in [25] and [26]) may lead to improvement. A triple-and-add adaptation (as in [11] and [27]) does not seem promising, by the nature of the recurrence relation. However, efficiency improvements are likely to be found by studying the characteristic 2 and 3 cases.

5.2 Complexity

Since the algorithm involves a fixed number of precomputations, and a double-and-add loop with a fixed number of computations per step, the algorithm is linear time in the size of m, as is Miller's algorithm. Miller's algorithm also consists of a double-and-add loop, and we call the two internal steps Double and DoubleAdd, as for the Elliptic Net Algorithm. In Miller's algorithm the cost of DoubleAdd is almost twice that of Double. By contrast, in the Elliptic Net Algorithm these steps take the same time, so the complexity is independent of Hamming weight. This makes the choice of appropriate curves for cryptographical implementations somewhat easier [6], and may help discourage side channel attacks.

Denote squaring and multiplication in $\mathbb{F}_q$ by S and M. Denote squaring and multiplication in $\mathbb{F}_{q^k}$ by S_k and M_k. Assume that multiplying an element of $\mathbb{F}_q$ by one of $\mathbb{F}_{q^k}$ takes k multiplications in $\mathbb{F}_q$. Recall that E is defined over $\mathbb{F}_q$, $P \in E(\mathbb{F}_q)$, and $Q \in E(\mathbb{F}_{q^k})$. Then any term $W(n,0)$, being a term in the elliptic divisibility sequence associated to E, P, has a value in $\mathbb{F}_q$. Under the optimisations discussed in Sect. 5.1, each Double or DoubleAdd step requires $6S + (6k+26)M + S_k + 2M_k$. Furthermore, under the condition that $2y_P \in \mathbb{F}_q$ is a cube, then precomputing its cube root will save four multiplications in $\mathbb{F}_q$ per step.

The Elliptic Net Algorithm requires no inversions. Miller's algorithm in affine coordinates requires one or two $\mathbb{F}_q$ inversion per step. In situations where inversions are costly (depending on implementation, they may cost anywhere from approximately 4 to 80 multiplications [28]), one may implement Miller's algorithm in homogeneous coordinates.

For the purpose of comparison, we consider an optimised implementation of Miller's algorithm in Jacobian coordinates analysed by Neal Koblitz and Alfred

Menezes [29]. In their implementation, they assume $x(Q) \in E(\mathbb{F}_{q^{k/2}})$ (this is possible by using a twist of the curve, see for example [30]). Applying this additional assumption to the elliptic net algorithm, $W(1,1)$ will be an element of $\mathbb{F}_{q^{k/2}}$, reducing one of the multiplications in Double to one half the time. The comparison is summarised in Tables 1 and 2. In the latter, a squaring is assumed to be comparable to a multiplication (although it is more usually assumed to be 0.8 times as fast), and a multiplication in $\mathbb{F}_{q^k}$ is assumed to take $k^{1.5}$ multiplications in $\mathbb{F}_q$ (see [29]). The number of steps constitutes a range because the Double and DoubleAdd steps may differ in cost.

Table 1. Comparison of Operations for Double and DoubleAdd steps

Algorithm	Double	DoubleAdd
Optimised Miller's [29]	$4S + (k+7)M + S_k + M_k$	$7S + (2k+19)M + S_k + 2M_k$
Elliptic Net Algorithm	$6S + (6k+26)M + S_k + \frac{3}{2}M_k$	$6S + (6k+26)M + S_k + 2M_k$

Table 2. $\mathbb{F}_q$ Multiplications per Step

Embedding degree	2	4	6	8	10	12
Optimised Miller's	18-38	31-58	46-82	64-109	84-140	106-174
Elliptic Net	51-52	76-80	104-112	136-147	171-186	207-228

5.3 A Remark on Implementations

The elliptic net algorithm has been implemented by the author for PARI/GP (see [31]) and is available at [32]. It has also been implemented in C++ by Michael Scott and Augusto Jun Devegili for a pairing-friendly curve of degree 2. The implementation by Ben Lynn in the Pairing Based Cryptography Library [33] is applicable to curves of various sizes and embedding degrees and includes a program to compare the Elliptic Net algorithm with Miller's. Preliminary data agree with the complexity analysis above.

6 Conclusions

The Elliptic Net Algorithm has no significant restrictions on the points, curves or finite fields to which it applies, and requires no inversions. The efficiency of the algorithm is comparable to Miller's algorithm. One expects that the Elliptic Net Algorithm will yield to further optimisation, possibly providing an efficient alternative to Miller's algorithm in many cases. The theory of elliptic nets here introduced may also yield other applications in the field of elliptic curve cryptography.

Acknowledgments. The author would like to thank Rafe Jones, Anna Lysyanskaya, Michelle Manes, Michael Scott, Joseph Silverman and Jonathan Wise for helpful discussions and editorial comments; and the anonymous referees for suggestions. This work was supported by NSERC Award PGS D2 331379-2006.

References

[1] Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. Inform. Theory 39(5), 1639–1646 (1993)
[2] Frey, G., Rück, H.G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comp. 62(206), 865–874 (1994)
[3] Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: Symposium on Cryptography and Information Security, Okinawa, Japan (2000)
[4] Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: Bosma, W. (ed.) Algorithmic Number Theory. LNCS, vol. 1838, pp. 385–393. Springer, Heidelberg (2000)
[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] Duquesne, S., Lange, T.: Pairing-based cryptography. In: Handbook of elliptic and hyperelliptic curve cryptography. Discrete Math. Appl, Boca Raton, pp. 573–590. Chapman & Hall/CRC, Boca Raton, FL (2006)
[7] Paterson, K.G.: Cryptography from pairings. In: Advances in elliptic curve cryptography. London Math. Soc. Lecture Note Ser., vol. 317, pp. 215–251. Cambridge Univ. Press, Cambridge (2005)
[8] Barreto, P.S.L.M.: The pairing-based crypto lounge. http://planeta.terra.com.br/informatica/paulbarreto/pblounge.html
[9] Miller, V.: Short programs for functions on curves (1986)
[10] Duquesne, S., Frey, G.: Implementation of pairings. In: Handbook of elliptic and hyperelliptic curve cryptography. Discrete Math. Appl., Boca Raton, pp. 389–404. Chapman & Hall/CRC, Boca Raton, FL (2006)
[11] Galbraith, S.D., Harrison, K., Soldera, D.: Implementing the Tate pairing. In: Fieker, C., Kohel, D.R. (eds.) Algorithmic Number Theory. LNCS, vol. 2369, pp. 324–337. Springer, Heidelberg (2002)
[12] Ward, M.: Memoir on elliptic divisibility sequences. Amer. J. Math. 70, 31–74 (1948)
[13] Everest, G., Poorten, A.v.d., Shparlinski, I., Ward, T.: Elliptic Divisibility Sequences. American Mathematical Society, Providence, pp. 163–175 (2003)
[14] Shipsey, R.: Elliptic Divibility Sequences. PhD thesis, Goldsmiths, University of London (2001)
[15] Stange, K.E.: Elliptic Nets. PhD thesis, Brown University (in preparation)
[16] Silverman, J.H.: The arithmetic of elliptic curves (Corrected reprint of the 1986 original). Graduate Texts in Mathematics, vol. 106. Springer, New York (1992)
[17] Silverman, J.H.: Advanced topics in the arithmetic of elliptic curves. Graduate Texts in Mathematics, vol. 151. Springer, New York (1994)
[18] Swart, C.: Elliptic curves and related sequences. PhD thesis, Royal Holloway and Bedford New College, University of London (2003)
[19] van der Poorten, A.J.: Elliptic curves and continued fractions. J. Integer Seq. Article 05.2.5, (electronic) 8(2), 19 (2005)

[20] Propp, J.: Robbins forum http://www.math.wisc.edu/~propp/about-robbins
[21] Duquesne, S., Frey, G.: Background on pairings. In: Handbook of elliptic and hyperelliptic curve cryptography. Discrete Math. Appl, Boca Raton, pp. 115–124. Chapman & Hall/CRC, Boca Raton, FL (2006)
[22] Galbraith, S.: Pairings. In: Advances in elliptic curve cryptography. London Math. Soc. Lecture Note Ser., vol. 317, pp. 183–213. Cambridge Univ. Press, Cambridge (2005)
[23] Frey, G., Lange, T.: Background on curves and Jacobians. In: Handbook of elliptic and hyperelliptic curve cryptography. Discrete Math. Appl., Boca Raton, pp. 45–85. Chapman & Hall/CRC, Boca Raton, FL (2006)
[24] Chandrasekharan, K.: Elliptic functions. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 281. Springer, Heidelberg (1985)
[25] Blake, I.F., Seroussi, G., Smart, N.P.: Elliptic curves in cryptography (Reprint of the 1999 original). London Mathematical Society Lecture Note Series, vol. 265. Cambridge University Press, Cambridge (2000)
[26] Hankerson, D., Hernandez, J.L., Menezes, A.: Software implementation of elliptic curve cryptography over binary fields. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 1–24. Springer, Heidelberg (2000)
[27] 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)
[28] Ciet, M., Joye, M., Lauter, K., Montgomery, P.L.: Trading inversions for multiplications in elliptic curve cryptography. Des. Codes Cryptogr. 39(2), 189–206 (2006)
[29] Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)
[30] 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)
[31] The PARI Group: Pari/gp development headquarters http://pari.math.u-bordeaux.fr/
[32] Stange, K.E.: Pari/gp scripts for tate pairing via elliptic nets. http://www.math.brown.edu/~stange/tatepairing/
[33] Lynn, B.: Pairing-based cryptography library http://crypto.stanford.edu/pbc/

Eta Pairing Computation on General Divisors over Hyperelliptic Curves $y^2 = x^7 - x \pm 1$

Eunjeong Lee[1,⋆], Hyang-Sook Lee[2], and Yoonjin Lee[3]

[1] Korea Institute for Advanced Study, Seoul 130-722, S. Korea
ejlee@kias.re.kr
[2] Department of Mathematics, Ewha Womans University, Seoul 120-750, S. Korea
hsl@ewha.ac.kr
[3] Department of Mathematics, Simon Fraser University, Burnaby, British Columbia, V5A 1S6, Canada
yoonjinl@sfu.ca

Abstract. Recent developments on the Tate or Eta pairing computation over hyperelliptic curves by Duursma-Lee and Barreto et al. have focused on *degenerate* divisors. We present two efficient methods that work for *general* divisors to compute the Eta paring over divisor class groups of the hyperelliptic curves $H/\mathbb{F}_{7^n} : y^2 = x^7 - x \pm 1$ of genus 3. The first method generalizes the method of Barreto et al. so that it holds for general divisors, and we call it the *pointwise* method. For the second method, we take a novel approach using *resultant*. We focus on the case that two divisors of the pairing have supporting points in $H(\mathbb{F}_{7^{3n}})$, not in $H(\mathbb{F}_{7^n})$. Our analysis shows that the resultant method is faster than the pointwise method, and our implementation result supports the theoretical analysis. In addition to the fact that the two methods work for general divisors, they also provide very explicit algorithms.

Keywords: Eta pairing, Tate pairing, hyperelliptic curves, divisors, resultant, pairing-based cryptosystem.

Introduction

Recent developments of pairing-based protocols call for efficient computation of pairings ([5], [6], [15], [24], [27], [28]). Barreto et al. [2] and Galbraith et al. [13] provided the fast computation of Tate pairing over supersingular elliptic curves $y^2 = x^3 - x \pm 1$ in characteristic three. In 2003, Duursma and Lee [11] provided a closed formula for the efficient computation of the Tate pairing on $y^2 = x^p - x \pm 1,\ p = 3 \pmod 4$ in characteristic p. After then, Barreto et al. [3] proposed the efficient computations of Tate pairing on supersingular abelian varieties using the *Eta pairing* approach. More recently, Hess et al. [16] proposed

⋆ The authors[1,2] were supported by KOSEF, grant number R01-2005-000-10713-0, and the author[3] was supported by NSERC and SFU PRG. The authors also express their gratitude to KIAS institute.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 349–366, 2007.

the *Ate pairing* on elliptic curves, instead of the Tate pairing, for a bilinear map in pairing-based protocols.

Over hyperelliptic curves, divisor operations are more complicated than point operations over elliptic curves. Thus, the Tate pairing or Eta pairing computation over a hyperelliptic curve is not considered as efficient as that over an elliptic curve. However, the Eta pairing was faster over hyperelliptic curves with genus 2 than elliptic curves according to the implementation result in [2]. Moreover, in some special cases, it was shown that hyperelliptic curve cryptosystem (HCC) can be made more efficient than elliptic curve cryptosystems (ECC) by giving the explicit formula for divisor operations ([22], [25]). For the higher genus, preserving the same security level, we can decrease the size of the defining field. In fact, some examples given in [22] show that for the efficiency of cryptosystems, the size of the defining field is more important than the complexity of group operation formula. Therefore, it is worthwhile to work over some special types of hyperelliptic curves for efficient Tate or Eta pairing computations.

Recent developments ([3], [11]) on the Tate pairing computation on hyperelliptic curves over a finite field $\mathbb{F}_q$ have focused on the case of *degenerate* divisors. However, in the pairing-based cryptography, the efficient Tate pairing implementation over *general* divisors is significantly more important. For instance, in the Boneh-Franklin identity-based encryption scheme, the private keys are general divisors, and therefore the decryption process requires computing a pairing of general divisors. For the case of genus 2, the result in [7] presents both divisor-wise and pointwise approach, and it turns out that the divisor-wise approach is more efficient than the pointwise approach. For the case of genus ≥ 3, no Tate pairing computation method has been developed for general divisors.

In this paper, we present two algorithms for computing the Eta pairing on *general* divisors over hyperelliptic curves $y^2 = x^7 - x \pm 1$. As well as the two methods work for general divisors, they also provide very explicit algorithms. The first algorithm is a generalization of the algorithm for the Eta pairing computation on degenerate divisors by Barreto et al. [3], called the *pointwise* method. For the second algorithm, we take a novel approach using *resultant*. It is a hard task to find an explicit algorithm for the Eta pairing computation only by using symmetric functions from the product of the Eta paring value on each pair of supporting points. However, an advantage of using the resultant is that we can make the computation steps much simpler and more explicit so that we can obtain an explicit algorithm.

For the complexity analysis, we focus on the case that both divisors of the Eta pairing consist of supporting points in $H(\mathbb{F}_{7^{3n}})$, not in $H(\mathbb{F}_{7^n})$. Our analysis shows that in this case the resultant method is faster than the pointwise method. In more detail, the resultant method is 48.5% faster than the pointwise computation in the best case and 15.3% faster in the worst case, and our implementation result supports the theoretical analysis. This is the first implementation over hyperelliptic curves with genus 3.

We organize this paper as follows. In Section 1, we give a brief summary of the Tate pairing and the Eta pairing, Section 2 discusses the pointwise method,

and Section 3 presents the resultant method. In Section 4, we compare the complexities of two methods, and Section 5 provides experimental results based on our implementation using NTL [26] software package. We used SINGULAR [14] software package for symbolic computations.

1 Tate Pairing and Eta Pairing

Let $\mathbb{F}_q$ be a finite field with q elements, and $H/\mathbb{F}_q$ be a hyperelliptic curve over $\mathbb{F}_q$. We denote by J_H the group of degree zero divisor classes of H. Note that each divisor class can be uniquely represented by the *reduced divisor* using the *Mumford representation* [23]. Reduced divisors of the curve H can be found as discussed in [19] and [23], and most of reduced divisors in J_H with genus 3 are written as $D = [U_D, V_D] = [x^3 + u_{D,2}x^2 + u_{D,1}x + u_{D,0},\ v_{D,2}x^2 + v_{D,1}x + v_{D,0}]$.

Tate Pairing on H_d

We recall the definition of the Tate pairing [12]. Let ℓ be a positive divisor of the order of $J_H(\mathbb{F}_q)$ with $\gcd(\ell,\ q) = 1$, and k be the smallest integer such that $\ell \mid (q^k - 1)$; such k is called *the embedding degree*. Let $J_H[\ell] = \{D \in J_H \mid \ell D = \mathcal{O}\}$. The *Tate pairing* is a map

$$\langle\ \cdot\ \rangle_\ell : J_H[\ell] \times J_H(\mathbb{F}_{q^k})/\ell J_H(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^\ell$$
$$\langle D, E\rangle_\ell = f_D(E'),$$

where $\mathrm{div}(f_D) = \ell D$ and $E' \sim E$ with $support(E') \cap support(\mathrm{div}(f_D)) = \emptyset$. We define the Tate paring value by $t(D,E) = \langle D, E\rangle_\ell^{\frac{q^k-1}{\ell}}$ so that the pairing value is defined uniquely. Here ℓ can be replaced by any integer N such that $\ell \mid N \mid q^k - 1$ [13]. Thus $t(D,E) = \langle D, E\rangle_N^{\frac{q^k-1}{N}}$.

We consider a hyperelliptic curve H_d over $\mathbb{F}_q$ defined by $y^2 = x^p - x + d$, $d = \pm 1$, for $p \equiv 3 \pmod 4$, where $q = p^n$ with $\gcd(2p, n) = 1$, and we let $F/\mathbb{F}_q$ and $K/\mathbb{F}_q$ be the extensions of degree $[F : \mathbb{F}_q] = p$ and degree $[K : \mathbb{F}_q] = 2p$, respectively. Over the extension field K, the curve is the quotient of a hermitian curve, hence it is Hasse-Weil maximal. And the class group over K is annihilated by $p^{pn} + 1$; this can be also seen from the following Lemma 1. It shows that for $P \in H_d(K)$, $(p^{pn} + 1)((P) - (\mathcal{O}))$ is principal ([9], [10]). We write $x^{(i)}$ for x^{p^i}.

Lemma 1 ([9], [10]). *Let $P = (\alpha, \beta) \in H_d$. The function*

$$h_{p,P} = \beta^p y - (\alpha^p - x + d)^{\frac{p+1}{2}}$$

has divisor $(h_{p,P}) = p(P) + (P') - (p+1)\mathcal{O}$, where $P' = (\alpha^{(2)} + 2d, \beta^{(2)})$.

Let $v_{p,P}$ be a vertical line passing through P' and $\mathcal{O}$ and $f_{p,P} = h_{p,P}/v_{p,P}$. Then from Lemma 1, it follows that

$$(f_{p,P}) = p(P) - (-P') - (p-1)\mathcal{O} \tag{1}$$

with $-P' = (\alpha^{(2)} + 2d, -\beta^{(2)})$.

From Lemma 1, we observe that $p((P)-(\mathcal{O})) \equiv ([p]P)-(\mathcal{O})$, thus the multiplication by p over H_d has an extremely special form such as $[p] = \phi\pi^2$, where $\phi = (x+2d, -y)$ and π is a Frobenius map of p^{th} power.

Let $m = (n+1)/2$, then from [9], [10] we have

$$|J_{H_\pm}(\mathbb{F}_q)| = (1+p^n)^3 \pm (\frac{p}{n})p^m(1+p^n+p^{2n}). \tag{2}$$

Therefore, the embedding degree k of the curve H_d is equal to $2p$.

Eta Pairing on H_d

Now we discuss the *Eta pairing* introduced in [3] which is very useful for efficient computation of the Tate pairing. We consider a hyperelliptic curve H_d over some finite field $\mathbb{F}_{p^n}$, and let ψ be an endomorphism on the curve H_d given by

$$\psi : H_d(K) \to H_d(K), \quad \psi(x,y) = (\rho - x, \sigma y), \tag{3}$$

where $\rho \in F$ is a root of $\rho^p - \rho + 2d = 0$, and $\sigma, \bar{\sigma} \in K$ are the roots of $\sigma^2 + 1 = 0$.

For efficient Tate pairing computation, we consider *the twisted Tate pairing*

$$\begin{aligned} \hat{t} \ : \ J_{H_d}[\ell] \times J_{H_d}(\mathbb{F}_{p^{2pn}})/\ell J_{H_d}(\mathbb{F}_{p^{2pn}}) \ &\longrightarrow \ \mathbb{F}^*_{p^{2pn}} \\ \hat{t}(D,\ E) \ &= \ f_D(\psi(E))^{p^{pn}-1}, \end{aligned}$$

where $(f_D) = (p^{pn}+1)D$ from [11, Theorem 4].

For two divisors D and E in J_{H_d}, the *Eta pairing* [3] is defined by

$$\eta_T(D,E) = f_{T,D}(\psi(E)),$$

where $D' + (f_{T,D}) = TD$ for a reduced divisor D' with T an integer satisfying certain conditions [3, Theorem 1]. This is a generalization of Duursma-Lee's method [11] and gives a further improvement with shorter loop length by choosing a proper T. Over the hyperelliptic curve H_d, the Eta pairing is optimal when T is q, so we work on the case $T = q$ and in this case we denote η_T by η.

For divisors D, E in J_{H_d}, the Eta pairing can be computed by

$$\eta(D,E) = \prod_{i=0}^{n-1} f_{p,D_i}(\psi(E))^{p^{n-1-i}}, \tag{4}$$

where $D_{i+1} + (f_{p,D_i}) = pD_i$ with a divisor $D_0 = D$ and some rational function f_{p,D_i}.

We have the relation of the Tate pairing and the Eta pairing over the curve H_d [3, Section 8] as follows:

$$\hat{t}(D,E) = \eta(D,E)^{(p^{pn}-1)p^{n(p-1)+1}}. \tag{5}$$

It is therefore enough to compute $\eta(D,E)$ to obtain $\hat{t}(D,\ E)$ for any divisors $D, E \in J_{H_d}(\mathbb{F}_q)$. When all the points in $support(D)$ and $support(E)$ are $\mathbb{F}_q$-rational points, using Eq. (5) makes the Eta pairing computation very efficient

as mentioned in [3]. In Section 2, we will extend the concept of the Eta pairing on general divisors, that is, the supports of D and E are not necessarily $\mathbb{F}_q$-rational points.

Throughout this paper, we focus on the hyperelliptic curve $H_d : y^2 = x^p - x + d,\ d = \pm 1, p \equiv 3 \pmod 4$ of genus $g = 3$, therefore we work on the case $p = 7$; this case is cryptographically useful [11].

2 Pointwise Computation of the Eta Pairing

This section presents a generalization of the pointwise method developed in [3] and [11] for computing the Eta pairing over the general divisors D and E. This method can be used for any general divisors, but we focus on the case that both divisors D and E consist of the supporting points in $H(\mathbb{F}_{7^{3n}})$, not in $H(\mathbb{F}_{7^n})$. We also analyze the complexity of the pointwise computation of the Eta pairing.

For divisors D, E in J_{H_d}, the Eta pairing can be computed by

$$\eta(D,E)^{7^{7n-1}} = \left(\prod_{i=0}^{n-1} f_{7,D_i}(\psi(E))^{7^{n-1-i}}\right)^{7^{7n-1}}, \tag{6}$$

where D and E have the form $D = (P_1)+(P_2)+(P_3)-3(\mathcal{O}),\ \ E = (Q_1)+(Q_2)+(Q_3)-3(\mathcal{O})$ for points P_k and Q_j contained in $H_d(\mathbb{F}_{7^{3n}})$ with $k, j = 1, 2, 3$.

For $i \geq 1$, let $D_i = (P_{i,1}) + (P_{i,2}) + (P_{i,3}) - 3(\mathcal{O})$, then $f_{7,D_i}(\psi(E))$ can be computed by

$$f_{7,D_i}(\psi(E)) = \prod_{k,j=1}^{3} f_{7,P_{i,k}}(\psi(Q_j)), \tag{7}$$

where $(f_{7,P_{i,k}}) = 7(P_{i,k}) - (-P'_{i,k}) - 6(\mathcal{O}),\ P_{0,k} = P_k,\ P_{i,k} = (\alpha_k{}^{(2i)} + 2id,\ (-1)^i\beta_k{}^{(2i)}) \sim 7^i[P_{0,k}]$ and $[-P'_{i,k}] = [7P_{i,k}]$ for $i \geq 1$.

We notice that $\prod_{k,j=1}^{3} v_{7,P_{i,k}}(\psi(Q_j))$ belongs to $\mathbb{F}_{7^{7n}}$, therefore $\prod_{k,j=1}^{3} v_{7,P_{i,k}}(\psi(Q_j))^{7^{7n-1}} = 1$. It thus follows that

$$\prod_{k,j=1}^{3} f_{7,P_{i,k}}(\psi(Q_j))^{7^{7n-1}} = \prod_{k,j=1}^{3} h_{7,P_{i,k}}(\psi(Q_j))^{7^{7n-1}}.$$

Consequently, we have

$$\eta(D,E)^{7^{7n-1}} = \left(\prod_{i=0}^{n-1} h_{7,D_i}(\psi(E))^{7^{n-1-i}}\right)^{7^{7n-1}}. \tag{8}$$

Algorithm 1 shows the pointwise computation of the Eta pairing.

Algorithm 1. Pointwise computation of the Eta pairing

INPUT: $D, E \in J_{H_d}(\mathbb{F}_{7^n})$
OUTPUT: $\eta(D,\ E)^{7^{7n-1}}$
1. $g \leftarrow 1$
2. Compute $P_k = (\alpha_k, \beta_k),\ k = 1, 2, 3$ and $Q_j = (x_j, y_j),\ j = 1, 2, 3$ which are supporting points for D and E, respectively.
3. For $i = 0$ to $n - 1$ do
4. For $k, j = 1$ to 3 do
5. compute $h_{k,j} = \beta_k^7 \cdot y_j \cdot \sigma - (\alpha_k^7 + x_j + d - \rho)^4$
6. set $\alpha_k \leftarrow \alpha_k^{7^2} + 2d,\ \beta_k \leftarrow \beta_k^{7^2}$
7. set $g \leftarrow g^7 \cdot \prod_{k,j=1}^{3} h_{k,j}$
8. Return $g^{7^{7n-1}} = \eta(D, E)^{7^{7n-1}}$.

Let M_i denote the time cost for a multiplication in $\mathbb{F}_{7^{in}}$ and S_i denote the time cost for a squaring in $\mathbb{F}_{7^{in}}$ for $i = 2, 3$. For simplicity, we assume that a squaring cost is similar to a multiplication cost, and we omit the computation cost for 7^{th} powering since it is negligible compared with the other operations.

We find the total complexity of Algorithm 1 as follows. In Step 5, for each of k, j, it requires two multiplications M_3 and two squarings S_3 in $\mathbb{F}_{7^{3n}}$; since $(A-\rho)^4 := (\alpha_k^7 + x_j + d - \rho)^4$ would require two S_3 and one M_3 for calculation of $A^2, A^3 = A \cdot A^2, (A^2)^2$. Thus the total for Step 5 is $9(2M_3 + 2S_3)$. On the other hand, Step 7 needs eight multiplications in $\mathbb{F}_{7^{3(14n)}}$ and one multiplication in $\mathbb{F}_{7^{14n}}$. For computing $\eta(D,\ E)$, the total complexity is therefore

$$T_P := \ 2T_{3rt} + n(9 \cdot 4M_3 + 8M_{42} + 1M_{14}) = 2T_{3rt} + n\ (36M_3 + 8M_{42} + 1M_{14}), \quad (9)$$

where T_{3rt} is the total time required for finding the supporting points of D and E.

3 Computation of the Eta Pairing by Using the Resultant

In this section, we use the resultant for the Eta pairing computation of the general divisors D and E. For given divisor inputs D, E with the Mumford representation, we want to be able to express all the intermediate formulas for the final Eta pairing value in terms of only the coefficients of the Mumford representations of D and E. An approach only by using the symmetric functions would end up with overly complicated formula. By using the resultant for the evaluation of a rational function at a divisor, we can make the computation steps much simpler and more explicit so that we can obtain an explicit algorithm. We analyze our algorithm for the case that D and E have supporting points in $H(\mathbb{F}_{7^{3n}})$, not in $H(\mathbb{F}_{7^n})$, and it turns out that this approach is faster than the pointwise method.

To obtain the value of $\eta(D, E)$, we find the explicit formulas for $D_i = [7^i]D$ and f_{7,D_i} for $i \geq 1$, and we also obtain the evaluation formula of rational function f_{7,D_i} at a divisor in a very explicit way.

Let D be a reduced divisor of H_d such that

$$D = (P_1) + (P_2) + (P_3) - 3\mathcal{O} = [U_D, V_D],$$

where $P_j = (\alpha_j, \beta_j)$ for $j = 1, 2, 3$, $U_D = x^3 + u_{D,2}x^2 + u_{D,1}x + u_{D,0}$, and $V_D = v_{D,2}x^2 + v_{D,1}x + v_{D,0} \in \mathbb{F}_{7^n}[x]$. Let $D_0 = D$, $D_{i+1} + (f_{7,D_i}) = 7D_i$, and $D_i = [U_{D_i}, V_{D_i}]$ for each positive integer i.

The following lemma provides us with explicit formulas for U_{D_i} and V_{D_i} in terms of the coefficients of U_D and V_D for $i \geq 1$. The proof can be obtained from the knowledge of Section 5 in Appendix of [19].

Lemma 2. *Let* [7] *be the multiplication map by 7 on the divisor class group of* $H_d/\mathbb{F}_{7^n}$. *Then we have, for* $i \geq 1$, $[7^i]D = D_i = [U_{D_i}, V_{D_i}]$ *with*

$$U_{D_i} = x^3 + (u_{D,2}^{(2i)} + id)x^2 + (u_{D,1}^{(2i)} + 3idu_{D,2}^{(2i)} - 2i^2)x + u_{D,0}^{(2i)} - 2idu_{D,1}^{(2i)} - 3i^2u_{D,2}^{(2i)} - i^3d,$$
$$V_{D_i} = (-1)^i v_{D,2}^{(2i)}x^2 + (-1)^i(3idv_{D,2}^{(2i)} + v_{D,1}^{(2i)})x + (-1)^i(-3i^2v_{D,2}^{(2i)} - 2idv_{D,1}^{(2i)} + v_{D,0}^{(2i)}).$$

For any divisor $E = [U_E, V_E]$ in $J_{H_d}(\mathbb{F}_q)$, the endomorphism ψ in Eq. (3) on divisors are easily deduced as follows: $\psi(E) = [U_{\psi(E)}, V_{\psi(E)}]$, where

$$U_{\psi(E)} = x^3 - (3\rho + u_{E,2})x^2 + (3\rho^2 + 2u_{E,2}\rho + u_{E,1})x - (\rho^3 + u_{E,2}\rho^2 + u_{E,1}\rho + u_{E,0}),$$
$$V_{\psi(E)} = \sigma(v_{E,2}x^2 - (2\rho v_{E,2} + v_{E,1})x + v_{E,2}\rho^2 + v_{E,1}\rho + v_{E,0}). \tag{10}$$

As seen in Eq. (8), it is sufficient to find h_{7,D_i}. In the following proposition, we thus find the function $h_{7,D}$ such that $D_1 + (h_{7,D}/v_{7,D}) = 7D$ in an explicit way.

Proposition 1. *Let* D *be a reduced divisor with* $D = [U_D, V_D]$ *and* τ *be a map*

$$\tau : H_d \to \tilde{H}_d, \quad (x, y) \to (X, Y) = (x - \xi, y), \quad \textit{where } \xi = 2u_{D,2}.$$

Then

(i) $\tilde{D} = \tau(D) = [X^3 + \tilde{u}_1X + \tilde{u}_0, \tilde{v}_2X^2 + \tilde{v}_1X + \tilde{v}_0]$, *where*

$$\tilde{u}_1 = 3\xi^2 + 2\xi u_{D,2} + u_{D,1}, \quad \tilde{u}_0 = \xi^3 + u_{D,2}\xi^2 + u_{D,1}\xi + u_{D,0}, \tag{11}$$
$$\tilde{v}_2 = v_{D,2}, \quad \tilde{v}_1 = 2\xi v_{D,2} + v_{D,1}, \quad \tilde{v}_0 = v_{D,2}\xi^2 + v_{D,1}\xi + v_{D,0}.$$

(ii)
$$h_{7,D}(x, y) = (\tilde{h}_{7,\tilde{D}} \circ \tau)(x, y),$$

where

$$\tilde{h}_{7,\tilde{D}}(X, Y) = \delta_1 Y^3 + s(Z)Y^2 + t(Z)Y - (Z^3 + \tilde{u}_1^7 Z + \tilde{u}_0^7)^4,$$

where $Z = X - \xi^7 + \xi + d$, *and* δ_1, $s(Z)$ *and* $t(Z)$ *are described in Table 5 of Appendix.*

Proof. Let $\tilde{H}_d$ be the image of H_d under the isomorphism τ. Since $D = [U_D, V_D] := \text{g.c.d.}(U_D,\ y - V_D)$ where $U_D(x),\ y - V_D(x) \in \mathbb{F}_{7^n}(H_d)$ (refer to [19]), we have $\tau(D) = \tilde{D} = [U_{\tilde{D}}, V_{\tilde{D}}] = \text{g.c.d.}(U_{\tilde{D}},\ Y - V_{\tilde{D}})$ where $U_{\tilde{D}} = U_D \circ \tau^{-1},\ (Y - V_{\tilde{D}}) = (y - V_D) \circ \tau^{-1} \in \mathbb{F}_{7^n}(\tilde{H}_d)$. Thus, we obtain Eq. (11) from the calculation of $U_D \circ \tau^{-1}$ and $(y - V_D) \circ \tau^{-1}$.

For $P = (\alpha, \beta) \in H_d$,

$$\begin{aligned} h_{7,P}(x,y) = h_{7,P}(\tau^{-1})(X,Y) &= \beta^7 Y - (\alpha^7 - X - \xi + d)^4 \\ &= \beta^7 Y - ((\alpha - \xi)^7 + \xi^7 - \xi + d - X)^4 \\ &= \tilde{\beta}^7 Y - (\tilde{\alpha}^7 + \xi^7 - \xi + d - X)^4 \\ &= \tilde{h}_{7,\tau(P)}(X,Y) \\ &= \tilde{h}_{7,\tau(P)}(\tau(x,y)). \end{aligned}$$

Thus, for $D = (P_1) + (P_2) + (P_3) - 3(\mathcal{O})$,

$$\begin{aligned} h_{7,D}(x,y) &= h_{7,P_1}(x,y) h_{7,P_2}(x,y) h_{7,P_3}(x,y) \\ &= \prod_{j=1}^{3} \tilde{h}_{7,\tau(P_j)}(\tau(x,y)) \\ &= (\prod_{j=1}^{3} \tilde{h}_{7,\tau(P_j)})(\tau(x,y)) \\ &= \tilde{h}_{7,\tilde{D}}(\tau(x,y)), \end{aligned}$$

where

$$\tilde{h}_{7,\tilde{D}}(X,Y) = \prod_{j=1}^{3} \left(\tilde{\beta}_j^7 Y - (\tilde{\alpha}_j^7 + \xi^7 - \xi + d - X)^4 \right). \tag{12}$$

If we apply the *Elimination method* in [8] to Eq. (12) with elimination order $\{\tilde{\alpha}_1, \tilde{\alpha}_2, \tilde{\alpha}_3\} > \{\tilde{u}_1, \tilde{u}_0\}$, then we can obtain Eq. (12) as a function of $\tilde{u}_1$ and $\tilde{u}_0$. The coefficients for $\tilde{h}_{7,\tilde{D}}(X,Y)$ are described in Table 5, where the second column shows the corresponding coefficients in terms of $\tilde{v}_i$ and $\tilde{u}_i$ with $i = 0, 1, 2$. □

Remark: For computing $h_{7,D}$, we can use resultant instead of elimination. If we apply elimination or resultant directly to $U_D(x)$, then the formulae for $h_{7,D}$ is a huge polynomial in terms of the coefficients of U_D and V_D. The translation τ plays an important role to reduce the size of the formulae.

Now we use resultant to evaluate a rational function at a divisor, which is necessary to achieve our goal. For the definition of resultant and its properties, we refer to [29, Ch. VI].

Theorem 1 ([29]). *Let F be a field. For $A, B \in F[x]$ with deg $A = m$, deg $B = n$, we have*

$$res(A,B) = a^n \prod_{i=1}^{m} B(\alpha_i),$$

where $\alpha_1, \alpha_2, \cdots, \alpha_m \in \bar{F}$ (= algebraic closure of F) are all the roots of A and a is the leading coefficient of A.

With the same notations as in Theorem 1, furthermore, we have

$$res(A, B) = (-1)^{mn} res(B, A). \tag{13}$$

In addition, efficient reduction method for computing the resultant is also introduced in [29, Ch. VI]. When $m \geq n$, by Euclidean division algorithm, there exists $Q(x), R(x) \in F(x)$ such that $A(x) = Q(x)B(x) + R(x)$ with $\deg R < n$. Then

$$res(A, B) = (-1)^{mn} res(B, R). \tag{14}$$

Now we are ready to use the resultant for the Eta pairing computation.

Theorem 2. *Let D, E be divisors of the curve H_d defined by $D = [U_D, V_D]$, $E = [U_E, V_E]$. Let τ_λ be a map $\tau_\lambda : (x, y) \to (x - \lambda, y)$.*

(i) Let $\xi_i = 2u_{D_i,2}$, $\theta_i = \xi^7 - \xi + d$ and $\rho' = \rho - d$.
Then we have $\hat{E}_i = \tau_{\xi_i+\theta_i} \circ \psi(E) = [U_{\hat{E}}(\hat{X}), V_{\hat{E}}(\hat{X})]$ such that

$$\begin{aligned}
u_{\hat{E},2} &= -3\rho' - (\xi_i + \theta_i + u_{E,2}),\\
u_{\hat{E},1} &= -3\rho'^2 - (2(\xi_i + \theta_i) + 2u_{E,2})\rho' + (2(\xi_i + \theta_i)^2 - 3(\xi_i + \theta_i)u_{E,2} - u_{E,1}),\\
u_{\hat{E},0} &= -\rho'^3 - (u_{E,2})\rho'^2 + (2(\xi_i + \theta_i)^2 - 3(\xi_i + \theta_i)u_{E,2} - u_{E,1})\rho'\\
&\quad - ((\xi_i + \theta_i)^3 + 3(\xi_i + \theta_i)^2 u_{E,2} + 2(\xi_i + \theta_i)u_{E,1} - u_{E,0}),\\
v_{\hat{E},2} &= \sigma v_{E,2},\\
v_{\hat{E},1} &= -\sigma(2v_{E,2}\rho' + \hat{v}_1), \qquad \hat{v}_1 = 2v_{E,2}(\xi_i + \theta_i) + v_{E,1},\\
v_{\hat{E},0} &= \sigma(v_{E,2}\rho'^2 + \hat{v}_1\rho' + \hat{v}_0), \qquad \hat{v}_0 = v_{E,2}(\xi_i + \theta_i)^2 + v_{E,1}(\xi_i + \theta_i) + v_{E,0}.
\end{aligned} \tag{15}$$

(ii) Let $\tilde{D}_i = \tau_{\xi_i}(D_i)$. Then $\tilde{h}_{7,\tilde{D}_i} = (-1)^i \delta_1^{7^{2i}} Y^3 + s_i(Z)Y^2 + t_i(Z)Y + w_i(Z)$, where $Z_i = X - \theta_i$,

$$s_i(Z) = \sum_{j=2}^{6} \delta_j^{7^{2i}} Z_i^{6-j},\ t_i(Z) = (-1)^i \sum_{j=7}^{15} (-1)^i \delta_j^{7^{2i}} Z_i^{15-j},\ w_i(Z) = -(Z^3 + \tilde{u}_1^{7^{2i+1}} Z_i + \tilde{u}_0^{7^{2i}}$$

and δ_j's are given in Table 5.

(iii) The Eta pairing of D and E is given by

$$\eta(D, E)^{7^{7n-1}} = \left(\prod_{i=0}^{n-1} res(U_{\hat{E}_i},\ H_{\tilde{D}_i,\hat{E}} \bmod U_{\hat{E}_i})^{7^{n-i-1}} \right)^{7^{7n-1}}, \tag{16}$$

where $H_{\tilde{D},\hat{E}}$ is given in Table 6 of Appendix.

Proof. (i) Eq. (15) is obtained from the calculations of $U_E \circ \tau_{\xi_i+\theta_i}^{-1} \circ \psi^{-1}$ and $(y - V_E) \circ \tau_{\xi_i+\theta_i}^{-1} \circ \psi^{-1}$.

(ii) We can compute $\tilde{h}_{7,\tilde{D}_i}$ by using Table 5 taking D_i as an input. Let $\delta_{i,j}$, $j = 1, ..., 15$, be the coefficients of $\tilde{h}_{7,\tilde{D}_i}$ for each D_i described in Table 5. Then from Lemma 2, we can compute the following values:

$$\begin{aligned} \xi_i &= \xi_0^{7^{2i}}, \\ \tilde{u}_{i,j} &= \tilde{u}_{0,j}^{7^{2i}}, \ j = 0, 1, \\ \tilde{v}_{i,j} &= (-1)^i \tilde{v}_{0,j}, \ \text{for } j = 0, 1, 2. \end{aligned}$$

These relations give the following simple expressions for $\delta_{i,1}, ..., \delta_{i,15}$:

$$\delta_{i,j} = \begin{cases} \delta_{0,j}^{7^{2i}} & \text{if } j = 2, 3, 4, 5, 6 \\ (-1)^i \delta_{0,j}^{7^{2i}} & \text{otherwise.} \end{cases} \tag{17}$$

(iii) We recall

$$\eta(D, E)^{7^{7n-1}} = \left(\prod_{i=0}^{n-1} h_{7,D_i}(\psi(E))^{7^{n-i-1}} \right)^{7^{7n-1}}.$$

Using Theorem 1 and Proposition 1, we have the evaluation $h_{7,D_i}(\psi(E))$ as following:

$$\begin{aligned} h_{7,D_i}(\psi(E)) &= \prod_{j=1}^{3} h_{7,D_i}(\psi(Q_j)) \\ &= res(h_{7,D_i}(\psi(x, V_E(x))), \ U_E(x)). \end{aligned} \tag{18}$$

Since

$$h_{7,D_i}(\psi(E)) = h_{7,D_i} \circ \tau_{\xi_i}^{-1} \circ \tau_{\theta_i}^{-1} \circ \tau_{\xi_i+\theta_i}(\psi(E)),$$

Eq. (18) equals to

$$res(H_{\tilde{D},\hat{E}}, U_{\hat{E}}),$$

where $H_{\tilde{D},\hat{E}} = h_{7,D_i} \circ \tau_{\xi_i}^{-1}$ and $\hat{E} = \tau_{\xi+\theta_i} \circ \psi(E)$.

Now by using the reduction method in Eq. (14), we can compute $res(H_{\tilde{D},\hat{E}}(\hat{X}), U_{\hat{E}}(\hat{X}))$ by

$$res(U_{\hat{E}}, H_{\tilde{D},\hat{E}}) = res(U_{\hat{E}}, R_i),$$

where $R_i = H_{\tilde{D},\hat{E}} \bmod U_{\hat{E}}$. □

Now we describe an algorithm for computing the Eta pairing on divisors, and we also compute its complexity. From Theorem 2, the Eta pairing given in Eq. (5)

Algorithm 2. Eta pairing computation by using resultant

INPUT: $D = [U_D,\ V_D],\ E = [U_E,\ V_E] \in J_{H_d}(\mathbb{F}_{7^n})$, endomorphism ψ
OUTPUT: $\eta(D,\ E)^{7^{7n-1}}$
1. Set $\xi \leftarrow 2u_{D,2}$ and $\theta \leftarrow \xi^7 - \xi + d$.
2. Compute $u_1 = 3\xi^2 + 2\xi u_{D,2} + u_{D,1}$ and $u_0 = \xi^3 + u_{D,2}\xi^2 + u_{D,1}\xi + u_{D,0}$.
3. Compute δ_j for $j = 1, ..., 15$ using Table 5.
4. $g \leftarrow 1$,
5. for $i = 0$ to $n-1$ do
6. compute $\hat{E} = \tau_{\xi+\theta}(\psi(E))$
7. compute $H_{\tilde{D},\hat{E}}$ and $R = H_{\tilde{D},\hat{E}} \pmod{U_{\hat{E}}}$ (Table 6).
8. compute $h_{7,D_i}(\psi(E)) = res(U_{\hat{E}}, R)$.
9. $g \leftarrow g^7 \cdot h_{7,D_i}(\psi(E))$
10. set $u_0 \leftarrow u_0^{7^2},\ u_1 \leftarrow u_1^{7^2}$
11. set $\xi \leftarrow \xi^{7^2},\ \theta \leftarrow \theta^{7^2},\ \delta_j \leftarrow \delta_j^{7^2}$ if $j = 2, 3, 4, 5, 6$, and $\delta_j \leftarrow (-1)^i \delta_j^{7^2}$ otherwise.
12. Return $g^{7^{7n-1}} = \eta(D,\ E)^{7^{7n-1}}$.

can be computed by using Algorithm 2. Since $v_{\hat{E},j} = \sigma \cdot$ (some element in $\mathbb{F}_{7^{7n}}$), $j = 0, 1, 2$, we note that $H_{\tilde{D},\hat{E}}$ in the step 7 of Algorithm 2 can be written as

$$H_{\tilde{D},\hat{E}} = -x^{12} + \sum_{i=0}^{10}(d_i\sigma + e_i)x^i,\ \ d_i, e_i \in \mathbb{F}_{7^{7n}} \text{ for } 0 \le i \le 10.$$

To find $H_{\tilde{D},\hat{E}} \pmod{U_{\hat{E}}}$ in the step 7, we use the following recursive relations:

$$\begin{aligned}
&\xi_i = u_{D,2}^{7^{2i+1}} x^i \equiv a_i x^2 + b_i x + c_i \pmod{U_E},\ \ 3 \le i \le 12,\\
&a_3 = -u_{\hat{E},2},\ \ b_3 = -u_{\hat{E},1},\ \ c_3 = -u_{\hat{E},0},\\
&a_i = a_{i-1}a_3 + b_{i-1},\ b_i = a_{i-1}b_3 + c_{i-1},\ c_i = a_{i-1}c_3.
\end{aligned}$$

Then R can be computed by

$$\begin{aligned}
R &= H_{\tilde{D},\hat{E}} \pmod{U_{\hat{E}}}\\
&= (\sigma(d_2 + \sum_{i=3}^{10} a_i d_i) + (a_{12} + e_2 + \sum_{i=3}^{10} a_i e_i))x^2\\
&+ (\sigma(d_1 + \sum_{i=3}^{10} b_i d_i) + (b_{12} + e_1 + \sum_{i=3}^{10} b_i e_i))x\\
&+ (\sigma(d_0 + \sum_{i=3}^{10} c_i d_i) + (c_{12} + e_0 + \sum_{i=3}^{10} c_i e_i)).
\end{aligned} \tag{19}$$

Now we discuss the complexity of Algorithm 2 by counting the number of operations which are necessary for computing $\eta(D, E)$. We denote the time for multiplications in $\mathbb{F}_{7^{14n}}, \mathbb{F}_{7^{7n}}$ and $\mathbb{F}_{7^n}$ by M_{14}, M_7 and M, respectively. We also $M_{1,7}$ denote the time cost for a multiplication between $\mathbb{F}_{7^n}$ and $\mathbb{F}_{7^{7n}}$. Noting that, in Step 6, $u_{\hat{E},j}, j = 0, 1, 2$ and $v_{\hat{E},0}, v_{\hat{E},1}$ belong to $\mathbb{F}_{7^{7n}}$, and from Eq. (10)

we have $v_{\hat{E},2} = \sigma \cdot$ (some element in $\mathbb{F}_{7^n}$). The computation cost of $H_{\tilde{D},\hat{E}}$ in Step 7 is counted in Table 6. We need $7M + 18M_7$ for computing $x^i \pmod{U_E}$ since we do not need the computation x^{11}, and we need $48M_7$ to compute R in Eq. (19). Furthermore, we need $3M$ in the step 2, and $37M$ in the step 3 from Table 5. For each loop, we need $5M$ in the step 6, $17M + 31M_{1,7} + 4M_7$ in the step 7 from Table 6, $1M_{14}$ in the step 9. The total complexity of this algorithm is therefore

$$40M + n(29M + 31M_{1,7} + 70M_7 + T_{res} + 1M_{14}), \tag{20}$$

where T_{res} is the computation cost for the resultant $res(U_{\hat{E}}, R)$ of $U_{\hat{E}}$ and R in $\mathbb{F}_{7^{14n}}$. We calculate the resultant by computing the determinant of two polynomials with degree 2 and 3 in Mumford representation. Then we have $T_{res} = 48M_7$.

4 Complexity Comparison

In this section we compare the complexities of our two methods given in Section 2 and 3.

When an extension degree is of the form $k = 2^i 3^j$, the computation cost for a multiplication in $\mathbb{F}_{q^k}$ is theoretically $3^i 5^j$ times of the cost for a multiplication $\mathbb{F}_q$ ([18], [20]). From this observation, we assume that

$$\begin{aligned} &1 \text{ mult. in } \mathbb{F}_{7^{3n}}(M_3) \approx 5M, \qquad 1 \text{ mult. in } \mathbb{F}_{7^{3(7n)}}(M_{21}) \approx 5M_7, \\ &1 \text{ mult. in } \mathbb{F}_{7^{14n}}(M_{14}) \approx 3M_7, \end{aligned} \tag{21}$$

and we also let $M_{1,7} \approx 7M$.

With the above assumptions, the pointwise computation cost in Eq. (9) is $T_P := 2T_{3rt} + n(36 \cdot 5M + 123M_7)$, where T_{3rt} is the time for finding all the roots of a cubic polynomial over $\mathbb{F}_{7^{3n}}$. By *Berlekamp-Rabin algorithm* [4], we have $T_{3rt} = O(3^2 \log 3 \log 7^{3n}) \cdot M_3 \approx 27n \cdot 4 \cdot M_3 = 108nM_3$.

Counting the cost for T_{3rt}, we finally have

$$T_P \approx n(1260M + 123M_7). \tag{22}$$

On the other hand, the total time for the resultant method in Eq. (20) is $T_R := 40M + n(246M + 70M_7 + T_{res} + 1M_{14})$, where T_{res} is the time for computing the resultant of two polynomials over $\mathbb{F}_{7^{14n}}$. As mentioned in Section 3, we have $T_{res} = 48M_7$. Thus, the computation cost of our resultant approach is approximately

$$T_R = 40M + n(246M + 121M_7). \tag{23}$$

To analyze T_P and T_R, we need to estimate the ratio of M_7 and M. According to [18, Section 4.3], there are cases for which a multiplication in $\mathbb{F}_{q^m}$ can be done with m multiplications in $\mathbb{F}_q$. Therefore, we estimate $7M \leq M_7 \leq 49M$.

To compare the complexities of two methods, we summarize T_P, T_R and the ratio T_P/T_R in Table 1 for a few security levels [20]. The last row of Table 1 shows

Table 1. Complexity comparison

security (bits)	80	128	192
bitlength of 7^{14n}	1140	3072	8192
n	29	79	211
pointwise (T_P)	$36540m + 3567M_7$	$99540m + 9717M_7$	$265860m + 25953M_7$
resultant (T_R)	$7174m + 3509M_7$	$19474m + 9559M_7$	$51946m + 25531M_7$
$\frac{T_P}{T_R}$	$1.17982 \leq \frac{T_P}{T_R} \leq 1.93808$	$1.17998 \leq \frac{T_P}{T_R} \leq 1.93963$	$1.18004 \leq \frac{T_P}{T_R} \leq 1.94019$

the range of the ratio $\frac{T_P}{T_R}$. We can conclude that the Eta pairing computation using resultant is 48.5% faster than the pointwise computation in the best case and 15.3% faster in the worst case. For fixed n, as M_7/M decreases, the ratio T_P/T_R increases. This implies that better performance of a multiplication in $\mathbb{F}_{7^{7n}}$ makes resultant method more efficient than pointwise method on general divisors which split in $\mathbb{F}_{7^{3n}}$, neither in $\mathbb{F}_{7^{2n}}$ nor in $\mathbb{F}_{7^n}$. Furthermore, we observe that when M_7/M is fixed, as n is increasing, $\frac{T_P}{T_R}$ is also increasing. Thus, for higher security level, the resultant method gives better efficiency than the pointwise method.

5 Experimental Results

We proposed two methods by the resultant and pointwise approach for computing the Eta pairing over the genus 3 hyperelliptic curve $H_d : y^2 = x^7 - x + d, d = \pm 1$ over $\mathbb{F}_{7^n}$. Our analysis in Section 4 showed that the resultant approach is up to 48.5% faster than the pointwise approach. In this section, we provide experiment results based on our implementation of the methods using NTL software package. Ours is the first implementation for the Eta pairing computation for genus 3 hyperelliptic curves.

We measure M, M_3 and M_7, the three important parameters used in the analysis in Section 4, in NTL. Then we measure the running times of the implementations of the methods and compare the results with our analysis.

For each security level s, we first need to find a prime n such that $2^s \approx 7^{3n}$, and also find a large prime ℓ dividing $|J_{H_d}(\mathbb{F}_{7^n})|$ such that $\ell \approx 2^s$. The formula for $|J_{H_d}(\mathbb{F}_{7^n})|$ is given in Eq. (2). By searching for good candidates for ℓ and n from $n = 29$ through $n = 79$, we find the four values of n, namely, $29, 43, 47$ and 73 with corresponding primes as given in Appendix.

Table 2 shows the amount of time to perform the field multiplications in $\mathbb{F}_{7^n}$, $\mathbb{F}_{7^{3n}}$ and $\mathbb{F}_{7^{7n}}$ using NTL. In detail, we used class ZZ_pE for finite field operations in F_{7^n}, and we used class ZZ_pEX for modular polynomial arithmetic to implement operations in $F_{7^{3n}}$ and $F_{7^{7n}}$. The table was computed by taking average time of 5000 multiplications of random elements in each field. According to Table 2, the speed of field operations in NTL is not quite optimal for cryptographic applications. However, our goal is to compare efficiency of two algorithms depending on field operations, and therefore using NTL is sufficient for our purpose since both algorithms are implemented on the same library.

In Section 4, we assumed the ratio M_3/M is 5 for the field operations M_3 in $\mathbb{F}_{7^{3n}}$ and M in $\mathbb{F}_{7^n}$. However, in NTL the actual ratio M_3/M is approximately 7

Table 2. Multiplication timings (in milliseconds)

n	29	43	47	73
$\mathbb{F}_{7^n}$ (M)	0.2562	0.5686	0.6626	1.6062
$\mathbb{F}_{7^{3n}}(M_3)$	2.4	4.3218	4.8906	11.1562
$\mathbb{F}_{7^{7n}}$ (M_7)	7.8438	16.672	17.5156	27.8688
M_3/M	9.36768	7.60077	7.38092	6.94571
M_7/M	30.6159	29.3211	26.4347	17.3508

or 9 as shown in Table 2, and M_7/M in Table 2 is in the range $7 \leq M_7/M \leq 49$ as we expected. In our implementation M_{42} is optimized to $56M_3$. Therefore, each complexity for the Eta pairing computation is given by $T_R = 40M + n(246M + 121M_7)$ and

$$T_P = 2\,T_{3rt} + n\,(36M_3 + 8M_{42} + 1M) = 2T_{3rt} + n(36M_3 + 56{\cdot}8M_3 + 3M_7) \approx n(700M_3 + 3M_7).$$

According to the actual ratio of the field operations in NTL, Table 1 is adjusted to obtain Table 3. As shown in Table 3, for $n \geq 43$, as n increases, T_P/T_R also increases as we expected in Section 4. On the other hand, when n increases from 29 to 43, T_P/T_R decreases, which is opposite to what we expected, and we guess that the reason is the following: For instance, we observe that when n changes from 29 to 43, the decrement of M_3/M (resp. M_7/M) is 1.767 (resp. 1.295). On the other hand, when n changes from 43 to 47, the decrement of M_3/M (resp. M_7/M) is 0.220 (resp. 2.886). So, the decrement of M_3/M is much larger than M_7/M when n changes from 29 to 43, while M_3/M is much smaller than M_7/M when n changes from 29 to 43.

From Table 3, the resultant method is 40.57% (resp. 29.38%, 34.32%, and 52.26%) faster than the pointwise method for $n = 29$ (resp. 43, 47, and 73). These examples support our theoretical complexity analysis in Section 4, that is, for higher security level the resultant method is more efficient than the pointwise method for the Eta pairing computation.

Table 3. Complexity comparison: examples in NTL

bitlength of 7^{14n}	1140	1690	1847	2869
n	29	43	47	73
pointwise (T_P)	$20300M_3 + 87M_7$	$30100M_3 + 129M_7$	$32900M_3 + 141M_7$	$51100M_3 + 219M_7$
resultant (T_R)	$7174M + 3509M_7$	$10618M + 5203M_7$	$11602M + 5687M_7$	$17998M + 8833M_7$
$\frac{T_P}{T_R}$	1.68254	1.42525	1.52257	2.09465

Table 4 shows the implementation results of the Eta pairing for selected examples. The resultant method is 48.8% (resp. 39.1%, 38.7%, and 43.4%) faster than the pointwise method for $n = 29$ (resp. 43, 47, and 73). We performed fifty calculations with random samples for each method and took the average time.

Table 4. Experimental results (in seconds)

bit-length of ℓ	237	338	373	608
bit-length of 7^{14n}	1140	1690	1847	2869
n	29	43	47	73
pointwise method(T_P)	38.3564	99.2936	120.672	399.962
resultant method(T_R)	19.6315	60.4622	73.9278	226.412
$\frac{T_P}{T_R}$	1.95382	1.64224	1.63229	1.76652

The experiments ran on a machine with 2.2Ghz Opteron, and we used Microsoft Visual C++ 6.0.

Our implementation shows that the performance ratio $\frac{T_P}{T_R}$ for $n = 29$ is larger than the ratio for $n = 73$, while the theoretical complexity analysis in Table 1 shows the other way around. This difference occurs because of the relative time cost $\frac{T_{3rt}}{T_P}$ for computing all the supporting points of an input divisor. In more detail, in the theoretical complexity analysis, for $\frac{T_{3rt}}{T_P}$ we have a flat ratio approximately 0.152. On the other hand, in our implementation, $\frac{T_{3rt}}{T_P}$ is 0.447242 (resp. $0.436616, 0.41533$, and 0.170231) for $n = 29$ (resp. $43, 47$, and 73). Thus the ratio $\frac{T_{3rt}}{T_P}$ is various depending on the values of n, and in fact, the ratio is largest when $n = 29$ and smallest when $n = 73$.

6 Conclusions and Future Work

In this paper, we present two algorithms for computing the Eta pairing on *general* divisors over hyperelliptic curves $y^2 = x^7 - x \pm 1$. We compare complexities of two algorithms on the case that both divisors of the Eta pairing consist of all the supporting points in $H(\mathbb{F}_{7^{3n}})$, but not in $H(\mathbb{F}_{7^n})$. Our analysis shows that, in this case, the resultant approach is up to 48.5 % faster than the pointwise approach.

In this work for complexity analysis, we focus on the case that the general divisors D and E have all the supporting points in $H_d(\mathbb{F}_{7^{3n}})$, but not in $H(\mathbb{F}_{7^n})$. However, there are other possibilities, that is, a general divisor D consists of two points in $H_d(\mathbb{F}_{q^2})$ and the other in $H_d(\mathbb{F}_q)$, or all three points in $H_d(\mathbb{F}_q)$, depending on how $U_D(x)$ in $\mathbb{F}_q[x]$ splits. As we also have the same possibilities for E, we have nine possible cases for D and E. In the near future, we will analyze the theoretical complexity in each of nine cases for both methods. Furthermore, we will implement the resultant method using more efficient library for field operations. Since we expect that the time for a multiplication in $\mathbb{F}_{7^n}$ can be improved almost 100 times faster than NTL library, the implementation using such an efficient library would provide a cryptographically meaningful timing result.

Acknowledgment. We thank anonymous referees for their helpful comments.

References

1. Araki, K., Miura, S., Satoh, T.: Overview of elliptic curve crptography. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 29–49. Springer, Heidelberg (1998)
2. 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)
3. Barreto, P.S.L.M., Galbraith, S., ÓhÉigeartaigh, C., Scott, M.: Efficient Pairing Computation on Supersingular Abelian Varieties. Des. Codes Cryptogr. 42, 239–271 (2007)
4. Berlekamp, E.R: Factoring polynomials over large finite fields. Math. Comp. 24, 713–735 (1970)
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. 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 (2002)
7. Choie, Y., Lee, E.: Implementation of Tate pairing on hyperelliptic curves of genus 2. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 97–111. Springer, Heidelberg (2004)
8. Cox, D., Little, J., O'Shea, D.: Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra: with 91 illustrations. Springer, New York (1997)
9. Duursma, I.: Class numbers for hyperelliptic curves. In: Pellikaan, Perret, Vladuts (eds.) Arithmetic, Geometry and Coding Theory, pp. 45–52. deGruyter, Berlin (1996)
10. Duursma, I., Sakurai, K.: Efficient algorithms for the Jacobian variety of hyperelliptic curves $y^2 = x^p - x + 1$ over a finite field of odd characteristic p. In: Coding theory, cryptography and related areas (Guanajuato, 1998) pp. 73–89. Springer, Heidelberg (2000)
11. Duursma, I., Lee, H.: Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
12. Frey, G., Rück, H.-G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comp. 62(206), 865–874 (1994)
13. Galbraith, S.D., Harrison, K., Soldera, D.: Implementing the Tate pairing. In: Fieker, C., Kohel, D.R. (eds.) Algorithmic Number Theory - V. LNCS, vol. 2369, pp. 324–337. Springer, Heidelberg (2002)
14. Greuel, G.-M., Pfister, G., Schönemann, H.: SINGULAR 3.0. A Computer Algebra System for PolynomialComputations. Centre for Computer Algebra, University ofKaiserslautern (2005) http://www.singular.uni-kl.de
15. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: Bosma, W. (ed.) Algorithmic Number Theory IV. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
16. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. IEEE Trans. Information Theory 52, 4595–4602 (2006)
17. Karatsuba, A., Ofman, Y.: Multiplication of Multidigit Numbers on Automata (Engl. transl.) Sov. Phys.-Dokl. 7(7), 595–596 (1963)

18. Knuth, D.E.: The Art of Computer Programming, vol. II. Addison Wesley, London (2004)
19. Koblitz, N.: Algebraic Aspects of Cryptography. Springer, Heidelberg (1998)
20. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding. LNCS, vol. 3796, pp. 3–36. Springer, Heidelberg (2005)
21. Kwon, S.: Efficient Tate Pairing Computation for Elliptic Curves over Binary Fields. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 134–145. Springer, Heidelberg (2005)
22. Lange, T.: Formulae for arithmetic on genus 2 hyperelliptic curves. Appl. Algebra Engrg. Comm. Comput. 15(5), 295–328 (2005)
23. Mumford, D.: Tata Lectures on Theta II, Birkhauser (1984)
24. Paterson, K.G.: ID-based signature from pairings on elliptic curves. Electronics Letters 38(18), 1025–1026 (2002)
25. Pelzl, J., Wollinger, T., Paar, C.: Low cost security: explicit formulae for genus-4 hyperelliptic curves. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 1–16. Springer, Heidelberg (2004)
26. Shoup, V.: A library for doing number theory, Software (2001) http://www.shoup.net/ntl/
27. Smart, N.: On the Performance of Hyperelliptic Cryptosystems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 165–175. Springer, Heidelberg (1999)
28. Smart, N.P.: An identity based authentication key agreement protocol based on pairing. Electronics Letters 38, 630–632 (2002)
29. Yap, C.K.: Fundamental Problems in Algorithmic Algebra. Oxford University Press, Oxford (2000)

Appendix

In Section 5, by searching for good candidates for ℓ and n from $n = 29$ through $n = 79$, we find the following:

When $n = 29$, for $H_{(-1)}$ curve,

$\ell = 29542758054398104450874217525165651042521871765435101109943075021065009 7$.

When $n = 43$, for $H_{(-1)}$ curve,

$\ell = 53718618569186388018821703986374275351705576366850017552481452390195758887874407533286287888356386 4467$.

When $n = 47$, for $H_{(-1)}$ curve,

$\ell = 13749772461004425111203179773331321128112017469863375270022695103034065149004498912831678964830780873139729982133$.

When $n = 73$, for $H_{(+1)}$ curve,

$\ell = 10553398064514656199046818606065495176614662671222319372367416319801315889940362184197553323184699007812855786020479789551940934976512907234753096204258803335765166769800421495325836 47$.

Table 5. $\tilde{h}_{7,\tilde{D}}$ formula for $\tilde{D}$

Input	$\tilde{D} = [X^3 + \tilde{u}_1 X + \tilde{u}_0, \tilde{v}_2 X^2 + \tilde{v}_1 X + \tilde{v}_0]$	Cost
Output	$\tilde{h}_{7,\tilde{D}}(X,Y) = \delta_1 Y^3 + s(Z)Y^2 + t(Z)Y - (Z^3 + \tilde{u}_1^7 Z + \tilde{u}_0^7)^4$ $s(Z) = \delta_2 Z^4 + \delta_3 Z^3 + \delta_4 Z^2 + \delta_5 Z + \delta_6$ $t(Z) = \delta_7 Z^8 + \delta_8 Z^7 + \delta_9 Z^6 + \delta_{10} Z^5 + \delta_{11} Z^4 + \delta_{12} Z^3 + \delta_{13} Z^2 + \delta_{14} Z + \delta_{15}$	
δ_1	$(\tilde{v}_2\tilde{u}_0(\tilde{v}_2(\tilde{v}_2\tilde{u}_0) + 3\tilde{v}_1(\tilde{v}_0 + 2\tilde{v}_2\tilde{u}_1)) + \tilde{v}_1^2(\tilde{v}_0\tilde{u}_1 - \tilde{v}_1\tilde{u}_0) + \tilde{v}_0(\tilde{v}_2\tilde{u}_1 - \tilde{v}_0)^2)^7$	$8M + 2S$
δ_2	$(4(2\tilde{v}_2\tilde{u}_1 + \tilde{v}_0)(-\tilde{v}_2\tilde{u}_1 + \tilde{v}_0) - \tilde{v}_1(3\tilde{v}_2\tilde{u}_0 + \tilde{v}_1\tilde{u}_1))^7$	$3M$
δ_3	$(-2\tilde{v}_2\tilde{u}_0(2\tilde{v}_2\tilde{u}_1 + \tilde{v}_0) + \tilde{v}_1(2\tilde{v}_2\tilde{u}_0 + \tilde{v}_0\tilde{u}_1))^7$	$2M$
δ_4	$(3\tilde{v}_2^2\tilde{u}_0^2 + \tilde{v}_1\tilde{u}_0(-2\tilde{v}_2\tilde{u}_1 + 3\tilde{v}_0) + \tilde{v}_0\tilde{u}_1(2\tilde{v}_2\tilde{u}_1 - 2\tilde{v}_0))^7$	$2M + 1S$
δ_5	$(\tilde{v}_2\tilde{v}_0(2\tilde{v}_1\tilde{v}_0 - 3\tilde{u}_1\tilde{u}_0) - \tilde{v}_1\tilde{u}_1(\tilde{v}_0\tilde{u}_1 - \tilde{v}_1\tilde{u}_0) + 2\tilde{v}_0^2\tilde{u}_0)^7$	$3M$
δ_6	$(2\tilde{u}_0(\tilde{v}_2\tilde{u}_0)(\tilde{v}_2\tilde{u}_1 - 2\tilde{v}_0) - (\tilde{v}_0\tilde{u}_1 + \tilde{v}_1\tilde{u}_0)(4\tilde{v}_1\tilde{u}_0 + 2\tilde{u}_1(\tilde{v}_0 - \tilde{v}_2\tilde{u}_1)))^7$	$4M$
δ_7	$(-2\tilde{v}_2\tilde{u}_1 + 3\tilde{v}_0)^7$	$0M$
δ_8	$(2\tilde{v}_2\tilde{u}_0 - \tilde{v}_1\tilde{u}_1)^7$	$0M$
δ_9	$(2\tilde{u}_1(\tilde{v}_0 - \tilde{v}_2\tilde{u}_1) + 2(2\tilde{v}_0\tilde{u}_1 - \tilde{v}_1\tilde{u}_0))^7$	$0M$
δ_{10}	$(\tilde{u}_1(2\tilde{v}_2\tilde{u}_0 + \tilde{v}_1\tilde{u}_1) + \tilde{v}_0\tilde{u}_0)^7$	$1M$
δ_{11}	$(\tilde{u}_1^2(-2\tilde{v}_2\tilde{u}_1 + \tilde{v}_0) + \tilde{u}_0(\tilde{v}_2\tilde{u}_0 + 2\tilde{v}_1\tilde{u}_1))^7$	$2M + 1S$
δ_{12}	$(\tilde{u}_0(3\tilde{u}_1(2\tilde{v}_2\tilde{u}_1 + \tilde{v}_0) - \tilde{v}_1\tilde{u}_0))^7$	$1M$
δ_{13}	$(\tilde{u}_0^2(-\tilde{v}_2\tilde{u}_1 + 2\tilde{v}_0) + 3\tilde{u}_1^2(\tilde{v}_1\tilde{u}_0 - \tilde{v}_0\tilde{u}_1))^7$	$2M + 1S$
δ_{14}	$(2\tilde{u}_0^2(\tilde{v}_2\tilde{u}_0 + 2\tilde{v}_1\tilde{u}_1) + 3(\tilde{u}_0\tilde{v}_0)\tilde{u}_1^2)^7$	$2M$
δ_{15}	$(\tilde{u}_1(\tilde{u}_0^2(-\tilde{v}_2\tilde{u}_1 + 2\tilde{v}_0) + 3\tilde{u}_1^2(\tilde{v}_1\tilde{u}_0 - \tilde{v}_0\tilde{u}_1)) + \tilde{u}_0^2(2\tilde{u}_1(\tilde{v}_2\tilde{u}_1 - \tilde{v}_0) + 4(\tilde{v}_0\tilde{u}_1 + \tilde{v}_1\tilde{u}_0))^7$	$2M$
Total cost	Notation: M denotes a multiplication in $\mathbb{F}_{7^n}$, and S a squaring in $\mathbb{F}_{7^n}$.	$32M + 5S$

Table 6. $H_{\tilde{D},\hat{E}}$ formula complexity counting

i	ith coefficient of $H_{\tilde{D}}$	Cost
12	-1	0
10	$\delta_7 v_{\hat{E},2} + 3\tilde{u}_1^7$	$1M$
9	$\delta_7 v_{\hat{E},1} + \delta_8 v_{\hat{E},2} + 3\tilde{u}_0^7$	$2M_{1,7}$
8	$\delta_2 v_{\hat{E},2}^2 + \delta_7 v_{\hat{E},0} + \delta_8 v_{\hat{E},1} + \delta_9 v_{\hat{E},2} + \tilde{u}_1^{14}$	$1M + 1S + 2M_{1,7}$
7	$2\delta_2 v_{\hat{E},1} v_{\hat{E},2} + \delta_3 v_{\hat{E},2}^2 + \delta_8 v_{\hat{E},0} + \delta_9 v_{\hat{E},1} + \delta_{10} v_{\hat{E},2} + 2\tilde{u}_0^7 \tilde{u}_1^7$	$2M + 3M_{1,7}$
6	$\delta_1 v_{\hat{E},2}^3 + 2\delta_2 v_{\hat{E},0} v_{\hat{E},2} + \delta_2 v_{\hat{E},1}^2 + 2\delta_3 v_{\hat{E},1} v_{\hat{E},2} + \delta_4 v_{\hat{E},2}^2 + \delta_9 v_{\hat{E},0}$ $+\delta_{10} v_{\hat{E},1} + \delta_{11} v_{\hat{E},2} + \tilde{u}_0^{14} + 3\tilde{u}_1^{21}$	$3M + 4M_{1,7} + 1S$
5	$3\delta_1 v_{\hat{E},1} v_{\hat{E},2}^2 + 2\delta_2 v_{\hat{E},0} v_{\hat{E},1} + 2\delta_3 v_{\hat{E},0} v_{\hat{E},2} + \delta_3 v_{\hat{E},1}^2 + 2\delta_4 v_{\hat{E},1} v_{\hat{E},2}$ $+\delta_5 v_{\hat{E},2}^2 + \delta_{10} v_{\hat{E},0} + \delta_{11} v_{\hat{E},1} + \delta_{12} v_{\hat{E},2} + 2\tilde{u}_0^7 \tilde{u}_1^{14}$	$7M_{1,7} + 1M$
4	$3\delta_1 v_{\hat{E},0} v_{\hat{E},2}^2 + 3\delta_1 v_{\hat{E},1}^2 v_{\hat{E},2} + \delta_2 v_{\hat{E},0}^2 + 2\delta_3 v_{\hat{E},0} v_{\hat{E},1} + 2\delta_4 v_{\hat{E},0} v_{\hat{E},2} + \delta_4 v_{\hat{E},1}^2$ $+2\delta_5 v_{\hat{E},1} v_{\hat{E},2} + \delta_6 v_{\hat{E},2}^2 + \delta_{11} v_{\hat{E},0} + \delta_{12} v_{\hat{E},1} + \delta_{13} v_{\hat{E},2} + 2\tilde{u}_0^{14} \tilde{u}_1^7 - \tilde{u}_1^{28}$	$5M_{1,7} + 2M$
3	$-\delta_1 v_{\hat{E},0} v_{\hat{E},1} v_{\hat{E},2} + \delta_1 v_{\hat{E},1}^3 + \delta_3 v_{\hat{E},0}^2 + 2\delta_4 v_{\hat{E},0} v_{\hat{E},1} + 2\delta_5 v_{\hat{E},0} v_{\hat{E},2} + \delta_5 v_{\hat{E},1}^2$ $+2\delta_6 v_{\hat{E},1} v_{\hat{E},2} + \delta_{12} v_{\hat{E},0} + \delta_{13} v_{\hat{E},1} + \delta_{14} v_{\hat{E},2} + 3\tilde{u}_0^{21} + 3\tilde{u}_0^7 \tilde{u}_1^{21}$	$1M_7 + 5M_{1,7} + 1M$
2	$(3\delta_1 v_{\hat{E},0}^2 v_{\hat{E},2} + 3\delta_1 v_{\hat{E},0} v_{\hat{E},1}^2 + \delta_4 v_{\hat{E},0}^2 + 2\delta_5 v_{\hat{E},0} v_{\hat{E},1} + 2\delta_6 v_{\hat{E},0} v_{\hat{E},2} + \delta_6 v_{\hat{E},1}^2$ $+\delta_{13} v_{\hat{E},0} + \delta_{14} v_{\hat{E},1} + \delta_{15} v_{\hat{E},2} + \tilde{u}_0^{14} \tilde{u}_1^{14}$	$1M_7 + 2M_{1,7} + 1M + 1S$
1	$3\delta_1 v_{\hat{E},0}^2 v_{\hat{E},1} + \delta_5 v_{\hat{E},0}^2 + 2\delta_6 v_{\hat{E},0} v_{\hat{E},1} + \delta_{14} v_{\hat{E},0} + \delta_{15} v_{\hat{E},1} + 3\tilde{u}_0^{21} \tilde{u}_1^7$	$1M + 1M_7 + 1M_{1,7}$
0	$\delta_1 v_{\hat{E},0}^3 + \delta_6 v_{\hat{E},0}^2 + \delta_{15} v_{\hat{E},0} - \tilde{u}_0^{28}$	$1M_7 + 1S$
	Total cost	$4S + 13M + 31M_{1,7} + 4M_7$

Provably Secure Pairing-Based Convertible Undeniable Signature with Short Signature Length*

Xinyi Huang, Yi Mu, Willy Susilo, and Wei Wu

Centre for Computer and Information Security Research
School of Computer Science & Software Engineering
University of Wollongong, Australia
{xh068,ymu,wsusilo}@uow.edu.au, weiwu81@gmail.com

Abstract. Undeniable signatures, introduced by Chaum and van Antwerpen, is a useful cryptography primitive to limit the publicly verifiable property of ordinary digital signatures. In an undeniable signature scheme, the validity or invalidity of the signature can only be verified via the confirmation/disavowal protocol with the help of the signer. An extended concept, convertible undeniable signatures, was introduced by Boyar, Chaum, Damgård and Pedersen. In the new concept, the signer can publish some selective proofs to convert one or more undeniable signatures into publicly verifiable ones, or issue a universal proof to make all his undeniable signatures publicly verifiable. In this paper, we first present a security model for convertible undeniable signature schemes, and then propose a new construction from bilinear pairings. Compared with the other schemes in the literature, the new construction has three advantages: Our scheme is both selectively and universally convertible; the signature length of our scheme is as short as BLS signature; meanwhile, all the security properties are formally proven under some conventional assumptions in the random oracle model.

Keywords: Undeniable Signatures, Convertible, Short Signature, Bilinear Pairings, Provable Security.

1 Introduction

Digital signatures, introduced in the pioneering paper of Diffie and Hellman [7], allow a signer with a secret key to sign messages such that anyone with access to the corresponding public key can verify the authenticity of the message. A signature verifier can convince any third party about this fact by presenting the digital signature on a message. The ease of copying and transmitting digital signatures in some implementations is of great convenience, but seems undesirable in some personally or commercially sensitive applications.

Undeniable signature is a concept introduced by Chaum and van Antwerpen in Cypto'89 [4]. In such kind of signatures, the validity or invalidity of an

* Supported by ARC Discovery Grant DP0557493 and DP0663306.

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 367–391, 2007.

undeniable signature can only be verified via the Confirmation/Disavowal protocol with the help of the signer. They are useful in the situations where the validity of a signature must not be publicly verifiable. For example, a software vendor might want to embed signatures into his products and allow only paying customers to check the authenticity of these products. If the vendor actually signed the message, he must be able to convince the customer of this fact using a confirmation protocol and, if he did not, he must also be able to convince the customer that he is not the signer via a disavowal protocol. These proofs have to be *non-transferable*: once a verifier is convinced that the vendor did or did not sign a message, he should be unable to transmit this conviction to another third party.

The first undeniable signature was proposed by Chaum and van Antwerpen [4] and it was further improved by Chaum in [5]. However, the unforgeability of the FDH (full domain hash) variant of Chaum's scheme remained as an open problem until Okamoto and Pointcheval [37] introduced a new class of computational problems: gap problems and proved the unforgeability of the FDH variant of Chaum's scheme is equivalent to the Gap Diffie-Hellman problem. Unfortunately, Ogata, Kurosawa and Heng [36] pointed out that the above claim in [37] is incorrect. Instead, they proved that the unforgeability is equivalent to the Computational Diffie-Hellman problem. As shown in the same paper, there are basically three kinds of Confirmation/Disavowal protocols: zero-knowledge interactive proof, 3-move honest verifier zero knowledge (HVZK) and non-interactive zero knowledge proof using the designated verifier techniques. In Eurocrypt 2005, Kurosawa and Heng [22] proposed the 3-move witness indistinguishable (WI) protocol for Chaum's undeniable signature and proved that the FDH variant of Chaum's scheme with this protocol is secure against active and concurrent attack. Using the Σ-compiler proposed by Furukawa, Kurosawa and Imai [19], one can obtain a very efficient Chaum's undeniable signature with 2-move confirmation protocol and disavowal protocol which are concurrent deniable zero-knowledge as well. Since the introduction of undeniable signatures, there have been several proposed schemes in the literature [1,3,8,10,12,13,14,15,17,21,25,26,27,29,30,32,40,41,42,43].

The concept of convertible undeniable signatures was introduced in [1] by Boyar, Chaum, Damgård and Pedersen, where the convertibility refers to the ability of the signer to convert one or more his undeniable signatures into publicly verifiable. "Convert" in the undeniable signatures has two types: **Selectively Convert** and **Universally Convert**. A signer can use the **Selectively Convert** algorithm to generate a proof for an undeniable signature. Then, one can check the validity of this signature using the proof and signer's public key. However, the validity of other undeniable signatures remains unknown and can only be verified via the confirmation/disavowal protocol with the help of the signer. The signer can also use the **Universally Convert** algorithm to generate a universal proof which can convert all his undeniable signatures into publicly verifiable ones. Thus, one can check the validity of any undeniable signature without the help of the signer. Convertible undeniable signatures have been found useful in

the applications such as the problem of keeping the digital records of confidential political decisions [8]. Unfortunately, the scheme proposed in [1] has been broken by Michels, Petersen and Horster [30] who proposed a repaired version with heuristic security[1]. In Eurocrypt'96, Damgård and Pedersen [8] proposed two convertible undeniable signature schemes, in which forging signatures is provably equivalent to forging El Gamal signature. An efficient convertible undeniable signature based on Schnorr signature was proposed by Michels and Stadler in [31]. The new scheme can be used as a basis of an efficient extension to threshold signature. Other constructions in RSA systems were also introduced. The first RSA based (convertible) undeniable signature was proposed by Gennaro, Rabin and Krawczyk in CRYPTO'97 [14], which was later improved by Miyazaki [29]. Very recently, Kurosawa and Takagi [23] proposed a new approach for constructing selectively convertible undeniable signature schemes, and presented two schemes based on RSA related assumptions. Furthermore, Kurosawa and Takagi's second scheme is the first selectively convertible scheme whose security can be proven without random oracles. Based on the computation of characters, Monnerat and Vaudenay proposed a novel construction of the undeniable signature which offers the advantage of having an arbitrarily short signature (depending on the required security level) [32]. Monnerat and Vaudenay also generalized and optimized their scheme in [33] and [34], respectively, and claimed that their scheme proposed in [33] can achieve the selective convertibility, without providing a formal security proof to support this claim. Laguillaumie and Vergnaud proposed a new convertible undeniable signature scheme from pairing [27]. The signature of their scheme only consists of an element of the group in elliptic curve and some additional random salt such that the total signature length can be as short as 272 bits. In addition, the first construction of Identity based selectively convertible undeniable signature was proposed by Libert and Quisquater. The details of the known convertible undeniable signatures are provided in the following table.

Scheme	Selectively Convert	Universally Convert	Security
Boyar*et al.*'s [1]	✓	✓	broken [30]
Damgård-Pedersen's [8]	✓	✓	DL related assumptions
Michels-Petersen-Horster's [30]	✓	✓	Proofs in generic group model [39]
Michels-Stadler's [31]	✓	✓	sketchy proof provided
Gennaro-Rabin-Krawczyk's [15]	✓	✓	RSA related assumptions
Miyazaki's [29]	✓	✓	sketchy proof provided
Libert-Quisquater's [25]	✓		pairing related assumptions
Monnerat-Vaudenay's [33]	✓		GHI related assumptions [33]
Laguillaumie-Vergnaud's [27]	✓	✓	pairing related assumptions
Kurosawa-Takagi's [23]	✓		RSA related assumptions

Motivations. When we add the "Convertible" property to undeniable signatures, it enables the signer to issue selective or universal proofs which can convert his undeniable signatures into self-authenticating. Thus, the signer does

[1] Very recently, Aimani and Vergnaud [39] provided the proof of Michels-Petersen-Horster's scheme [30] in the generic group model.

not require to convince the validity of his signatures to each verifier. However, these selective or universal proofs also provide some additional information to the adversaries and might help the latter to break the schemes. For example, Boyar-Chaum-Damgård-Pedersen's scheme [1] is insecure after the signer issues the universal proof. A key only adversary can forge the signature scheme universally after obtaining the universal proof. Therefore, how to define the security of convertible undeniable signatures is crucial. There are some security models of convertible undeniable signatures [8,23,27,31], but some properties still remain unclear. For example, the relationship between Anonymity and Invisibility has been proven closely related in undeniable signatures [13], however, there is no discussion about their relationship in the convertible undeniable signatures. Another example is the property of Non-Impersonation. In the original definition given by Kurosawa and Heng [22], this is equivalent to prevent impersonation by employing confirmation and disavowal protocols. In a convertible undeniable signature, this notion should be extended such that one should be unable to impersonate the signer to execute confirmation and disavowal protocols, or generate one of the selective proofs or the universal proof. Therefore, it is worthwhile to construct a security model to clearly define the security properties of the convertible undeniable signature and their relationships.

As shown at the above table, not all convertible undeniable signature schemes have the universal convertibility and some of them [25,33,23] are only selectively convertible. The schemes in [29,30,31] are universally convertible, but their proofs are not satisfying. There are only three schemes [8,15,27] which have formal proofs and meanwhile, can provide both selective and universal convertibility. Among these schemes, Laguillaumie-Vergnaud's scheme [27] is based on the pairings and has the shortest signature length. The unforgeability of the scheme in [27] is based on the hardness of a well known problem: Computational Diffie Hellman problem. However, the invisibility of their scheme is based on a non-standard decisional assumption: $(\ell, 1)$-xyz-**DCAA** assumption defined in [27]. How to obtain a short convertible undeniable signature with provable security under conventional assumptions is another motivation of this paper.

Our Contributions. In this paper, we first define a security model of convertible undeniable signatures. In the new model, all the properties of convertible undeniable signatures are defined formally. The relationship between the Invisibility and Anonymity in the convertible undeniable signature is also proved in this paper. We believe the new model is of independent interest.

We also present a new construction of convertible undeniable signature. Our new scheme is based on bilinear pairings and enjoys short signature length. Meanwhile, the new scheme is provably secure in the random oracle model. Formal proofs are provided to show that the new scheme satisfies all the security properties. Compared with the other schemes in the literature, the new construction has three advantages: Our scheme is both selectively and universally convertible; the signature length of our scheme is as short as BLS signature [2]; meanwhile, its security is under some conventional assumptions.

Organization of the Paper. In the next section, we will review some preliminaries required throughout the paper. The definition and security models of convertible undeniable signature are proposed in Section 3. We describe our convertible undeniable signature scheme together with its security analysis in Section 4. Finally, Section 5 concludes this paper.

2 Bilinear Pairings and Complexity Assumptions

Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two groups of prime order p and let g be a generator of $\mathbb{G}_1$. The map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is said to be an admissible bilinear pairing if the following three conditions hold true:

- e is bilinear, i.e. $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
- e is non-degenerate, i.e. $e(g, g) \neq 1_{\mathbb{G}_T}$.
- e is efficiently computable.

We say that $(\mathbb{G}_1, \mathbb{G}_T)$ are bilinear groups if there exists the bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ as above, and e, and the group action in $\mathbb{G}_1$ and $\mathbb{G}_T$ can be computed efficiently. For more details on the construction of such pairings, we refer the reader to [2].

The following three problems are assumed to be hard for any polynomial time algorithm.

Discrete Logarithm Problem: Given $(g, g^a) \in \mathbb{G}_1$, find a.

Computational Diffie-Hellman Problem: Given a triple $\mathbb{G}_1$ elements (g, g^a, g^b), find the element $C = g^{ab}$.

3-Decisional Diffie-Hellman (3-DDH) Problem in $\mathbb{G}_1$[2]: Given $(g, g^a, g^b, g^c, h) \in \mathbb{G}_1^5$, decide whether $h \stackrel{?}{=} g^{abc}$.

3 Definitions and Security Models of Convertible Undeniable Signature

In this section, we first define the outline of a convertible undeniable signature scheme. Then, we define the security properties of convertible undeniable signatures.

3.1 Outline of Convertible Undeniable Signature

The convertible undeniable signature scheme consists of the following algorithms:

Common Parameter Generation: a probabilistic algorithm that on input a security parameter k, outputs a string cp which denotes the common scheme

[2] The 3-DDH problem is also called xyz-DDH problem by other researchers in [24,27]. The traditional DDH problem is not hard in the bilinear pairing setting. Considering the 3-DDH problem and assuming its difficulty (together with the ease of the DDH problem), we are able to design cryptographic protocols achieving a trade-off between authenticity and privacy.

parameters including the message space $\mathcal{M}$ and the signature space $\mathcal{S}$. cp is shared among all the users in the system.

Key Generation: a probabilistic algorithm that on input a common parameter cp, outputs a secret/public key-pair (sk, pk) for a user in the system.

Undeniable Sign: a probabilistic (or deterministic) algorithm that on input the common parameter cp, signer S's secret/public key-pair (sk_s, pk_s) and the message m, outputs S's convertible undeniable signature σ.

Undeniable Verify: a deterministic algorithm that on input the common parameter cp, S's secret key secret/public key-pair (sk_s, pk_s) and message-signature pair (m, σ), outputs 1 if it is a valid message-signature pair. Otherwise, outputs 0.

Confirmation Protocol: an interactive (or non-interactive) algorithm that on input the common parameter cp, S's secret key sk_s, (possibly) Verifier V's public key pk_v and message-signature pair (m, σ), outputs a *non-transferable* transcript $Trans$ which can convince V about the validity of σ.

Disavowal Protocol: an interactive (non-interactive) algorithm that on input the common parameter cp, S's secret key sk_s, (possibly) V's public key pk_v and message-signature pair (m, σ), outputs a *non-transferable* transcript $Trans$ that shows the invalidity of σ to V.

Selectively Convert: a probabilistic (or deterministic) algorithm that on input the common parameter cp, S's secret key sk_s and the message-signature pair (m, σ), outputs a selective proof $\Pi_{pk_s}^{(m,\sigma)}$ of the given message-signature pair.

Selectively Verify: a deterministic algorithm that on input the common parameter cp, S's public key pk_s, message-signature pair (m, σ) and the selective proof $\Pi_{pk_s}^{(m,\sigma)}$, outputs the verification decision $d \in \{Acc, Rej\}$.

Universally Convert: a deterministic algorithm that on input the common parameter cp and S's secret key sk_s, outputs the universal proof Π_{pk_s}.

Universally Verify: a deterministic algorithm that on input the common parameter cp, S's public key pk_s, *any* message-signature pair (m, σ) and the universal proof Π_{pk_s}, outputs the verification decision $d \in \{Acc, Rej\}$.

We allow the adversary to access the following oracles and submit their queries adaptively:

- Key Generation Oracle: On a key generation query for the i^{th} user, this oracle runs the **Key Generation** algorithm to generate a secret/public key pair (sk_i, pk_i) of this user and returns the public key pk_i to the adversary.
- Undeniable Sign Oracle: On an undeniable sign query (m, pk_s), this oracle runs the **Undeniable Sign** algorithm to generate the undeniable signature σ and returns it to the adversary.
- Verify Oracle: On a verify query (m, σ, pk_s) (and possibly pk_v), this oracle first runs the **Undeniable Verify** algorithm to decide whether (m, σ) is a valid message-signature pair under the public key pk_s and outputs the decision result $d \in \{0, 1\}$. According to the decision result d, the oracle

responds based on whether a passive attack or an active/concurrent attack is mounted.

1. Active/Concurrent attack: The Verify Oracle executes the confirmation or disavowal protocol with adversary (acting as a cheating verifier) depending on the verification result $d \in \{0, 1\}$.
2. Passive attack: The Verify Oracle returns a transcript of confirmation protocol if $d = 1$. Otherwise, the oracle returns a transcript of disavowal protocol.

– Selectively Convert Oracle: On a selectively convert query (m, σ, pk_s), this oracle runs the **Selectively Convert** algorithm to generate the selective proof $\Pi_{pk_s}^{(m,\sigma)}$ and returns it to the adversary.
– Universally Convert Oracle: On a universally convert query pk_s, this oracle runs **Universally Convert** algorithm to generate the universal proof Π_{pk_s} and returns it to the adversary.
– Corruption Oracle: On a corruption query pk, this oracle returns the corresponding secret key to the adversary.

We assume that when the adversary issues some queries related to the public key pk, pk is returned from the Key Generation Oracle. The security of the convertible undeniable signature will be defined using the game between these oracles and the adversary. We will focus on the case when the Confirmation/Disavowal protocol is non-interactive for the rest of the paper[3]. The security when confirmation/disavowal protocol is interactive can be defined analogously.

3.2 Completeness

Essentially, the completeness means that valid (invalid) signatures can always be proved valid (invalid). It can be described as following two cases:

1. If the **Undeniable Verify** algorithm outputs 1 for a message-signature pair (m, σ) and the secret/public key pair (sk_s, pk_s), then (i) (m, σ) can be confirmed by the **Confirmation** protocol. (ii) On input (m, σ) together with a valid selective proof $\Pi_{pk_s}^{(m,\sigma)}$, **Selectively Verify** algorithm outputs Acc. (iii) On input (m, σ) together with a valid universal proof Π_{pk_s}, **Universally Verify** algorithm outputs Acc.
2. If the **Undeniable Verify** algorithm outputs 0 for a message-signature pair (m, σ) and the secret/public key pair (sk_s, pk_s), then (i)(m, σ) can be denied by the **Disavowal** protocol. (ii) On input (m, σ) together with a valid selective proof $\Pi_{pk_s}^{(m,\sigma)}$, **Selectively Verify** algorithm outputs Rej. (iii) On input (m, σ) together with a valid universal proof Π_{pk_s}, **Universally Verify** algorithm outputs Rej.

[3] If the Confirmation/Disavowal protocol is non-interactive, there is no need to consider the active/concurrent attack [36].

3.3 Non-transferability

The Non-Transferability requires that the transcripts of the confirmation and disavowal protocol with the designated verifier V can only convince V the validity or invalidity of a message-signature pair (m, σ). No one else could be convinced by this $Trans$ even if V shares all his secret information (including his secret key) with this party. We review the definition of the Non-Transferability given by Monnerat and Vaudenay in [35]:

The Confirmation/Disavowal protocol protocol is *non-transferable* if there exists a probabilistic polynomial time algorithm $\mathcal{A}$ with input the verifier V's secret key sk_v such that for any other computationally unbounded algorithm, any pair (m, σ), the transcript of the Confirmation/Disavowal generated by $\mathcal{A}$ is indistinguishable from that generated by the signer S.

3.4 Unforgeability

The standard notion of the security for digital signatures was defined by Goldwasser, Micali and Rivest [11], the existential unforgeability of the convertible undeniable signature scheme is defined similarly: adversary has access to the Key Generation Oracle, Undeniable Sign Oracle, Verify Oracle. In addition, we allow the adversary to submit queries to Selectively Convert Oracle and Universally Convert Oracle adaptively. This is to ensure that the knowledge of the selective or universal proof does not help the adversary to forge a new valid message signature pair. Furthermore, we also allow adversaries to access the Corruption Oracle to simulate the fact that the adversary might corrupt some users in the system. The unforgeability of the convertible undeniable signature is defined by the game between the oracles in Section 3.1 and an adaptively chosen message and chosen public key forger $\mathcal{F}$. We say $\mathcal{F}$ wins if $\mathcal{F}$ outputs a valid message-signature pair (m^*, σ^*) under the public keys pk_s^* with the restrictions that:

1. (m^*, pk_s^*) has never been submitted to the Undeniable Sign Oracle.
2. pk_s^* has never been submitted to the Corruption Oracle.

The success probability of an adaptively chosen message and chosen public key forger $\mathcal{F}$ wins the above game is defined as $\mathsf{Succ}\ \mathcal{F}_{EUF,\ CUS}^{CMA,\ CPKA}$.

Definition 1. *We say a convertible undeniable signature scheme is unforgeable against a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *forger* $\mathcal{F}_{EUF,\ CUS}^{CMA,\ CPKA}$, *if* $\mathcal{F}_{EUF,\ CUS}^{CMA,\ CPKA}$ *runs in time at most* t, *makes at most* q_{KG} *queries to* Key Generation Oracle, q_{US} *queries to* Undeniable Sign Oracle, q_V *queries to* Verify Oracle, q_{SC} *queries to the* Selectively Convert Oracle, q_{UC} *queries to the* Universally Convert Oracle, q_C *queries to the* Corruption Oracle *and* $\mathsf{Succ}\ \mathcal{F}_{EUF,\ CUS}^{CMA,\ CPKA}$ *is negligible.*

3.5 Invisibility

The property invisibility was first introduced into the undeniable signature by Chaum, van Heijst and Pfitzmann [6]. We will extend the this notion into the

setting of convertible undeniable signatures. That is, given a message-signature pair (m, σ) and the public key pk_s of the signer S, it should be difficult to decide whether it is a valid message-signature pair without the help of the signer, the selective proof $\Pi_{pk_s}^{(m,\sigma)}$ or universal proof Π_{pk_s}. It is defined using the game between the oracles in Section 3.1 and an adaptively chosen message attacker and chosen public key distinguisher $\mathcal{D}_{INV,\ CUS}^{CMA,\ CPKA}$. The whole game is divided into two phases.

- Phase 1: In this phase, the distinguisher $\mathcal{D}$ can adaptively access all the Oracles.
- Challenge: When the distinguisher $\mathcal{D}$ decides the first phase is over, he submits (m^*, pk_s^*) to Undeniable Sign Oracle as the challenge with the constraints that
  1. pk_s^* has not been submitted to the Corruption Oracle or Universally Convert Oracle during Phase 1.
  2. (m^*, pk_s^*) has not been submitted to the Undeniable Sign Oracle during Phase 1.

  As a response, the Undeniable Sign Oracle chooses a random bit $\gamma \in \{0, 1\}$. If $\gamma = 1$, this oracle will run **Undeniable Sign** algorithm to generate the undeniable signature σ and sets $\sigma^* = \sigma$. Otherwise, this oracle chooses a random element σ^* in the signature space $\mathcal{S}$. Then, it returns the challenging signature σ^* to $\mathcal{D}$.
- Phase 2: On receiving the challenging signature, the distinguisher $\mathcal{D}$ can still access all the oracles adaptively except that:
  1. pk_s^* cannot been submitted to the Corruption Oracle or Universally Convert Oracle.
  2. (m^*, pk_s^*) cannot been submitted to the Undeniable Sign Oracle.
  3. (m^*, σ^*, pk_s^*) cannot be submitted to the Selectively Convert Oracle or Verify Oracle.
- Guessing: Finally, the distinguisher $\mathcal{D}$ outputs a guess γ'. The adversary wins the game if $\gamma = \gamma'$.

Remark: If the selective proof of a message-signature pair (m^*, σ) only depends on m^*, then (m^*, pk_s^*) cannot be submitted to the Selectively Convert Oracle during Phase 1 and 2.

The advantage of the distinguisher $\mathcal{D}_{INV,\ CUS}^{CMA,\ CPKA}$ has in the above game is defined as $\mathsf{Adv}\ \mathcal{D}_{INV,\ CUS}^{CMA,\ CPKA} = |\Pr[\gamma = \gamma'] - 1/2|$.

Definition 2. *We say a convertible undeniable signature scheme is invisible against a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *distinguisher* $\mathcal{D}_{INV,\ CUS}^{CMA,\ CPKA}$*, if the distinguisher runs in time at most* t*, makes at most* q_{KG} *queries to* Key Generation Oracle*,* q_{US} *queries to* Undeniable Sign Oracle*,* q_V *queries to* Verify Oracle*,* q_{SC} *queries to the* Selectively Convert Oracle*,* q_{UC} *queries to the* Universally Convert Oracle*,* q_C *queries to the* Corruption Oracle *and* $\mathsf{Adv}\ \mathcal{D}_{INV,\ CUS}^{CMA,\ CPKA}$ *is negligible.*

3.6 Anonymity

Essentially, the anonymity property requires that given a valid message-signature pair (m, σ) and two possible signers' public keys pk_0, pk_1, it is computational impossible to decide who generated this signature. This property was introduced to the undeniable signature by Galbraith and Mao [13]. The authors suggested that anonymity is the most relevant security property for undeniable signature. In convertible undeniable signatures, this property is defined using the game between the oracles in Section 3.1 and an adaptively chosen message attacker and chosen public key distinguisher $\mathcal{D}^{CMA,\ CPKA}_{ANONY,\ CUS}$. Similarly to the models defined in Section 3.5, the whole game is divided into two phases.

- Phase 1: In this phase, the distinguisher $\mathcal{D}$ can adaptively access to all the Oracles.
- Challenge: When the distinguisher $\mathcal{D}$ decides the first phase is over, he submits (m^*, pk_0^*, pk_1^*) to Undeniable Sign Oracle as the challenge with the constraints that
  1. Neither pk_0^* nor pk_1^* has been submitted to the Corruption Oracle or Universally Convert Oracle during Phase 1.
  2. Neither (m^*, pk_0^*) nor (m^*, pk_1^*) has been submitted to the Undeniable Sign Oracle during Phase 1.

  As a response, the Undeniable Sign Oracle chooses a random bit $\gamma \in \{0, 1\}$. If $\gamma = 0$, this oracle will run **Undeniable Sign** algorithm to generate the undeniable signature σ under the secret key sk_0^*. Otherwise, it will run **Undeniable Sign** algorithm to generate the undeniable signature σ under the secret key sk_1^*. Then, it returns the challenging signature to $\mathcal{D}$.
- Phase 2: On receiving the challenging signature, the distinguisher $\mathcal{D}$ can still access all the oracles adaptively except that:
  1. Neither pk_0^* nor pk_1^* can be submitted to the Corruption Oracle or Universally Convert Oracle.
  2. Neither (m^*, pk_0^*) nor (m^*, pk_1^*) can be submitted to the Undeniable Sign Oracle.
  3. Neither (m^*, σ^*, pk_0^*) nor (m^*, σ^*, pk_1^*) can be submitted to the Selectively Convert Oracle or Verify Oracle.
- Guessing: Finally, the distinguisher $\mathcal{D}$ outputs a guess γ'. The adversary wins the game if $\gamma = \gamma'$.

Remark: If the selective proof of a message-signature pair (m^*, σ) only depends on m^*, then neither (m^*, pk_0^*) nor (m^*, pk_1^*) can be submitted to the Selectively Convert Oracle during Phase 1 and 2.

The advantage of the distinguisher $\mathcal{D}^{CMA,\ CPKA}_{ANONY,\ CUS}$ has in the above game is defined as $\mathsf{Adv}\ \mathcal{D}^{CMA,\ CPKA}_{ANONY,\ CUS} = |\Pr[\gamma' = \gamma] - 1/2|$.

Definition 3. *We say a convertible undeniable signature scheme is anonymous against a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *distinguisher* $\mathcal{D}^{CMA,\ CPKA}_{ANONY,\ CUS}$*, if the distinguisher runs in time at most* t*, makes at most* q_{KG} *queries to* Key Generation Oracle*,* q_{US} *queries to* Undeniable Sign Oracle*,* q_V *queries to* Verify Oracle*,* q_{SC} *queries to the* Selectively Convert Oracle*,* q_{UC} *queries to the* Universally Convert Oracle*,* q_C *queries to the* Corruption Oracle *and* $\mathsf{Adv}\ \mathcal{D}^{CMA,\ CPKA}_{ANONY,\ CUS}$ *is negligible.*

Equivalence Between the Anonymity and Invisibility in the Convertible Undeniable Signature

In [13], Galbraith and Mao showed that the property anonymity and invisibility are closely related in undeniable signature. Therefore, a natural question is whether these two properties have the same close relationship in convertible undeniable signature. Note that we cannot directly obtain this result according to the proof given by Galbraith and Mao [13], due to the reason that the distinguishers defined in the convertible undeniable signature are stronger than those defined in [13]. As we shall demonstrate later, these two properties are equivalent.

Theorem 1. *The properties anonymity and invisibility in the convertible undeniable signature are equivalent.*

Proof. Our proof draws inspiration from the techniques in [13]. The proof of this theorem consists of the following two lemmas:

Lemma 1. *If a convertible undeniable signature scheme is invisible in the sense of Definition 2, then it is also anonymous in the sense of Definition 3.*

Proof. Suppose there exists an adversary $\mathcal{D}_A$ who has non negligible advantage $\mathsf{Adv}\ \mathcal{D}^{CMA,\ CPKA}_{ANONY,\ CUS} = \varepsilon$ in the game of anonymity defined in Section 3.6, then we will show there is an adversary $\mathcal{D}_I$ who can use $\mathcal{D}_A$ to have the advantage $\varepsilon/2$ in the game of invisibility defined in Section 3.5 and thus this scheme is not invisible.

In the proof, $\mathcal{D}_I$ will forward $\mathcal{D}_A$'s queries to his own oracles and therefore $\mathcal{D}_I$ will not abort during the simulation. When $\mathcal{D}_A$ outputs (m^*, pk_0^*, pk_1^*) as the challenge, $\mathcal{D}_I$ will choose a random number $\gamma" \in \{0,1\}$ and output $(m^*, pk^*_{\gamma"})$ as his own challenge. As defined in the model of Invisibility, $\mathcal{D}_I$ will receive a signature σ^* such that it is a valid signature under the public key $pk^*_{\gamma"}$ if $\gamma = 1$ and is not a valid signature if $\gamma = 0$. Then $\mathcal{D}_I$ returns σ^* to $\mathcal{D}_A$. Now $\mathcal{D}_A$ can continue to submit his queries and $\mathcal{D}_I$ can answer those queries as before. At last, $\mathcal{D}_A$ outputs his guess γ'. If $\gamma' = \gamma"$, $\mathcal{D}_I$ will output 1, otherwise $\gamma' \neq \gamma"$, $\mathcal{D}_I$ will output 0. Therefore the probability that $\mathcal{D}_I$ can output a valid guess is $\Pr[\gamma' = \gamma"|\gamma = 1]\Pr[\gamma = 1] + \Pr[\gamma' \neq \gamma"|\gamma = 0]\Pr[\gamma = 0]$. Note that if $\gamma = 1$, that is σ^* is a valid signature under the public key $pk^*_{\gamma"}$, then $\mathcal{D}_A$ can output $\gamma' = \gamma"$ with probability $1/2 + \varepsilon$. If $\gamma = 0$, then $\Pr[\gamma' \neq \gamma"|\gamma = 0] = 1/2$ due to the random choice of $\gamma"$. Therefore, the advantage that $\mathcal{D}_I$ has is: $\mathsf{Adv}\ \mathcal{D}^{CMA,\ CPKA}_{INV,\ CUS} = (\frac{1}{2} + \varepsilon)\frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{1}{2}\varepsilon = \frac{1}{2}\mathsf{Adv}\ \mathcal{D}^{CMA,\ CPKA}_{ANONY,\ CUS}$. □

Lemma 2. *If a convertible undeniable signature scheme is anonymous in the sense of Definition 3, then it is also invisible in the sense of Definition 2.*

Proof. Suppose there exists a distinguisher $\mathcal{D}_I$ who has non negligible advantage $\mathsf{Adv}\ \mathcal{D}^{CMA,\ CPKA}_{INV,\ CUS} = \varepsilon$ in the game of invisibility defined in Section 3.5, then we will show there is a distinguisher $\mathcal{D}_A$ who can use $\mathcal{D}_I$ to has almost the same advantage in the game of anonymity defined in Section 3.6 and thus this scheme is not anonymous.

In our proof, $\mathcal{D}_A$ will answer all queries from $\mathcal{D}_I$. To do this, $\mathcal{D}_A$ will forward $\mathcal{D}_I$'s queries to his own oracles and return the responses to $\mathcal{D}_I$. Additionally, $\mathcal{D}_A$ will submit one more Key Generation query to his Key Generation Oracle and obtain the response $\widetilde{pk_s^*}$ (Note that this public key will not be given to $\mathcal{D}_I$). When $\mathcal{D}_I$ outputs (m^*, pk_s^*) as the challenge, $\mathcal{D}_A$ will submit $(m^*, \widetilde{pk_s^*}, pk_s^*)$ as his own challenge. As defined in the model of Anonymous, $\mathcal{D}_A$ will receive a challenging signature σ^* such that it is a valid signature under the public key $\widetilde{pk_s^*}$ if $\gamma = 0$. Otherwise, it is a valid signature under the public key pk_s^*. Then $\mathcal{D}_A$ returns it to $\mathcal{D}_I$. Now $\mathcal{D}_I$ can continue to submit his queries and $\mathcal{D}_A$ can answer those queries as before. At last, $\mathcal{D}_I$ outputs his guess γ' and $\mathcal{D}_A$ also forwards γ' as his answer. Note that if σ^* is a valid signature under the public key $\widetilde{pk_s^*}$, that is $\gamma = 0$, then with negligible probability ε', it is also valid under the public key pk_s^*. So $\mathcal{D}_I$ will output $\gamma' = 0$ with probability $1/2 + \varepsilon$. Similarly, if σ^* is a valid signature under the public key pk_s^*, that is $\gamma = 1$, then with probability $1/2 + \varepsilon$, γ' will be 1. Therefore, the advantage that $\mathcal{D}_A$ has is $\mathsf{Adv}\ \mathcal{D}^{CMA,\ CPKA}_{ANOY,\ CUS} \geq (\frac{1}{2} + \varepsilon)(1 - \varepsilon') - \frac{1}{2} = \varepsilon - \varepsilon'(\frac{1}{2} + \varepsilon) \geq \mathsf{Adv}\ \mathcal{D}^{CMA,\ CPKA}_{INV,\ CUS} - \varepsilon'$ where ε' is negligible. □

3.7 Soundness

Soundness requires that even the signer himself cannot convince a verifier V that a valid (invalid) signature is invalid (valid) without corrupting $V's$ secret key. This property has been regarded as one of the necessary properties of the **Confirmation**/**Disavowal** protocol [23,31,35]. It is defined by the game against an adversary $\widetilde{S}$ where $\widetilde{S}$ can adaptively access all the oracles in Section 3.1. After all the queries, we say $\widetilde{S}$ wins the game if $\widetilde{S}$ can output $(m^*, \sigma^*, Trans^*, pk_s, pk_v)$ with the restrictions that

1. pk_v has not been submitted to the Corruption Oracle.
2. σ^* is not a valid undeniable signature of message m^* under the public key pk_s and $Trans^*$ is a transcripts output by **Confirmation** protocol. Or, σ^* is a valid undeniable signature of message m^* under the public key pk_s and $Trans^*$ is a transcript output by **Disavowal** protocol.

The success probability of an adaptively chosen message and chosen public key adversary $\widetilde{S}$ wins the above game is defined as $\mathsf{Succ}\ \widetilde{S}^{CMA,\ CPKA}_{Sound,\ CUS}$.

Definition 4. *We say a convertible undeniable signature scheme satisfies the property of soundness against a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *adversary* $\widetilde{S}^{CMA,\ CPKA}_{Sound,\ CUS}$, *if* $\widetilde{S}^{CMA,\ CPKA}_{Sound,\ CUS}$ *runs in time at most* t, *makes at most* q_{KG} *queries to* Key Generation Oracle, q_{US} *queries to* Undeniable Sign Oracle, q_V *queries to* Verify Oracle, q_{SC} *queries to the* Selectively Convert Oracle, q_{UC} *to the* Universally Convert Oracle, q_C *queries to the* Corruption Oracle *and* $\mathsf{Succ}\ \widetilde{S}^{CMA,\ CPKA}_{Sound,\ CUS}$ *is negligible.*

3.8 Non-impersonation

The notion Non-Impersonation was first introduced into the undeniable signature by Kurosawa1 and Heng [22]. Basically, the notion Non-Impersonation in the convertible undeniable signature requires that:

1. Only with the knowledge of the public key of the signer S, it should be difficult for an impersonator $\mathcal{I}$ to generate the selective proof for a message-signature pair.
2. Only with the knowledge of the public key of the signer S, it should be difficult for an impersonator $\mathcal{I}$ to generate the universal proof.
3. Only with the knowledge of the public keys of the signer S and the verifier V, it should be difficult for an impersonator $\mathcal{I}$ to generate a transcript of the **Confirmation** protocol or **Disavowal** protocol.

It is defined using the game between the oracles in Section 3.1 and an adaptively chosen message attacker and chosen public key impersonator $\mathcal{I}_{CUS}^{CMA,\ CPKA}$.

1. CASE_1: **Impersonation of Selectively Convert Algorithm**
   In this case, the impersonator $\mathcal{I}$ can adaptively access all the Oracles. After all the queries, $\mathcal{I}$ outputs a valid selective proof $\Pi_{pk_s^*}^{(m^*,\sigma^*)}$ with the restrictions that (m^*, σ^*, pk_s^*) has not been submitted to the Selectively Convert Oracle and pk_s^* has not been submitted to the Corruption Oracle. Similarly in the definition of **Invisibility**, if the selective proof of a message-signature pair (m^*, σ) only depends on m^*, then (m^*, pk_s^*) cannot be submitted to the Selectively Convert Oracle.

   The success probability of an adaptively chosen message and chosen public key impersonator $\mathcal{I}$ wins the above game is defined as $\mathsf{Succ}\ \mathcal{I}_{CASE_1,\ CUS}^{CMA,\ CPKA}$.
2. CASE_2: **Impersonation of Universally Convert algorithm**
   In this case, the impersonator $\mathcal{I}$ can adaptively access all the Oracles. After all the queries, $\mathcal{I}$ outputs a universal proof $\Pi_{pk_s^*}$ with the restriction that pk_s^* has not been submitted to the Universally Convert Oracle or Corruption Oracle.

   The success probability of an adaptively chosen message and chosen public key adversary $\mathcal{I}$ wins the above game is defined as $\mathsf{Succ}\ \mathcal{I}_{CASE_2,\ CUS}^{CMA,\ CPKA}$.
3. CASE_3: **Impersonation of Confirmation/Disavowal protocol**
   In this case, the impersonator $\mathcal{I}$ can adaptively access all the Oracles. After all the queries, $\mathcal{I}$ can output $(m^*, \sigma^*, Trans^*, pk_s, pk_v)$ such that $Trans^*$ can confirm or deny the undeniable signature σ^*. The only restrictions are that neither pk_s^* nor pk_v^* has been submitted to the Corruption Oracle and $(m^*, \sigma^*, pk_s, pk_v)$ cannot be submitted to the Verify Oracle.

   The success probability of an adaptively chosen message and chosen public key impersonator $\mathcal{I}$ wins the above game is defined as $\mathsf{Succ}\ \mathcal{I}_{CASE_3,\ CUS}^{CMA,\ CPKA}$.

Definition 5. *We say that a convertible undeniable signature scheme is non-impersonated against a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *adversary* $\mathcal{I}_{CUS}^{CMA,\ CPKA}$, *if* $\mathcal{I}_{CUS}^{CMA,\ CPKA}$ *runs in time at most* t, *makes at most* q_{KG} *queries to* Key Generation Oracle, q_{US} *queries to* Undeniable Sign Oracle, q_V *queries to* Verify Oracle,

q_{SC} queries to the Selectively Convert Oracle*, q_{UC} to the* Universally Convert Oracle*, q_C queries to the* Corruption Oracle *and* Succ $\mathcal{I}^{CMA,\ CPKA}_{CASE_i,\ CUS}(i = 1, 2, 3)$ *is negligible.*

4 Proposed Scheme

In this section, we will describe our convertible undeniable signature scheme with security and efficiency analysis.

4.1 Concrete Scheme

Our new scheme consists of the following algorithms:

Common Parameter Generation: Let $(\mathbb{G}_1, \mathbb{G}_T)$ be bilinear groups where $|\mathbb{G}_1| = |\mathbb{G}_T| = p$, for some prime number $p \geq 2^k$, k be the system security number and g be the generator of $\mathbb{G}_1$. e denotes the bilinear pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Let $h_0, h_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*, h_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ be three secure cryptographic hash functions.

Key Generation: The signer S picks two secret values $x_s, y_s \in_R \mathbb{Z}_p^*$ and sets the secret key $sk_s = (x_s, y_s)$. Then S computes the public key $pk_s = (X_s, Y_s) = (g^{x_s}, g^{y_s})$. Similarly, verifier V's secret/public key-pair is $(sk_v, pk_v) = (x_v, y_v, X_v, Y_v)$ where x_v, y_v are randomly chosen in $\mathbb{Z}_p^*$.

Undeniable Sign: For a message m to be signed, S uses his secret key (x_s, y_s) to compute the convertible undeniable signature $\sigma = h_0(m)^{x_s y_s} \cdot h_1(m)^{y_s}$.

Undeniable Verify: For a message-signature pair (m, σ), S checks whether $\sigma \stackrel{?}{=} h_0(m)^{x_s y_s} \cdot h_1(m)^{y_s}$. If the equality holds, S will accept it as a valid undeniable signature and output 1. Otherwise, outputs 0.

Confirmation Protocol: Given the verifier V's public key $pk_v = (X_v, Y_v)$ and a valid message-signature pair (m, σ) to be confirmed, the signer S will use the designated verifier techniques [16] to prove its validity.

- Signer S chooses $c_v, d_v \in_R \mathbb{Z}_p, r \in_R \mathbb{Z}_p^*$ and computes:
  1. $A = g^r, B = e(h_0(m), X_s)^r, C = g^{d_v} Y_v^{c_v}$,
  2. $h = h_2(m \| \sigma \| pk_v \| A \| B \| C)$, $c_s = h - c_v \pmod p$, and $d_s = r - c_s y_s \pmod p$.

  S then sends (c_s, c_v, d_s, d_v) to the verifier V.
- On receiving (c_s, c_v, d_s, d_v) and the message-signature pair (m, σ), the verifier V
  1. computes $W = e(\sigma, g)/e(h_1(m), Y_s)$,
  2. computes $A' = g^{d_s} Y_s^{c_s}$, $B' = e(h_0(m), X_s)^{d_s} W^{c_s}$, $C' = g^{d_v} Y_v^{c_v}$,
  3. checks whether $c_s + c_v \stackrel{?}{=} h_2(m \| \sigma \| pk_v \| A' \| B' \| C')$.

  If the equality holds, V will accept σ as a valid undeniable signature of message m.

Disavowal Protocol: Given the verifier V's public key $pk_v = (X_v, Y_v)$ and a message-signature pair (m, σ) to be denied, the signer S will use the designated verifier techniques [16] to deny its validity.

– Signer S chooses $c_v, d_v, \alpha, \beta \in_R \mathbb{Z}_p, r \in \mathbb{Z}_p^*$ and computes:
  1. $W = e(\sigma, g)/e(h_1(m), Y_s)$,
  2. $A = [e(h_0(m), X_s)^{y_s}/W]^r \neq 1, B = e(h_0(m), X_s)^{\alpha}/W^{\beta}, C = g^{\alpha}/Y_s^{\beta}$, $D = g^{d_v} Y_v^{c_v}$,
  3. $h = h_2(m\|\sigma\|pk_v\|A\|B\|C\|D)$ and $c_s = h - c_v \pmod p$,
  4. $d_s = \alpha + c_s y_s r \pmod p$ and $\widehat{d_s} = \beta + c_s r \pmod p$.

  S then sends $(A, c_s, c_v, d_s, \widehat{d_s}, d_v)$ to verifier V.
– On receiving $(A, c_s, c_v, d_s, \widehat{d_s}, d_v)$ and the message-signature pair (m, σ), the Verifier V
  1. computes $W = e(\sigma, g)/e(h_1(m), Y_s)$, $B' = e(h_0(m), X_s)^{d_s}/(W^{\widehat{d_s}} \cdot A^{c_s})$, $C' = g^{d_s}/Y_s^{\widehat{d_s}}$ and $D' = g^{d_v} Y_v^{c_v}$,
  2. checks whether $A \neq 1$ and $c_s + c_v \stackrel{?}{=} h_2(m\|\sigma\|pk_v\|A\|B'\|C'\|D')$

  If $A \neq 1$ and $c_s + c_v = h_2(m\|\sigma\|pk_v\|A\|B'\|C'\|D')$, V will believe that σ is not a valid undeniable signature.

Selectively Convert: When S wants to make his undeniable message-signature pair (m, σ) publicly verifiable, he computes the selective proof as $\Pi_{pk_s}^{(m,\sigma)} = h_0(m)^{y_s}$.

Selectively Verify: For a message-signature pair (m, σ) and its selective proof $\Pi_{pk_s}^{(m,\sigma)}$,

1. anyone can verify whether $e(\Pi_{pk_s}^{(m,\sigma)}, g) \stackrel{?}{=} e(h_0(m), Y_s)$. If this equality holds, go to next step. Otherwise, $\Pi_{pk_s}^{(m,\sigma)}$ is invalid.
2. compute $W = e(\sigma, g)/e(h_1(m), Y_s)$ and check whether $W \stackrel{?}{=} e(\Pi_{pk_s}^{(m,\sigma)}, X_s)$. If this equality holds as well, one can accept σ as a valid undeniable signature. Otherwise, it is invalid.

Universally Convert: When S wants to make all his undeniable signatures publicly verifiable, he computes $\Pi_{pk_s} = g^{x_s y_s}$ and publishes Π_{pk_s}.

Universally Verify: For any message-signature pair (m, σ) and the universal proof Π_{pk_s},

1. anyone can verify whether $e(g, \Pi_{pk_s}) \stackrel{?}{=} e(X_s, Y_s)$. If this equality holds, go to next step. Otherwise, Π_{pk_s} is invalid.
2. compute $W = e(\sigma, g)/e(h_1(m), Y_s)$ and check $W \stackrel{?}{=} e(h_0(m), \Pi_{pk_s})$. If this equality holds as well, one can accept σ as a valid undeniable signature. Otherwise, it is invalid.

4.2 Security Analysis of the Proposed Scheme

In this section, we will give a formal security analysis of our proposed scheme.

Theorem 2. *The proposed scheme satisfies the property* **Completeness** *and* **Non-Transferability**.

Proof. The completeness of our proposed is clearly satisfied. The Confirmation and Disavowal protocols in our scheme are non-transferable because the designated verifiers techniques [16] serve as the building blocks. Due to page limitation, the detail of the proof will be presented in the full version of this paper.

Theorem 3. *If there exists a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *forger* $\mathcal{F}_{EUF,\ CUS}^{CMA,\ CPKA}$ *who can submit additional* q_H *queries to random oracles and win the game of* ***Unforgeability*** *defined in Section 3.4 with non-negligible success probability* $\mathsf{Succ}\ \mathcal{F}_{EUF,\ CUS}^{CMA,\ CPKA}$*, then there exists an algorithm* $\mathcal{A}$ *who can use* $\mathcal{F}$ *to solve a random instance of Computational Diffie-Hellman problem with probability* $Succ_{\mathcal{A},\mathbb{G}_1}^{CDH} \geq \frac{1}{q_{KG}(q_{US}+1)}(1-\frac{1}{q_{US}+1})^{q_{US}}(1-\frac{1}{q_{KG}})^{q_C}\mathsf{Succ}\ \mathcal{F}_{EUF,CUS}^{CMA,CPKA}$ *in polynomial time.*

Proof. Suppose there exists an adaptively chosen message and chosen public key forger $\mathcal{F}$ who can forge a valid signature of our proposed scheme, then there exists an algorithm $\mathcal{A}$ who can solve the Computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$ by running $\mathcal{F}$ as subroutine.

Let (g, g^a, g^b) be a random instance of the CDH problem, $\mathcal{A}$ will simulate all the oracles and answer $\mathcal{F}$'s queries as follows. At the beginning of Phase 1, $\mathcal{A}$ will choose a random number $\pi \in \{1, 2, \cdots, q_{KG}\}$.

- Random Oracles: In order to respond $\mathcal{F}$'s queries to random oracles, $\mathcal{A}$ will maintain three lists: h_0-*list*, h_1-*list* and h_2-*list*.
  1. h_0 queries: At any time, $\mathcal{F}$ can issue an h_0 query for the message m_i. In response, $\mathcal{A}$ will maintain an h_0-*list* which stores his responses to such queries. For a new query, $\mathcal{A}$ chooses a number $\mu_i \in \{0, 1\}$ such that $\Pr[\mu_i = 1] = \delta$ (the value of δ will be determined later). If $\mu_i = 1$, $\mathcal{A}$ will choose a random number $\nu_i \in \mathbb{Z}_p^*$ and set $h_0(m_i) = (g^a)^{\nu_i}$. Otherwise, $\mu_i = 0$ and $\mathcal{A}$ sets $h_0(m_i) = g^{\nu_i}$. Then $\mathcal{A}$ adds $(m_i, h_0(m_i), \mu_i, \nu_i)$ into h_0-*list* and returns $h_0(m_i)$ as the answer.
  2. h_1 queries: At any time, $\mathcal{F}$ can issue an h_1 query for the message m_i. In response, $\mathcal{A}$ will maintain an h_1-*list* which stores his responses to such queries. For a new query, $\mathcal{A}$ chooses a random number $\eta_i \in \mathbb{Z}_p^*$ and sets $h_1(m_i) = g^{\eta_i}$. Then $\mathcal{A}$ adds $(m_i, h_1(m_i), \eta_i)$ into h_1-*list* and returns $h_1(m_i)$ as the answer.
  3. h_2 queries: At any time, $\mathcal{F}$ can issue an h_2 query for the input ξ_i. In response, $\mathcal{A}$ will maintain an h_2-*list* which stores his responses to such queries. For a new query, $\mathcal{A}$ chooses a random number $h_i \in \mathbb{Z}_p$ and sets $h_2(\xi_i) = h_i$. Then $\mathcal{A}$ adds $(\xi_i, h_2(\xi_i))$ into h_2-*list* and returns $h_2(\xi_i)$ as the answer.
- Key Generation Oracle: At any time, $\mathcal{F}$ can require the i^{th} user's public key $pk_i = (X_i, Y_i)$. In response, $\mathcal{A}$ will maintain a *pk-list* which stores his responses to such queries. For a new query, $\mathcal{A}$ chooses two random numbers $x_i, y_i \in \mathbb{Z}_p^*$. If $i = \pi$, $\mathcal{A}$ will set $X_\pi = g^b \cdot g^{x_\pi}, Y_\pi = g^{y_\pi}$. Otherwise, $\mathcal{A}$ sets $X_i = g^{x_i}, Y_i = g^{y_i}$. Then $\mathcal{A}$ adds (pk_i, x_i, y_i) into *pk-list* and returns pk_i as the answer.
- Undeniable Sign Oracle: At any time, $\mathcal{F}$ can submit an undeniable sign query (m_i, pk_j). We assume that m_i has been submitted to the h_0 oracle and h_1 oracle. Therefore, two tuples $(m_i, h_0(m_i), \mu_i, \nu_i)$ and $(m_i, h_1(m_i), \eta_i)$ have been in h_0-*list* and h_1-*list* respectively. Otherwise, $\mathcal{A}$ runs above algorithms for responding to random oracles and generates these two tuples. Similarly,

pk_j is returned from Key Generation Oracle and the tuple (pk_j, x_j, y_j) exists in *pk-list*.

1. If $j \neq \pi$, then $X_j = g^{x_j}, Y_j = g^{y_j}$ which means $\mathcal{A}$ knows the signer's secret key $sk_j = (x_j, y_j)$. Therefore, $\mathcal{A}$ runs the **Undeniable Sign** algorithm defined in Section 4.1 and computes the undeniable signature as $\sigma_i = h_0(m_i)^{x_j y_j} \cdot h_1(m_i)^{y_j}$.
2. Else, if $j = \pi$ and $\mu_i = 0$, then $X_\pi = g^b \cdot g^{x_\pi}, Y_\pi = g^{y_\pi}, h_0(m_i) = g^{\nu_i}$ which means $\mathcal{A}$ does not know the signer's secret key $sk_\pi = (b + x_\pi, y_\pi)$. In this case, the valid undeniable signature of this message should be $\sigma_i = h_0(m_i)^{(b+x_\pi)y_\pi} \cdot h_1(m_i)^{y_\pi} = (g^{\nu_i})^{(b+x_\pi)y_\pi} \cdot h_1(m_i)^{y_\pi} = (g^b)^{\nu_i y_\pi} \cdot h_0(m_i)^{x_\pi y_\pi} \cdot h_1(m_i)^{y_\pi}$. Therefore, $\mathcal{A}$ can compute the valid signature σ_i.
3. Otherwise, $\mathcal{A}$ fails to simulate the Undeniable Sign Oracle and aborts.

– Verify Oracle: At any time, $\mathcal{F}$ can submit an undeniable verify query $(m_i, \sigma_i, pk_s, pk_v)$. Since pk_s is returned from Key Generation Oracle, the tuple (pk_s, x_s, y_s) is in *pk-list*. Note that $\mathcal{A}$ knows the secret key y_s such that $Y_s = g^{y_s}$. Therefore, $\mathcal{A}$ can check whether $e(\sigma_i, g)/e(h_1(m_i), Y_s) \stackrel{?}{=} e(h_0(m_i), X_s)^{y_s}$. If this equality holds, $\mathcal{A}$ runs the **Confirmation** protocol as defined in Section 4.1 with the knowledge y_s. Otherwise, he runs **Disavowal** protocol with the knowledge y_s.
– Selectively Convert Oracle: At any time, $\mathcal{F}$ can submit the selectively convert query (m_i, pk_j)[4]. Similarly, $\mathcal{A}$ knows the value y_j such that $Y_j = g^{y_j}$. Therefore, $\mathcal{A}$ can compute the selective proof $\Pi_{(m_i, pk_j)} = h_0(m_i)^{y_j}$.
– Universally Convert Oracle: At any time, $\mathcal{F}$ can submit the universally convert query pk_i. Similarly, $\mathcal{A}$ knows the value y_i such that $Y_i = g^{y_i}$. Therefore, $\mathcal{A}$ can compute the selective proof $\Pi_{pk_i} = (X_i)^{y_i}$.
– Corruption Oracle: At any time, $\mathcal{F}$ can submit the corruption query pk_i. If $i \neq \pi$, then $X_j = g^{x_j}, Y_j = g^{y_j}$. Therefore $\mathcal{A}$ returns (x_i, y_i) to $\mathcal{F}$ as the answer. Otherwise, $\mathcal{A}$ fails to simulate the Corruption Oracle and aborts.

After all the queries, $\mathcal{F}$ will output a valid forgery (m^*, σ^*, pk_s^*) such that (m^*, pk_s^*) has never been submitted to the Undeniable Sign Oracle and pk_s^* has never been submitted to the Corruption Oracle. By assumption, $(m^*, h_0(m^*), \mu^*, \nu^*)$ has been in h_0-*list*, $(m^*, h_1(m^*), \eta^*)$ has been in h_1-*list* and (pk_s^*, x^*, y^*) has been in *pk-list* .

1. $\mu^* = 1$ and $pk_s^* = pk_\pi$, then $h_0(m^*) = (g^a)^{\nu^*}$ and $pk_s^* = pk_\pi = (X_\pi, Y_\pi) = (g^b \cdot g^{x_\pi}, g^{y_\pi})$. $\mathcal{A}$ can solve the CDH problem as follows. Since (m^*, σ^*) is a valid message-signature pair, $\sigma^* = h_0(m^*)^{(b+x_\pi)y_\pi} h_1(m^*)^{y_\pi}$ which means $\sigma^* = (g^a)^{(b+x_\pi)y_\pi \nu^*} h_1(m^*)^{y_\pi}$. Therefore $\mathcal{A}$ can obtain $\sigma^* = (g^{ab})^{y_\pi \nu^*} \cdot (g^a)^{x_\pi y_\pi \nu^*} \cdot h_1(m^*)^{y_\pi}$ and $g^{ab} = (\frac{\sigma^*}{(g^a)^{x_\pi y_\pi \nu^*} \cdot h_1(m^*)^{y_\pi}})^{(y_\pi \nu^*)^{-1}}$.
2. Otherwise, $\mathcal{A}$ fails to solve the CDH problem.

[4] Note that the forger does not need to provide the corresponding undeniable signature when he issues the selectively convert query. The reason is that the user pk_j's selective proof of the message m_i in our scheme is deterministic.

If $\mathcal{A}$ does not abort during the simulation, he can solve this random instance of CDH problem with probability $\frac{\delta}{q_{KG}} \cdot \mathsf{Succ}\ \mathcal{F}_{EUF,CUS}^{CMA,CPKA}$. In addition, all the simulations can be carried out in polynomial time.

Now we consider the probability that $\mathcal{A}$ dose not abort during the simulation. As we can see, $\mathcal{A}$ will not abort during the simulation if and only if:

1. $\mathcal{A}$ does not fail during the simulation of Undeniable Sign Oracle, which happens with probability $(1 - \frac{\delta}{q_{KG}})^{q_{US}}$.
2. $\mathcal{A}$ does not fail during the simulation of Corruption Oracle, which happens with probability $(1 - \frac{1}{q_{KG}})^{q_C}$.

The probability that $\mathcal{A}$ does not abort during the simulation is $(1 - \frac{\delta}{q_{KG}})^{q_{US}}(1 - \frac{1}{q_{KG}})^{q_C} \geq (1-\delta)^{q_{US}}(1 - \frac{1}{q_{KG}})^{q_C}$. Therefore, $\mathcal{A}$ can successfully solve the CDH problem with probability $Succ_{\mathcal{A},\mathbb{G}_1}^{CDH} \geq \frac{\delta}{q_{KG}}(1-\delta)^{q_{US}}(1-\frac{1}{q_{KG}})^{q_C}\ \mathsf{Succ}\ \mathcal{F}_{EUF,CUS}^{CMA,CPKA}$. When $\delta = 1/(q_{US}+1)$, this probability is maximized at $Succ_{\mathcal{A},\mathbb{G}_1}^{CDH} \geq \frac{1}{q_{KG}(q_{US}+1)}(1-\frac{1}{q_{US}+1})^{q_{US}}(1 - \frac{1}{q_{KG}})^{q_C}\ \mathsf{Succ}\ \mathcal{F}_{EUF,CUS}^{CMA,CPKA}$. □

Theorem 4. *If there exists a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *distinguisher* $\mathcal{D}_{INV,\ CUS}^{CMA,\ CPKA}$ *who can make additional* q_H *queries to random oracles and have non-negligible advantage* $\mathsf{Adv}\ \mathcal{D}_{INV,\ CUS}^{CMA,\ CPKA}$ *in the game of* ***Invisibility*** *defined in Section 3.5, then there exists an algorithm* $\mathcal{A}$ *who can use* $\mathcal{D}$ *to solve a random instance of 3-Decisional Diffie-Hellman problem with advantage* $\mathsf{Adv}\ A_{\mathbb{G}_1}^{3-DDH} \geq \frac{4}{q_{KG}(2q_{US}+q_{SC})^2}(1 - \frac{2}{2q_{US}+q_{SC}+2})^{2q_{US}+q_{SC}+2}(1 - \frac{1}{q_{KG}})^{q_{UC}+q_C}\ \mathsf{Adv}\ \mathcal{D}_{INV,CUS}^{CMA,CPKA}$ *in polynomial time.*

Proof. Suppose that there exists an adaptively chosen message and chosen public key distinguisher $\mathcal{D}$ who can win the game defined in Section 3.5, then we will show there exists an algorithm $\mathcal{A}$ who can solve the 3-Decisional Diffie-Hellman (3-DDH) problem in $\mathbb{G}_1$ by running $\mathcal{D}$ as subroutine.

Let (g, g^a, g^b, g^c, h) be a random instance of the 3-DDH problem, $\mathcal{A}$ will simulate all the oracles and answer $\mathcal{D}$'s queries as follows. At the beginning of Phase 1, $\mathcal{A}$ will choose a random number $\pi \in \{1, 2, \cdots, q_{KG}\}$.

– Random Oracles: In order to respond $\mathcal{D}$'s queries to random oracle, $\mathcal{A}$ will maintain three lists: h_0-*list*, h_1-*list* and h_2-*list*.
  1. h_0 queries: At any time, $\mathcal{D}$ can issue an h_0 query for the message m_i. In response, $\mathcal{A}$ will maintain an h_0-*list* which stores his responses to such queries. For a new query, $\mathcal{A}$ chooses a number $\mu_i \in \{0, 1\}$ such that $\Pr[\mu_i = 1] = \delta$ (the value of δ will be determined later). If $\mu_i = 1$, $\mathcal{A}$ will choose a random number $\nu_i \in \mathbb{Z}_p^*$ and set $h_0(m_i) = (g^a)^{\nu_i}$. Otherwise, $\mu_i = 0$ and $\mathcal{A}$ will check h_1-*list*.
     (a) If m_i has not been submitted to h_1 oracle, $\mathcal{A}$ will choose a random number $\nu_i \in \mathbb{Z}_p^*$ and set $h_0(m_i) = g^{\nu_i}$.
     (b) Else, there is a tuple $(m_i, h_1(m_i), \zeta_i, \kappa_i, \eta_i)$ in h_1-*list*. If $\zeta_i = 0$, $\mathcal{A}$ sets $\nu_i = \eta_i$ and $h_0(m_i) = g^{\nu_i}$. Otherwise $\zeta_i = 1$, $\mathcal{A}$ will choose a random number $\nu_i \in \mathbb{Z}_p^*$ and set $h_0(m_i) = g^{\nu_i}$.

Then $\mathcal{A}$ adds $(m_i, h_0(m_i), \mu_i, \nu_i)$ into h_0-*list* and returns $h_0(m_i)$ as the answer.

2. h_1 queries: At any time, $\mathcal{D}$ can issue an h_1 query for the message m_i. In response, $\mathcal{A}$ will maintain an h_1-*list* which stores his responses to such queries. For a new query, $\mathcal{A}$ chooses a number $\zeta_i \in \{0, 1\}$ such that $\Pr[\zeta_i = 1] = \delta$. If $\zeta_i = 1$, $\mathcal{A}$ will choose two random numbers $\kappa_i, \eta_i \in \mathbb{Z}_p^*$ and sets $h_1(m_i) = (g^{\kappa_i})^{\eta_i}$. Otherwise, $\zeta_i = 0$ and $\mathcal{A}$ will check h_0-*list*.
   (a) If m_i has not been submitted to h_0 oracle, $\mathcal{A}$ will choose two random numbers $\kappa_i, \eta_i \in \mathbb{Z}_p^*$ and sets $h_1(m_i) = (g^{\kappa_i}/g^b)^{\eta_i}$, where g^b is the input of 3-DDH problem.
   (b) Otherwise, there is a tuple $(m_i, h_0(m_i), \mu_i, \nu_i)$ in the h_0-*list*. If $\mu_i = 0$, $\mathcal{A}$ will choose a random number $\kappa_i \in \mathbb{Z}_p^*$ and set $\eta_i = \nu_i$. Then it computes $h_1(m_i) = (g^{\kappa_i}/g^b)^{\eta_i}$. Otherwise, $\mathcal{A}$ will choose two random numbers $\kappa_i, \eta_i \in \mathbb{Z}_p^*$ and sets $h_1(m_i) = (g^{\kappa_i}/g^b)^{\eta_i}$.
   Then $\mathcal{A}$ adds $(m_i, h_1(m_i), \zeta_i, \kappa_i, \eta_i)$ into the h_1-*list* and returns $h_1(m_i)$ as the answer.
3. h_2 queries: The same as the simulation in the proof of Theorem 3.

– Key Generation Oracle: At any time, $\mathcal{D}$ can require the i^{th} user's public key $pk_i = (X_i, Y_i)$. In response, $\mathcal{A}$ will maintain a pk-*list* which stores his responses to such queries.
1. If $i = \pi$, $\mathcal{A}$ sets $pk_i = pk_\pi = (X_\pi, Y_\pi) = (g^b, g^c)$ where g^b, g^c are the input of 3-DDH problem. Then $\mathcal{A}$ adds $(pk_i, ?, ?)$ into pk-*list*. The symbol "?" means $\mathcal{A}$ does not know the corresponding secret keys.
2. For other queries, $\mathcal{A}$ will choose two random numbers $x_i, y_i \in \mathbb{Z}_p^*$ and set $pk_i = (X_i, Y_i) = (g^{x_i}, g^{y_i})$. Then $\mathcal{A}$ adds (pk_i, x_i, y_i) into pk-*list*.

In both cases, $\mathcal{A}$ will return pk_i to $\mathcal{D}$ as the answer.

– Undeniable Sign Oracle: At any time, $\mathcal{D}$ can require an undeniable sign query (m_i, pk_j). Similarly, we assume that m_i has been submitted to h_0 oracle and h_1 oracle. Therefore, $(m_i, h_0(m_i), \mu_i, \nu_i)$ and $(m_i, h_1(m_i), \zeta_i, \kappa_i, \eta_i)$ have been in h_0-*list* and h_1-*list*, respectively.
1. If $pk_j \neq pk_\pi$, then $X_j = g^{x_j}, Y_j = g^{y_j}$ which means $\mathcal{A}$ knows the signer's secret key $sk_j = (x_j, y_j)$. Therefore, $\mathcal{A}$ runs the **Undeniable Sign** algorithm to compute the signature.
2. Otherwise $pk_j = pk_\pi$, then $X_j = g^b, Y_j = g^c$.
   (a) If $\mu_i = 0$ and $\zeta_i = 0$, then $h_0(m_i) = g^{\nu_i}$, $h_1(m_i) = (g^{\kappa_i}/g^b)^{\eta_i}$. In this case, the valid undeniable signature of m_i should be

$$\begin{aligned}\sigma_i &= h_0(m_i)^{bc} \cdot h_1(m_i)^c = (g^{\nu_i})^{bc} \cdot (g^{\kappa_i}/g^b)^{c\eta_i} \\ &= g^{bc\nu_i} \cdot (g^c)^{\kappa_i\eta_i} \cdot (g)^{-bc\eta_i}\end{aligned}$$

   Note that $g^{bc\nu_i} \cdot g^{-bc\eta_i} = 1$, because $\mathcal{A}$ sets $\nu_i = \eta_i$ when $\mu_i = 0$ and $\zeta_i = 0$. Therefore, $\mathcal{A}$ can compute the valid undeniable signature as $\sigma_i = (g^c)^{\kappa_i\eta_i}$.
   (b) Otherwise, $\mathcal{A}$ fails to simulate the Undeniable Sign Oracle and aborts.

– Verify Oracle: At any time, $\mathcal{D}$ can submit an undeniable verify query $(m_i, \sigma_i, pk_s, pk_v)$.

1. If $pk_s \neq pk_\pi$, which means $X_s = g^{x_s}, Y_s = g^{y_s}$ which means $\mathcal{A}$ knows the signer's secret key $sk_s = (x_s, y_s)$. Therefore, $\mathcal{A}$ can run the **Undeniable Verify** algorithm and **Confirmation**/**Disavowal** protocols.
2. Otherwise $pk_s = pk_\pi$ which means $X_s = g^b, Y_s = g^c$ and $\mathcal{A}$ does not know the signer's secret key. Then $\mathcal{A}$ responds to the query as following:
   (a) If (m_i, σ_i) is returned from Undeniable Sign Oracle, $\mathcal{A}$ will return 1 and simulate the **Confirmation** protocol as follows.

   $\mathcal{A}$ randomly chooses four numbers $c_s, c_v, d_s, d_v \in \mathbb{Z}_p$ computes $A = g^{d_s} Y_s^{c_s}$, $B = e(h_0(m_i), X_s)^{d_s}\ [e(\sigma_i, g)/e(h_1(m_i), Y_s)]^{c_s}$, $C = g^{d_v} Y_v^{c_v}$. Then $\mathcal{A}$ sets $\xi_i = m_i \| \sigma_i \| pk_v \| A \| B \| C$ and checks whether $(\xi_i, \cdot)$ has been in h_2-$list$. If ξ_i has been submitted to h_2 oracle previously, $\mathcal{A}$ will choose other four random numbers and recompute A, B, C until ξ_i has never been submitted to h_2 oracle. Then $\mathcal{A}$ adds $(\xi_i, c_s + c_v)$ into h_2-$list$ and returns (c_s, c_v, d_s, d_v) to $\mathcal{D}$.
   (b) Otherwise, (m_i, σ_i) is not returned from Undeniable Sign Oracle. According to Theorem 3, the probability that it is a valid undeniable signature is negligible. $\mathcal{A}$ will return 0 and simulate the **Disavowal** protocol as follows.

   $\mathcal{A}$ randomly chooses an element $A \in \mathbb{G}_2^*$ and five numbers c_s, c_v, $d_s, \widehat{d_s}, d_v \in \mathbb{Z}_p$. Then $\mathcal{A}$ computes $W = e(\sigma, g)/e(h_1(m_i), Y_s)$, $B = e(h_0(m_i), X_s)^{d_s}/(W^{\widehat{d_s}} \cdot A^{c_s})$, $C = g^{d_s}/Y_s^{\widehat{d_s}}$, $D = g^{d_v} Y_v^{c_v}$ and set $\xi_i = m_i \| \sigma_i \| pk_v \| A \| B \| C \| D$. Now, $\mathcal{A}$ checks whether $(\xi_i, \cdot)$ has been in h_2-$list$. If ξ_i has been submitted to h_2 oracle previously, $\mathcal{A}$ will choose some other random numbers until ξ_i has never been submitted to h_2 oracle. Then $\mathcal{A}$ adds $(\xi_i, c_s + c_v)$ into h_2-$list$ and returns $(A, c_s, c_v, d_s, \widehat{d_s}, d_v)$ to $\mathcal{D}$.

– Selectively Convert Oracle: At any time, $\mathcal{D}$ can submit a selectively convert query (m_i, pk_j).
  1. If $pk_j \neq pk_\pi$, which means $X_j = g^{x_j}, Y_j = g^{y_j}$ and $\mathcal{A}$ knows the signer's secret key $sk_j = (x_j, y_j)$. $\mathcal{A}$ computes the selective proof $\Pi_{(m_i, pk_j)} = h_0(m_i)^{y_j}$.
  2. If $pk_j = pk_\pi$ and $\mu_i = 0$. which means $X_j = g^b, Y_j = g^c$, and $h_0(m_i) = g^{\nu_i}$. $\mathcal{A}$ computes the selective proof $\Pi_{(m_i, pk_j)} = (g^c)^{\nu_i} = (g^{\nu_i})^c = h_0(m_i)^c$.
  3. Otherwise, $\mathcal{A}$ fails to simulate the Selectively Convert Oracle and aborts.
– Universally Convert Oracle: At any time, $\mathcal{D}$ can submit the universally convert query pk_i. Similarly, If $pk_i \neq pk_\pi$, $X_i = g^{x_i}, Y_i = g^{y_i}$. $\mathcal{A}$ can compute $\Pi_{pk_i} = g^{x_i y_i}$. Otherwise, $\mathcal{A}$ fails to simulate the Universally Convert Oracle and aborts.
– Corruption Oracle: At any time, $\mathcal{D}$ can submit the corruption query pk_i. Similarly, if $pk_i \neq pk_\pi$, then $X_i = g^{x_i}, Y_i = g^{y_i}$. Therefore $\mathcal{A}$ returns (x_i, y_i) to $\mathcal{F}$ as the answer. Otherwise, $\mathcal{A}$ fails to simulate the Corruption Oracle and aborts.

At the end of Phase 1, $\mathcal{D}$ submits the challenging message m^* and the public key pk_s^*. Similarly, we assume that $(m^*, h_0(m^*), \mu^*, \nu^*)$ and $(m^*, h_1(m^*), \zeta^*, \kappa^*, \eta^*)$

have been in h_0-*list* and h_1-*list*, respectively. If it is not true, $\mathcal{A}$ can generate the above tuples as the same way he answers $\mathcal{D}$'s h_0 and h_1 queries.

1. If $pk_s^* = pk_\pi$, $\mu^* = 1$ and $\zeta^* = 1$, which means $h_0(m^*) = (g^a)^{\nu^*}$, $h_1(m^*) = (g^{\kappa^*})^{\eta^*}$ and $pk_s^* = (X_\pi, Y_\pi) = (g^b, g^c)$, $\mathcal{A}$ returns $\sigma^* = h^{\nu^*} \cdot (g^c)^{\kappa^*\eta^*}$ as the challenging signature where h is the input of 3-DDH problem.
2. Otherwise, $\mathcal{A}$ fails to solve the 3-DDH problem and aborts.

After receiving the challenging signature σ^*, $\mathcal{D}$ can still submit queries to the above oracles with restrictions defined in Section 3.5. After all the queries have been made, $\mathcal{D}$ submits his guess γ' to $\mathcal{A}$. $\mathcal{A}$ forwards γ' as his answer to the 3-DDH problem. Note that if $\gamma' = 1$, then σ^* is a valid signature of the message m^* with probability $1/2 + \mathsf{Adv}\ \mathcal{D}_{INV,CUS}^{CMA,CPKA}$, which means $\sigma^* = h_0(m^*)^{bc} h_1(m^*)^c = (g^{abc})^{\nu^*}(g^c)^{\kappa^*\eta^*}$. Since $\mathcal{A}$ computes σ^* as $h^{\nu^*} \cdot (g^c)^{\kappa^*\eta^*}$, therefore $h = g^{abc}$. Otherwise, σ^* is not a valid undeniable signature and $h \neq g^{abc}$. Therefore, if $\mathcal{A}$ does not abort during the simulation, it can solve this instance of 3-DDH problem with the advantage $\mathsf{Adv}\ A_{\mathbb{G}_1}^{3-DDH} \geq \frac{\delta^2}{q_{KG}} \mathsf{Adv}\ \mathcal{D}_{INV,US}^{CMA,CPKA}$.

It's remaining to consider the probability that $\mathcal{A}$ does not abort during the simulation. As we can see, $\mathcal{A}$ does not abort during the simulation if and only if

1. $\mathcal{A}$ does not abort during the simulation of Undeniable Sign Oracle, which happens with probability $((1 - \frac{1}{q_{KG}}) + \frac{1}{q_{KG}}(1-\delta)^2)^{q_{US}} \geq (1-\delta)^{2q_{US}}$.
2. $\mathcal{A}$ does not abort during the simulation of Selectively Convert Oracle, which happens with probability $((1 - \frac{1}{q_{KG}}) + \frac{1}{q_{KG}}(1-\delta))^{q_{SC}} \geq (1-\delta)^{q_{SC}}$.
3. $\mathcal{A}$ does not abort during the simulation of Universally Convert Oracle, which happens with probability $(1 - \frac{1}{q_{KG}})^{q_{UC}}$.
4. $\mathcal{A}$ does not abort during the simulation of Corruption Oracle, which happens with probability $(1 - \frac{1}{q_{KG}})^{q_C}$.

We can obtain that $\mathcal{A}$ does not abort during the simulation with the probability greater than $(1-\delta)^{2q_{US}+q_{SC}}(1 - \frac{1}{q_{KG}})^{q_{UC}+q_C}$. Therefore, the advantage that $\mathcal{A}$ has to solve the 3-DDH problem is $\mathsf{Adv}\ A_{\mathbb{G}_1}^{3-DDH} \geq \frac{\delta^2}{q_{KG}}(1-\delta)^{2q_{US}+q_{SC}}(1 - \frac{1}{q_{KG}})^{q_{UC}+q_C} \mathsf{Adv}\ \mathcal{D}_{INV,US}^{CMA,CPKA}$. When $\delta = 2/(2q_{US} + q_{SC} + 2)$, this probability is maximized at $\mathsf{Adv}\ A_{\mathbb{G}_1}^{3-DDH} = \frac{4}{q_{KG}(2q_{US}+q_{SC})^2}(1 - \frac{2}{2q_{US}+q_{SC}+2})^{2q_{US}+q_{SC}+2}(1 - \frac{1}{q_{KG}})^{q_{UC}+q_C}\ \mathsf{Adv}\ \mathcal{D}_{INV,US}^{CMA,CPKA}$. □

Theorem 5. *If there exists a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *adversary* $\widetilde{S}_{Sound,\ CUS}^{CMA,\ CPKA}$ *who can make additional* q_H *queries to random oracles and win the game of* ***Soundness*** *defined in Section 3.7 with non-negligible success probability* $\mathsf{Succ}\ \widetilde{S}_{Sound,\ CUS}^{CMA,\ CPKA}$*, then there exists an algorithm* $\mathcal{A}$ *who can use* $\widetilde{S}$ *to solve a random instance of Discrete Logarithm problem with probability* $Succ_{\mathcal{A},\mathbb{G}_1}^{DL} \geq \frac{1}{q_{KG}}(1 - \frac{1}{q_{KG}})^{q_C} \mathsf{Succ}\ \widetilde{S}_{Sound,\ CUS}^{CMA,CPKA}$ *in polynomial time.*

Theorem 6. *If there exists a* $(t, q_{KG}, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ *impersonator* $\mathcal{I}_{CUS}^{CMA,\ CPKA}$ *who can make additional* q_H *queries to random oracles and have non-negligible success probability* $\mathsf{Succ}\ \mathcal{I}_{CASE_i,\ CUS}^{CMA,\ CPKA}$ *for some* $i \in \{1, 2, 3\}$ *in the*

game of ***Impersonation*** *defined in Section 3.8, then there exists an algorithm* $\mathcal{A}$ *who can use* $\mathcal{I}$ *to solve a random instance of Computational Diffie-Hellman problem with non-negligible probability in polynomial time.*

Due to page limitation, the proofs of Theorem 5 and 6 will be presented in the full version of this paper.

4.3 Comparison with Other Schemes

In this section, we make a comparison between our scheme and the schemes proposed in [8,15,27]. For other existing schemes, some of them [25,33,23] are only selectively convertible, and others [29,30,31] are lacking of formal proofs. Among the schemes in [8,15,27], Laguillaumie and Vergnaud's scheme [27] has the shortest signature length (272 bits) but the highest computation cost. It requires 1 exponentiation in the signature generation and 2 pairings in the verification. In our scheme, the sign algorithm needs 2 exponentiations in $\mathbb{G}_1$, 2 hashes mapping to $\mathbb{G}_1$ and each verification requires 3 parings, which is slightly more time consuming than the scheme in [27]. However, the following table shows that our new scheme performs better in other aspects. Firstly, our scheme enjoys shorter signature length under the same system parameters in [27]. The signature of our scheme is only an element in the group $\mathbb{G}_1$. Secondly, the invisibility of our scheme is based on the hardness of 3-DDH problem, which is much better than $(\ell, 1)$-xyz-DCAA problem. As the authors said in [27], their scheme was also designed according to the 3-DDH problem, and instead, they created a decisional problem $(\ell, 1)$-xyz-DCAA to prove the invisibility and anonymity of their scheme.

Scheme	Signature	Selective and Universal Proof	Unforgeability	Invisibility
Scheme in [27]	$\lvert\mathbb{G}_1\rvert + \lvert n_r\rvert$	$\lvert\mathbb{G}_1\rvert$	CDH	$(\ell, 1)$-xyz-DCAA
Our Proposed Scheme	$\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_1\rvert$	CDH	3-DDH

Notations. $|\mathbb{G}_1|$: bit length of an element in the group $\mathbb{G}_1$. $|n_r|$: bit length of the random salt used in [27] which is set to be 112 in their scheme.

5 Conclusion

In this paper, we formalized the security model of the convertible undeniable signature scheme. The new model clearly defines the security properties that a convertible undeniable signature scheme should satisfy. Based on the new model, we proposed a new construction from pairings. Compared with the other schemes in the literature, our construction has the shortest signature length while providing both selective and universal convertibility. Meanwhile, all security properties of the new construction are formally proven under some conventional complexity assumptions.

Acknowledgements. The authors would like to thank Dr. Qianhong Wu and the anonymous referees for their valuable comments.

References

1. Boyar, J., Chaum, D., Damgård, I.B., Pedersen, T.P.: Convertible Undeniable Signatures. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 189–205. Springer, Heidelberg (1991)
2. 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)
3. Biehl, I., Paulus, S., Takagi, T.: Efficient Undeniable Signature Schemes Based on Ideal Arithmetic in Quadratic Orders. In: Designs, Codes and Cryptography, vol. 31(2), pp. 99–123. Springer, Netherlands (2004)
4. Chaum, D., Antwerpen, H.v.: Undeniable Signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
5. Chaum, D.: Zero-Knowledge Undeniable Signatures (Extended Abstract). In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 458–464. Springer, Heidelberg (1991)
6. Chaum, D., Heijst, E.v., Pfitzmann, B.: Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 470–484. Springer, Heidelberg (1992)
7. Diffie, W., Hellman, M.: New directions in cryptography. IEEE IT 22, 644–654 (1976)
8. Damgård, I.B., Pedersen, T.P.: New Convertible Undeniable Signature Schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 372–386. Springer, Heidelberg (1996)
9. Desmedt, Y., Yung, M.: Weaknesses of Undeniable Signature Schemes (Extended Abstract). In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 205–220. Springer, Heidelberg (1991)
10. Fujioka, A., Okamotoa, T., Ohta, K.: Interactive Bi-Proof Systems and Undeniable Signature Schemes. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 243–256. Springer, Heidelberg (1991)
11. Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure Against Adaptively Chosen Message Attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
12. Galbraith, S.D., Mao, W., Paterson, K.G.: RSA-Based Undeniable Signatures for General Moduli. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 200–217. Springer, Heidelberg (2002)
13. Galbraith, S.D., Mao, W.: Invisibility and Anonymity of Undeniable and Confirmer Signatures. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 80–97. Springer, Heidelberg (2003)
14. Gennaro, R., Krawczyk, H., Rabin, T.: RSA-Based Undeniable Signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 132–149. Springer, Heidelberg (1997)
15. Gennaro, R., Rabin, T., Krawczyk, H.: RSA-Based Undeniable Signatures. Journal of Cryptology 13(4), 397–416 (2000)
16. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
17. Jongkook, L., Shiryong, R., Jeungseop, K., Keeyoung, Y.: A New Undeniable Signature Scheme Using Smart Cards. In: Honary, B. (ed.) Cryptography and Coding. LNCS, vol. 2260, pp. 387–394. Springer, Heidelberg (2001)

18. Jakobsson, M.: Blackmailing Using Undeniable Signatures. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 425–427. Springer, Heidelberg (1995)
19. Furukawa, J., Kurosawa, K., Imai, H.: An Efficient Compiler from Σ-Protocol to 2-Move Deniable Zero-Knowledge. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 46–57. Springer, Heidelberg (2006)
20. Kudla, C., Paterson, K.G.: Non-interactive Designated Verifier Proofs and Undeniable Signatures. In: Smart, N.P. (ed.) Cryptography and Coding. LNCS, vol. 3796, pp. 136–154. Springer, Heidelberg (2005)
21. Kim, S., Won, D.: Threshold Entrusted Undeniable Signature. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 195–203. Springer, Heidelberg (2005)
22. Kurosawa, K., Heng, S.-H.: 3-Move Undeniable Signature Scheme. In: Fuhr, N., Lalmas, M., Malik, S., Szlávik, Z. (eds.) EUROCRYPT 2005. LNCS, vol. 3493, pp. 181–197. Springer, Heidelberg (2005)
23. Kurosawa, K., Takagi, T.: New Approach for Selectively Convertible Undeniable Signature Schemes. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 428–443. Springer, Heidelberg (2006)
24. Laguillaumie, F., Paillier, P., Vergnaud, D.: Universally Convertible Directed Signatures. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 682–701. Springer, Heidelberg (2005)
25. Libert, B., Quisquater, J.J.: Identity Based Undeniable Signatures. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 112–125. Springer, Heidelberg (2004)
26. Lyuu, Y.-D., Wu, M.-L.: Convertible Group Undeniable Signatures. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 48–61. Springer, Heidelberg (2003)
27. Laguillaumie, F., Vergnaud, D.: Time-Selective Convertible Undeniable Signatures. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 154–171. Springer, Heidelberg (2005)
28. Laguillaumie, F., Vergnaud, D.: Short Undeniable Signatures Without Random Oracles: The Missing Link. In: Maitra, S., Madhavan, C.E.V., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 283–296. Springer, Heidelberg (2005)
29. Miyazaki, T.: An Improved Scheme of the Gennaro-Krawczyk-Rabin Undeniable Signature System Based on RSA. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 135–149. Springer, Heidelberg (2001)
30. Michels, M., Petersen, H., Horster, P.: Breaking and Repairing a Convertible Undeniable Signature Scheme. In: Third ACM Conference on Computer and Communications Security, pp. 148–152. ACM Press, New York (1996)
31. Michels, M., Stadler, M.: Efficient Convertible Undeniable Signature Schemes. In: The 4th International Workshop on Selected Areas in Cryptography - SAC'97. pp. 231–244 (1997)
32. Monnerat, J., Vaudenay, S.: Undeniable Signatures Based on Characters: How to Sign with One Bit. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 69–85. Springer, Heidelberg (2004)
33. Monnerat, J., Vaudenay, S.: Generic Homomorphic Undeniable Signatures. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 354–371. Springer, Heidelberg (2004)
34. Monnerat, J., Vaudenay, S.: Optimization of the MOVA Undeniable Signature Scheme. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 196–209. Springer, Heidelberg (2005)
35. Monnerat, J., Vaudenay, S.: Short 2-Move Undeniable Signatures. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 19–36. Springer, Heidelberg (2006)

36. Ogata, W., Kurosawa, K., Heng, S.-H.: The Security of the FDH Variant of Chaum Undeniable Signature Scheme. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 328–345. Springer, Heidelberg (2005)
37. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
38. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology 13(3), 361–396 (2000)
39. Vergnaud, D., Aimani, L. E.: Gradually Convertible Undeniable Signatures. In: Applied Cryptography and Network Security. 5th International Conference, ACNS 2007. LNCS, Springer, Heidelberg (to appear, 2007)
40. Wang, G.: An Attack on Not-interactive Designated Verifier Proofs for Undeniable Signatures. Available online: http://eprint.iacr.org/2003/243
41. Wang, G., Qing, S., Wang, M., Zhou, Z.: Threshold Undeniable RSA Signature Scheme. In: Qing, S., Okamoto, T., Zhou, J. (eds.) ICICS 2001. LNCS, vol. 2229, pp. 221–232. Springer, Heidelberg (2001)
42. Wang, G., Zhou, J., Deng, R.H.: On the Security of the Lee-Hwang Group-Oriented Undeniable Signature Schemes. In: Katsikas, S.K., Lopez, J., Pernul, G. (eds.) TrustBus 2004. LNCS, vol. 3184, pp. 289–298. Springer, Heidelberg (2004), Available online: http://eprint.iacr.org/2002/150
43. Zhang, F., Safavi-Naini, R., Susilo, W.: Attack on Han et al.'s ID-based Confirmer (Undeniable) Signature at ACM-EC'03. Available online: http://eprint.iacr.org/2003/129

Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key*

Fuchun Guo[1], Yi Mu[2], and Zhide Chen[1]

[1] Key Lab of Network Security and Cryptology
School of Mathematics and Computer Science
Fujian Normal University, Fuzhou, China
fuchunguo1982@hotmail.com,
zhidechen@fjnu.edu.cn
[2] School of Computer Science and Software Engineering
University of Wollongong, Wollongong NSW 2522, Australia
ymu@uow.edu.au

Abstract. The beauty of identity-based encryption (IBE) lies in the convenience of public key handling, in the sense that any identification such as an email address, a name, or an IP number can serve as a public key of a party. However, such convenience is not inherited by a system where a party possesses many identities (e.g., many email addresses) and has to use them as his public keys. When this system is handled with a standard IBE, the user must manage all the private keys that are associated with all the public keys (identities). However, keeping these private keys is inconvenient to the user. In this paper, we solve this problem by proposing a novel identity-based encryption where we set a private key that maps multiple public keys (identities); namely, we can use a private key to decrypt multiple ciphertexts; each was encrypted with a different public key (identity).

Keywords: ID-based Encryption, Pairing.

1 Introduction

In 1984, Shamir [24] introduced the notion of identity-based (or ID-based) cryptography and presented a concrete signature scheme. The merit of an identity based system is that the public key can be an arbitrary string. Shamir's original motivation was to simplify certificate management in e-mail systems. In a traditional public key encryption such as ElGamal encryption, a public key is a random string so that we need a public key certificate to show the connection

* Supported by National Natural Science Foundation of China (#60502047), Education Bureau of Fujian Province (#JB05329), Science and Technology of Fujian Province (2006F5036) and Open Funds of Key Lab of Fujian Province University Network Security and Cryptology (07B001).

T. Takagi et al. (Eds.): Pairing 2007, LNCS 4575, pp. 392–406, 2007.

of a user and a public key. Such public key certification system is referred to as Public Key Infrastructure (PKI). If a public key can be set discretionarily, the public key can be set using user's identification information such as telephone number or email address. Those natural information for users eliminate the need of PKI.

Although Shamir [24] successfully constructed an ID-based signature, he could not work out how to construct ID-based encryption (IBE). This problem had remained open until Boneh and Franklin [7] introduced the notion of bilinear pairings and successfully constructed a concrete ID-based encryption scheme with CCA2 security in a random oracle model [2]. Based on it, some IBE schemes with a further research on its security have been published [10,4,26,16]. Although IBE has many applications, the shortcoming of IBE is the key escrow problem; namely, private keys must be computed by a trusted authority called Private Key Generator (PKG). There have been some attempts to solve the key escrow problem [11,20,15,25,19]. In this paper, we do not address the issue of key escrow. Instead, we will focus on a very interesting problem that is motivated by the following scenario.

Bob is a consultant for n companies and has n email addresses; each is used for a different company. For example, each company could assign an email address to Bob, who uses it to receive all emails associated with this particular company. Assume that a traditional IBE is applied in order to secure the system. Naturally, Bob would like to select the email address from a company as his public key in order to secure his email system for that company. It would not make sense if a company used another company's email alias as Bob's public key. He therefore turns out having n public keys and n corresponding private keys. When Bob saw so many keys to manage, he got headache. What Bob wishes is that he could just use a single private key to decrypt all ciphertexts sent to him regardless which public key was used for encryption.

In this paper, we present a novel IBE-SK (Identity-Based Encryption with single private key) scheme aiming to make Bob's wish come true. In our scheme, we allow a private key to map to multiple public keys which can be selected randomly as identities of Bob .

1.1 Related Works

Hierarchical Identity-Based Encryption (HIBE) was first proposed in [18] and [17] in 2002 and has further developed in [6,13]. It is a generalization of IBE that mirrors an organizational hierarchy, which allows the task of generating private keys to be delegated to lower levels. A 2-level HIBE system could achieve Bob's wish by this way: If Bob has two email addresses "bob@hotmail.com" and "bob@gmail.com", then he could ask the PKG for a private key corresponding to the ID="Bob". Anyone who sends email to "bob@hotmail.com" or "bob@gmail.com" can encrypt the email using the corresponding public key "Bob‖bob@hotmail.com" or "Bob‖bob@gmail.com" with a 2-level HIBE system. For both cases, Bob can decrypt emails with the only private key of ID "Bob".

The 2-level HIBE system seems having cured Bob's headache, but actually the treatment is not complete. The problem arises when Bob got an email address without his identity, e.g. an email address of "consultant@gmail.com". The case is normal but out of control because there is no an obvious identity. In this case, suppose Bob has the two email addresses "bob@hotmail.com" and "consultant@gmail.com", then a 2-level HIBE system will fail and Bob cannot use the single private key of ID "Bob" to decrypt emails for both cases. In fact, a 2-level HIBE system can be considered as encryption with the public key "Bob". It does not help!

Road Map: In Section 2, we will provide the definitions of our scheme and its security requirement. In Section 3, we present our novel IBE-SK scheme. In Section 4, we will analyze the security of our scheme. We conclude our paper in Section 5.

2 Definitions

In this section, we define our scheme and the security requirement for our scheme.

Definition 1. *Our IBE-SK scheme can be described as the four PPT algorithms:* Setup, KeyGen, Encrypt *and* Decrypt.

- Setup*: takes as input a security parameter* 1^k*, and outputs a master secret key* s*, the corresponding master public key* K_{pub}*, and some auxiliary parameter* AUX*.*
- KeyGen*: takes as input the multiple ID's* $\{ID_1, ID_2, \cdots, ID_j\}$ $(j \leq q)$ *of a party and the master secret key* s*, and outputs a single private key* d *for the party.*
- Encrypt*: takes as input a message* M*, the master public key* K_{pub}*, an* $ID_i \in \{ID_1, ID_2, \cdots, ID_j\}$*, and* AUX*, and outputs ciphertext* C*.*
- Decrypt*: takes as input* $\langle ID_i, C \rangle$*,* AUX*, the private key* d *and other ID's* $\{ID_1, \cdots, ID_{i-1}, ID_{i+1}, \cdots ID_j\}$*, and outputs the message* M*.*

2.1 Security Model

We say that our scheme $\mathcal{E}$ with a security parameter 1^k is indistinguishably secure against an adaptive chosen ciphertext attack (IND-ID-CCA2) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following IND-ID-CCA2 game:

Setup: The challenger takes a security parameter 1^k and runs the algorithm Setup. It gives the adversary the resulting master public key and keeps the master secret key to itself.

Phase 1: The adversary issues queries $q_1, q_2, \cdots, q_m$, when the queries are for key generation, the adversary can issue q queries one time at most:

- Key generation queries $\langle ID_{i+1}, ID_{i+2}, \cdots, ID_{i+j} \rangle$ $(j \leq q)$. The challenger responds by running algorithm KeyGen to generate the private key $d_{i,j}$ corresponding to the multiple public keys $\langle ID_{i+1}, ID_{i+2}, \cdots, ID_{i+j} \rangle (j \leq q)$. It sends $d_{i,j}$ to the adversary.
- Decryption query $\langle ID_i, C_i \rangle$. The challenger responds by running algorithm KeyGen to generate the private key d_i corresponding to ID_i. It then runs algorithm Decrypt to decrypt the ciphertext C_i using the private key d_i. It sends the resulting plaintext to the adversary.

These queries may be asked adaptively according to the replies of queries.

Challenge: Once the adversary decides that **Phase 1** is over it outputs two equal length plaintexts m_0, m_1 and an identity ID_{ch} on which it wishes to be challenged. The only constraint is that ID_{ch} did not appear in any key generation query in Phase 1. The challenger picks a random bit $c \in \{0, 1\}$ and sets $C = \mathsf{Encrypt}(K_{pub}, ID_{ch}, m_c)$. It sends C as the challenge to the adversary.

Phase 2: Same to the **Phase 1**. The constraint is that ID_{ch} should not appear in any key generation query and $\langle ID_{ch}, C \rangle$ should not appear in any decryption query.

Guess: Finally, the adversary outputs a guess $c' \in \{0, 1\}$ and wins the game if $c = c'$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA2 adversary. We define the advantage of adversary $\mathcal{A}$ in attacking the IBE-SK scheme $\mathcal{E}$ as the function of the security parameter k:

$$Adv_{\mathcal{E},\mathcal{A}}(k) = |Pr[c' = c] - \frac{1}{2}|$$

We say that the IBE-SK scheme is secure against adversary $\mathcal{A}$ in IND-ID-CCA2 model if the function $Adv_{\mathcal{E},\mathcal{A}}(k)$ is negligible for sufficiently large k. Our IBE-SK scheme will be proved secure in the random oracle model [2].

2.2 Bilinear Pairing

Let $\mathbb{G}_1$ be (additive) cyclic group of prime order p. Let P, Q be a generator of $\mathbb{G}_1$. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ (here $\mathbb{G}_2$ is a multiplicative group such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$) is called a bilinear pairing if this map satisfies the following properties:

- Bilinear: for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
- Non-degeneracy: $\hat{e}(P, P) \neq 1$. In other words, if P be a generator of $\mathbb{G}_1$, then $\hat{e}(P, P)$ generates $\mathbb{G}_2$.
- Computability: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

2.3 Complexity Assumption

q-SDH Problem. Let $\mathbb{G}_1$ be a group of prime order p and let P be a generator of $\mathbb{G}_1$. The q-SDH problem in $\mathbb{G}_1$ is that :Given $\langle P, sP, s^2P, \cdots, s^qP\rangle$ for some $s \in \mathbb{Z}_p^*$ ($q \ll p$), compute $\langle c, \frac{1}{c+s}P\rangle$ for any $c \in \mathbb{Z}_p^* \backslash \{-s\}$.

The q-SDH problem originates from a weaker assumption introduced by Mitsunari *et al.*[21] to construct traitor tracing schemes. Here, we extend it to a q-BSDH problem in bilinear pairing.

q-BSDH Problem. Let $\mathbb{G}_1, \mathbb{G}_2$ be two groups of prime order p. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear map and let P be a generator of $\mathbb{G}_1$. The q-BSDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ is that: Given $\langle P, sP, s^2P, \cdots, s^qP, bP\rangle$ for some $s, b \in \mathbb{Z}_p^*$ ($q \ll p$), compute $\langle c, \hat{e}(\frac{1}{c+s}P, bP)\rangle$ for any $c \in \mathbb{Z}_p^* \backslash \{-s\}$.

q-BSDH Assumption. Let $\mathcal{G}$ be a generator of bilinear map $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ with a secure parameter 1^k (see [7]) . We define that an algorithm $\mathcal{A}$ has an advantage $Adv_{\mathcal{G},\mathcal{A}}(k)$ in solve the q-BSDH problem, where

$$Adv_{\mathcal{G},\mathcal{A}}(k) = \mathsf{Pr}[\mathcal{A}(p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, sP, s^2P, \cdots, s^qP, bP) = \hat{e}(\frac{1}{c+s}P, bP) \wedge \forall c \in \mathbb{Z}_p^* \backslash \{-s\} \leftarrow \mathcal{A}].$$

We say that $\mathcal{G}$ satisfies the q-BSDH assumption if for any randomized polynomial time (in k) algorithm $\mathcal{A}$ we have that $Adv_{\mathcal{G},\mathcal{A}}(k)$ is a negligible function. When $\mathcal{G}$ satisfies the q-BSDH assumption we say that q-BSDH is hard in groups generated by $\mathcal{G}$.

Explanation to q-SDH Problem. When q is the maximum number of private key owners that an adversary may corrupt in an active attack, the above q-SDH problem and other related assumptions have recently come under (limited) number theoretic attack in a recent paper by Cheon [9]. But in our scheme, the number of q is the maximum value that a single private key can map, which can be very small. We believe that each user with 10 identities will be enough, i.e. $q = 10$. So, the number theoretic attack will fail and there is still no efficient algorithm to solve the assumption.

3 The Novel ID-Based Encryption

Without losing generality of the scheme, we assume that Bob has two IDs (ID_1, ID_2) and a single private key of d_{ID_1,ID_2} which maps ID_1 and ID_2. The following scheme shows how to use d_{ID_1,ID_2} to decrypt a ciphertext under the public key ID_1.

- Setup: Given a security parameter 1^k, the algorithm $\mathcal{G}$ outputs a master public key K_{pub} and a master secret key s as follow:
  - Step 1: Run $\mathcal{G}$ on input 1^k to generate a prime p, two groups $\mathbb{G}_1, \mathbb{G}_2$ of order p, and an admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Choose a random generator $P_{pub}^0 = P \in \mathbb{G}_1^*$.

Step 2: Pick a random $s \in \mathbb{Z}_p^*$ and set $P_{pub}^k = s^k P, k = 1, 2, \cdots, q$; The plaintext space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1^q \times \{0,1\}^{2n}$.

Step 3: Choose four cryptography hash functions: $H_1(x, Q) : \{0,1\}^* \to \mathbb{G}_1$; $H_2 : \mathbb{G}_2 \to \{0,1\}^n$; $H_3 : \{0,1\}^* \to \mathbb{Z}_p^*$; $H_4 : \{0,1\}^n \to \{0,1\}^n$.

The system parameters are

$$K_{pub} = \langle p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_{pub}^0, P_{pub}^1, \cdots, P_{pub}^q, H_1, H_2, H_3, H_4 \rangle$$

Here, the hash function $H_1(x, Q)$ is set by $H_1(x, Q) = H_0(x)Q$, where H_0 is a hash function of $H_0 : \{0,1\}^* \to \mathbb{Z}_p^*$ and Q is a generator of $\mathbb{G}_1$. So we have $H_1(x, Q) : \{0,1\}^* \to \mathbb{G}_1$. The hash function of H_0 will be reused in the next phase.

– KeyGen: For the given $\{ID_1, ID_2\} \in \{0,1\}^*$, the algorithm sets the private key d_{ID_1,ID_2} as

$$d_{ID_1,ID_2} = \frac{1}{H_0(ID_1) + s} \cdot \frac{1}{H_0(ID_2) + s} P, \quad H_0(ID_1),\ H_0(ID_2) \neq -s \bmod p$$

.

– Encrypt: To encrypt $m \in \{0,1\}^n$ under ID_1, do the following:
  - Compute $Q_{ID_1}^k = H_1(ID_1, P_{pub}^k) = H_0(ID_1)P_{pub}^k, k = 0, 1, \cdots, q-1$;
  - Choose a random $\sigma \in \{0,1\}^n$ and set $r = H_3(\sigma, m)$;
  - Compute $rQ_{ID_1}^k + rP_{pub}^{k+1}, k = 0, 1, \cdots, q-1$;

  Set the ciphertext to be

$$\langle U_1, U_2, \cdots, U_q, V, W \rangle = \langle rQ_{ID_1}^0 + rP_{pub}^1, rQ_{ID_1}^1 + rP_{pub}^2, \cdots, rQ_{ID_1}^{q-1} + rP_{pub}^q, H_2(\hat{e}(P,P)^r) \oplus \sigma, H_4(\sigma) \oplus m \rangle.$$

– Decrypt: Let $C = \langle U_1, U_2, \cdots, U_q, V, W \rangle$ be a ciphertext under ID_1. To decrypt C using the private key d_{ID_1,ID_2} and ID_2,
  - Compute $U = H_1(ID_2, U_1) + U_2$;
  - Compute $V \oplus H_2(\hat{e}(U, d_{ID_1,ID_2})) = \sigma'$;
  - Compute $m' = H_4(\sigma') \oplus W$;
  - Set $r' = H_3(\sigma', m')$ and check if $U_1 = r'(Q_{ID_1}^0 + P_{pub}^1)$. If true, accept $m' = m$ as the decryption; otherwise, abort.

It is easy to see that our scheme is correct:

$$U = H_1(ID_2, U_1) + U_2 = H_0(ID_2)(rQ_{ID_1}^0 + rP_{pub}^1) + rQ_{ID_1}^1 + rP_{pub}^2$$
$$= r(H_0(ID_1) + s)(H_0(ID_2) + s)P.$$

$$\hat{e}(U, d_{ID_1,ID_2}) = \hat{e}(r(H_0(ID_1)+s)(H_0(ID_2)+s)P, \frac{1}{H_0(ID_1) + s} \cdot \frac{1}{H_0(ID_2) + s} P)$$
$$= \hat{e}(P,P)^r.$$

We can easily extend it into the full scheme for multiple ID's (> 2). A general private key $d_{ID_1,\cdots,ID_q}$ for $\{ID_1, ID_2, \ldots, ID_q\}$ is computed as follow:

$$d_{ID_1,\cdots,ID_q} = \frac{1}{H_0(ID_1)+s} \cdot \frac{1}{H_0(ID_2)+s} \cdots \frac{1}{H_0(ID_q)+s} P, H_0(ID_i) \neq -s \bmod p$$

Observe that $U_i = rQ^i_{ID_1} + rP^{i+1}_{pub} = rs(rQ^{i-1}_{ID_1} + rP^i_{pub}) = sU_{i-1}$. The ciphertext is the same:

$$\langle U_1, U_2, \cdots, U_q, V, W\rangle = \langle U_1, sU_1, s^2U_1, \cdots, s^{q-1}U_1, V, W\rangle.$$

Then, for a general $r(H_0(ID_1)+s)\cdots(H_0(ID_q)+s)P = (H_0(ID_2)+s)\cdots(H_0(ID_q)+s)U_1$, we can compute it from $\{ID_2, ID_3, \cdots, ID_q\}$ and $U_1, sU_1, s^2U_1, \cdots, s^{q-1}U_1$ in the hash function $H(x, Q)$. So, Bob's single private key can map q multiple public keys at most. That is, we can use the single private key to decrypt q ciphertexts using different public keys.

4 Security

Since we will make use of hybrid encryption [14] in the security proof, we will first describe it.

4.1 Hybrid Encryption

The notion of Hybrid-Encryption was introduced by Fujisaki and Okamoto [14]. Let $\mathcal{E}$ be a probabilistic public key encryption scheme. We denote by $\mathcal{E}_{pk}(m, r)$ the encryption of m using the random bits r under the public key pk. Fujisaki and Okamoto define the hybrid encryption scheme $\mathcal{E}^{hy}$ as:

$$\mathcal{E}^{hy}_{pk}(m) = \langle \mathcal{E}_{pk}(\sigma||H_3(\sigma, m)), H_4(\sigma) \oplus m\rangle.$$

Here, σ is generated at random and H_3, H_4 are suitable cryptographic hash functions. Fujisaki and Okamoto show that if $\mathcal{E}$ is IND-CPA secure, then $\mathcal{E}^{hy}$ is IND-CCA2 secure. With our IBE-SK, Hybrid-Encryption can be readily realized in the following encryptions.

MPub: The public key is $K_{pub} = \langle p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P^0_{pub}, P^1_{pub}, \cdots, P^q_{pub}, H_2, Q^0_{ID}, Q^1_{ID}, \cdots, Q^{q-1}_{ID}\rangle$ and the private key is $d_{ID} = \frac{1}{c+s}P$, where $Q^k_{ID} = cP^k_{pub}$ for some $c \in \mathbb{Z}^*_p \backslash \{-s\}$. The phases of Encrypt and Decrypt are as follows:

– Encrypt: To encrypt $m \in \{0,1\}^n$ under the public key K_{pub}, implement the following: (1) choose a random $r \in \mathbb{Z}^*_p$ and compute $rQ^k_{ID} + rP^{k+1}_{pub}, k = 0, 1, \cdots, q-1$, and (2) set the ciphertext to be

$$\langle U_1, U_2, \cdots, U_q, V\rangle = \langle rQ^0_{ID} + rP^1_{pub}, rQ^1_{ID} + rP^2_{pub}, \cdots, rQ^{q-1}_{ID} + rP^q_{pub}, H_2(\hat{e}(P,P)^r) \oplus m\rangle.$$

– Decrypt: Let $C = \langle U_1, U_2, \cdots, U_q, V\rangle$ be a ciphertext using the public key of K_{pub}. With the private key $d_{ID} = \frac{1}{c+s}P$, the ciphertext is decrypted by $m = V \oplus H_2(\hat{e}(U_1, d_{ID}))$.

MPub^{hy}: The public key is $K_{pub} = \langle p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P^0_{pub}, P^1_{pub}, \cdots, P^q_{pub}, H_2, H_3, H_4, Q^0_{ID}, Q^1_{ID}, \cdots, Q^{q-1}_{ID}\rangle$ and the private key is $d_{ID} = \frac{1}{c+s}P$, where $Q^k_{ID} = cP^k_{pub}$ for some $c \in \mathbb{Z}^*_p \backslash \{-s\}$. The phases of Encrypt and Decrypt are as follows:

– Encrypt: To encrypt $m \in \{0,1\}^n$ under the public key K_{pub} do the following:
  - Choose a random $\sigma \in \{0,1\}^n$ and set $r = H_3(\sigma, m)$;
  - Compute $rQ^k_{ID} + rP^{k+1}_{pub}, k = 0, 1, \cdots, q-1$;

  Set the ciphertext to be

$$\langle U_1, U_2, \cdots, U_q, V, W\rangle = \langle rQ^0_{ID}+rP^1_{pub}, rQ^1_{ID}+rP^2_{pub}, \cdots, rQ^{q-1}_{ID}+rP^q_{pub}, H_2(\hat{e}(P,P)^r) \oplus \sigma, H_4(\sigma) \oplus m\rangle.$$

– Decrypt: Let $C = \langle U_1, U_2, \cdots, U_q, V, W\rangle$ be a ciphertext using the public key of K_{pub}. To decrypt C using the private key d_{ID}:
  - Compute $V \oplus H_2(\hat{e}(U_1, d_{ID})) = \sigma'$;
  - Compute $m' = H_4(\sigma') \oplus W$;
  - Set $r' = H_3(\sigma', m')$ and check if $U_1 = r'(Q^0_{ID}+P^1_{pub})$, if not,abort, accept $m' = m$ as the decryption.

According to the definition of Hybrid-Encryption in [14], we know that MPub^{hy} is a hybrid encryption based on the general public encryption of MPub.

4.2 Security

The following theorem shows that our scheme is secure against IND-ID-CCA2 assuming q-BSDH is hard in groups generated by $\mathcal{G}$.

Theorem 1. *Let the hash functions H_1, H_2, H_3, H_4 be random oracles. Then IBE-SK scheme is secure against IND-ID-CCA2 assuming q-BSDH is hard in groups generated by $\mathcal{G}$. Suppose there is an IND-ID-CCA2 adversary $\mathcal{A}$ that has advantage $\epsilon(k)$ against the IBE-SK scheme and $\mathcal{A}$ runs in time at most $t(k)$. Suppose $\mathcal{A}$ makes at most q_E key generation queries, at most q_D decryption queries, and at most $q_{H_2}, q_{H_3}, q_{H_4}$ queries to the hash functions H_2, H_3, H_4 respectively. Then there is a q-BSDH algorithm $\mathcal{B}$ for $\mathcal{G}$ with running time $t_1(k)$ where:*

$$Adv_{\mathcal{G},\mathcal{B}}(k) \geq 2FO_{adv}(\frac{\epsilon(k)}{e(1+q_E+q_D)}, q_{H_4}, q_{H_3}, q_D)/q_{H_2}$$

$$t_1(k) \leq FO_{time}(t(k), q_{H_4}, q_{H_3}).$$

The proof of Theorem 1 is based on the following result of Fujisaki and Okamoto.

Theorem 2 (Theorem 14 in [14]). *Suppose $\mathcal{A}$ is an IND-CCA2 adversary that achieves advantage $\epsilon(k)$ when attacking MPub^{hy}. Suppose $\mathcal{A}$ has running time $t(k)$, makes at most q_D decryption queries, and makes at most q_{H_3}, q_{H_4} queries to the hash functions H_3, H_4 respectively. Then there is an IND-CPA adversary $\mathcal{B}$ against MPub with running time $t_1(k)$ and advantage $\epsilon_1(k)$ where:*

$$\epsilon_1(k) \geq FO_{adv}(\epsilon(k), q_{H_4}, q_{H_3}, q_{H_D}) = \frac{1}{2(q_{H_4}+q_{H_3})}[(\epsilon(k)+1)(1-2/p)^{q_D}-1]$$

$$t_1(k) \leq FO_{time}(t(k), q_{H_4}, q_{H_3}) = t(k) + O((q_{H_4}+q_{H_3})n).$$

Here p is the size of the groups $\mathbb{G}_1, \mathbb{G}_2$ and n is the length of σ.

We now prove Theorem 1 in two steps. We first show that an IND-ID-CCA2 attack on IBE-SK can be converted to an IND-CCA2 attack on MPub^{hy}. With the result of an IND-CCA2 attack on MPub^{hy} can be converted to an IND-CPA attack on MPub [14], We then show that MPub is secure against IND-CPA if the q-BSDH assumption holds.

Lemma 1. *Let $\mathcal{A}$ be an IND-ID-CCA2 adversary that has advantage $\epsilon(k)$ against IBE-SK. Suppose $\mathcal{A}$ makes at most $q_E > 0$ private key generation queries and at most q_D decryption queries. Then there is an IND-CCA2 adversary $\mathcal{B}$ that has advantage at least $\frac{\epsilon(k)}{e(1+q_E+q_D)}$ against MPub^{hy}. Its running time is $O(time(\mathcal{A}_{Fir}))$.*

Proof. We construct an IND-CCA2 adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\frac{\epsilon(k)}{e(1+q_E+q_D)}$ against MPub^{hy}. The game between the challenger and the adversary $\mathcal{B}$ starts with the challenger first generating the public key $K_{pub} = \langle p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P^0_{pub}, P^1_{pub}, \cdots, P^q_{pub}, H_2, H_3, H_4, Q^0_{ID}, Q^1_{ID}, \cdots, Q^{q-1}_{ID}\rangle$, where $Q^i_{ID} = cP^i_{pub}$ for some $c \in \mathbb{Z}^*_p \backslash \{-s\}$. The challenger gives K_{pub} to algorithm $\mathcal{B}$.

Algorithm $\mathcal{B}$ mounts an IND-CCA2 attack on the key K_{pub} with the help of algorithm $\mathcal{A}$. Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup: Algorithm $\mathcal{B}$ gives $\mathcal{A}$ the IBE-SK system master public key $p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n$, $P^0_{pub}, P^1_{pub}, \cdots, P^q_{pub}, H_2, H_3, H_4$ that are taken from K_{pub}, and H_1 which is a random oracle controlled by $\mathcal{B}$ as described below.

H_1-queries: At any time algorithm $\mathcal{A}$ can query the random oracle H_1. To respond to these queries algorithm $\mathcal{B}$ maintains a list of tuples $\langle ID_j, Q^0_j, Q^1_j, \cdots, Q^{q-1}_j, b_j, coin_j\rangle$ as explained below. We refer to this list as the H^{list}_1 . The list is initially empty. When $\mathcal{A}$ queries the oracle H_1 at a point ID_i algorithm $\mathcal{B}$ responds as follows:

1. If the query ID_i already appears on the H^{list}_1 in a tuple $\langle ID_i, Q^0_i, Q^1_i, \cdots, Q^{q-1}_i, b_i, coin_i\rangle$, then algorithm $\mathcal{B}$ responds with $H_1(ID_i, P^k_{pub}) = Q^k_j \in \mathbb{G}^*_1, k = 0, 1, \cdots, q-1$.
2. Otherwise, $\mathcal{B}$ generates a random $coin \in \{0, 1\}$ so that $\mathsf{Pr}[coin = 0] = \delta$ for some δ that will be determined later.

3. Algorithm $\mathcal{B}$ picks a random $b_i \in \mathbb{Z}_p^*$ and computes Q_i^k as follows:
   If $coin = 0$ compute

$$Q_i^k = b_i P_{pub}^k - P_{pub}^{k+1} \in \mathbb{G}_p^*, \quad k = 0, 1, \cdots, q-1.$$

   If $coin = 1$ compute

$$Q_i^k = b_i (Q_{ID}^k + P_{pub}^{k+1}) - P_{pub}^{k+1} \in \mathbb{G}_1^*, \quad k = 0, 1, \cdots, q-1.$$

4. Algorithm $\mathcal{B}$ adds the tuple

$$\langle ID_i, Q_i^0, Q_i^1, \cdots, Q_i^{q-1}, b_i, coin_i \rangle$$

   to the H_1^{list} and responds to $\mathcal{A}$ with $Q_i^0, Q_i^1, \cdots, Q_i^{q-1}$. Note that in either way Q_i^k is uniform in $\mathbb{G}_1^*$ and is independent of $\mathcal{A}$'s current view as required.

Phase 1

Key genaration queries. Let $\{ID_{i+1}, ID_{i+2}, \cdots, ID_{i+j}\}$ $(j \leq q)$ be a private key generation queries issued by algorithm $\mathcal{A}$. Algorithm $\mathcal{B}$ responds to the queries as follows:

1. For each ID_k $(k = i+1, \cdots, i+j)$, run the above algorithm for responding to H_1-queries to obtain $Q_k^0, Q_k^1, \cdots, Q_k^{q-1}$. Let

$$\langle ID_k, Q_k^0, Q_k^1, \cdots, Q_k^{q-1}, b_k, coin_k \rangle$$

   be the corresponding tuple on the H_1^{list}. If $coin_k = 1$ then $\mathcal{B}$ reports failure and terminates. The attack on MPub^{hy} failed.
2. We know $coin_k = 0$, hence

$$Q_k^0 = b_k P_{pub}^0 - P_{pub}^1 = (b_k - s)P \qquad (k = i+1, \cdots, i+j)$$

   for all ID_k. Define

$$d_{i,j} = \frac{1}{b_{i+1} \cdots b_{i+j}} P.$$

   Observe that

$$d_{i,j} = \frac{1}{b_{i+1} - s + s} \frac{1}{b_{i+2} - s + s} \cdots \frac{1}{b_{i+j} - s + s} P$$

   We know that $d_{i,j}$ is the private key associated to the multiple public key $\{ID_{i+1}, ID_{i+2}, \cdots, ID_{i+j}\}$. Give $d_{i,j}$ to algorithm $\mathcal{A}$.

Decryption queries. Let $\langle ID_i, C_i \rangle$ be a decryption query issued by algorithm $\mathcal{A}$. Let $C_i = \langle U_{i1}, U_{i2}, \cdots, U_{iq}, V_i, W_i \rangle$. Algorithm $\mathcal{B}$ responds to this query as follows:

1. Run the above algorithm for responding to H_1-queries to obtain $Q_i^0, Q_i^1, \cdots, Q_i^{q-1}$ such that $H_1(ID_i, P_{pub}^k) = Q_i^k \in \mathbb{G}_1^*$. Let

$$\langle ID_i, Q_i^0, Q_i^1, \cdots, Q_i^{q-1}, b_i, coin_i \rangle$$

   be the corresponding tuple on the H_1^{list}.

2. Suppose $coin_i = 0$. In this case run the algorithm for responding to private key queries to obtain the private key for the public key ID_i. Then use the private key to respond to the decryption query.
3. Suppose $coin_i = 1$. Then

$$Q_i^k = b_i(Q_{ID}^k + P_{pub}^{k+1}) - P_{pub}^{k+1} = \big(b_i(c+s) - s\big)P_{pub}^k.$$

Recall that $U_i \in \mathbb{G}_1$ and

$$C_i = \langle U_{i1}, U_{i2}, \cdots, U_{iq}, V_i, W_i\rangle = \langle U_i, sU_i, \cdots, s^{q-1}U_i, V_i, W_i\rangle.$$

Set

$$C_i' = \langle \frac{1}{b_i}U_i, \frac{1}{b_i}sU_i, \cdots, \frac{1}{b_i}s^{q-1}U_i, V_i, W_i\rangle.$$

Let $d_i = \frac{1}{b_i(c+s)}P = \frac{1}{b_i(c+s)-s+s}P$ be the (unknown) IBE-SK private key corresponding to ID_i. Then the IBE-SK decryption of C_i using d_i is the same as the MPub^{hy} decryption of C_i' using d_{ID}. To see this, observe that:

$$b_i \cdot d_i = b_i \cdot \frac{1}{b_i(c+s)}P = \frac{1}{c+s}P = d_{ID},$$

$$\hat{e}(U_i, d_i) = \hat{e}(U_i, \frac{1}{b_i}d_{ID}) = \hat{e}(\frac{1}{b_i}U_i, d_{ID}).$$

Relay the decryption query $\langle C_i'\rangle$ to the challenger and relay the challenger's response back to $\mathcal{A}$.

Challenge: Once algorithm $\mathcal{A}$ decides that **Phase 1** is over, it outputs a public key ID_{ch} and two messages m_0, m_1 on which it wishes to be challenged. Algorithm $\mathcal{B}$ responds as follows:

1. Algorithm $\mathcal{B}$ gives the challenger m_0, m_1 as the messages that it wishes to be challenged on. The challenger responds with a MPub^{hy} ciphertext such that C is the encryption of m_c for a random $c \in \{0, 1\}$.
2. Next, $\mathcal{B}$ runs the algorithm for responding to H_1-queries to obtain a $Q_{ch}^0, Q_{ch}^1, \cdots, Q_{ch}^{q-1}$ such that $H_1(ID_{ch}, P_{pub}^k) = Q_{ch}^k \in \mathbb{G}_1^*$. Let $\langle ID_{ch}, Q_{ch}^0, Q_{ch}^1, \cdots, Q_{ch}^{q-1}, b, coin\rangle$ be the corresponding tuple on the H_1^{list}. If $coin = 0$ then $\mathcal{B}$ reports failure and terminates. The attack on MPub failed.
3. We know $coin = 1$ and therefore $Q_{ch}^k = b(Q_{ID}^k + P_{pub}^{k+1}) - P_{pub}^{k+1}$. Recall that when $C = \langle U, sU, \cdots, s^{q-1}U, V, W\rangle$. Set $C' = \langle bU, bsU, \cdots, bs^{q-1}U, V, W\rangle$. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with the challenge C'. C' is a ciphertext of m_c in IBE-SK under the public key ID_{ch} as required.

$$\begin{aligned} C' &= \langle bU, bsU, \cdots, bs^{q-1}U, V, W\rangle \\ &= \langle br(Q_{ID}^0 + P_{pub}^1), bsr(Q_{ID}^0 + P_{pub}^1), , \cdots, bs^{q-1}r(Q_{ID}^0 + P_{pub}^1), V, W\rangle \quad (1) \\ &= \langle r(Q_{ch}^0 + P_{pub}^1), sr(Q_{ch}^0 + P_{pub}^1), \cdots, s^{q-1}r(Q_{ch}^0 + P_{pub}^1), V, W\rangle \quad (2) \end{aligned}$$

The conversion from (1) to (2) is based on:

$$bs^k r(Q_{ID}^0 + P_{pub}^1) = s^k r(b(Q_{ID}^0 + P_{pub}^1) - P_{pub}^1 + P_{pub}^1) = s^k r(Q_{ch}^0 + P_{pub}^1).$$

Phase 2
Key generation queries. Same as phase 1.
Decryption queries. Same as phase 1. However, if the resulting decryption query relayed to the challenger is equal to the challenge ciphertext $C = \langle U, sU, \cdots, s^{q-1}U, V, W\rangle$ then $\mathcal{B}$ reports failure and terminates. The attack on MPub^{hy} failed.

Guess: Eventually algorithm $\mathcal{A}$ outputs a guess c' for c. Algorithm $\mathcal{B}$ outputs c' as its guess for c.

Claim 1. *If algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Furthermore, if $\mathcal{B}$ does not abort then $|\mathsf{Pr}[c = c'] - \frac{1}{2}| \geq \epsilon$. The probability is over the random bits used by $\mathcal{A}, \mathcal{B}$ and the challenger.*

Proof of Claim. [7] □

It remains to bound the probability that algorithm $\mathcal{B}$ aborts during the simulation. The algorithm could abort for three reasons: (1) a bad private key query from $\mathcal{A}$ during phases 1 or 2, (2) $\mathcal{A}$ chooses a bad ID_{ch} to be challenged on, or (3) a bad decryption query from $\mathcal{A}$ during phase 2. We define three corresponding events:

ε_1 is the event that $\mathcal{A}$ issues a private key query during phase 1 or 2 that causes algorithm $\mathcal{B}$ to abort; ε_2 is the event that $\mathcal{A}$ chooses a public key ID_{ch} to be challenged on that causes algorithm $\mathcal{B}$ to abort; ε_3 is the event that during phase 2 of the simulation Algorithm $\mathcal{A}$ issues a decryption query $\langle ID_i, C_i\rangle$ so that the decryption query that $\mathcal{B}$ would relay to the MPub challenger is equal to C. Recall that $C = \langle U, sU, \cdots, s^{q-1}U, V, W\rangle$ is the challenge ciphertext from the MPub^{hy} challenger.

Claim 2. $\mathsf{Pr}[\neg\varepsilon_1 \wedge \neg\varepsilon_2 \wedge \neg\varepsilon_3] \geq \delta^{q_E+q_D}(1-\delta)$.

Proof of Claim. [7] □

To conclude the proof of Lemma 1 it remains to optimize the choice of δ. Since $\mathsf{Pr}[\neg\varepsilon_1 \wedge \neg\varepsilon_2 \wedge \neg\varepsilon_3] \geq \delta^{q_E+q_D}(1-\delta)$ the success probability is maximized at $\delta_{opt} = 1 - 1/(1+q_E+q_D)$. Using δ_{opt}, the probability that $\mathcal{B}$ does not abort is at least $\frac{1}{e(1+q_E+q_D)}$. This shows that $\mathcal{B}$'s advantage is at least $\epsilon(k)/e(1+q_E+q_D)$ as required.

Lemma 2. *Let H_2 be a random oracle from $\mathbb{G}_2$ to $\{0,1\}^n$. Let A be an IND-CPA adversary that has advantage $\epsilon(k)$ against MPub. Suppose $\mathcal{A}$ makes a total of $q_{H_2} > 0$ queries to H_2. Then there is an algorithm $\mathcal{B}$ that solves the q-BSDH problem for $\mathcal{G}$ with advantage at least $2\epsilon(k)/q_{H_2}$ and a running time $O(time(\mathcal{A}))$.*

Proof. Algorithm $\mathcal{B}$ is given as input the q-BSDH parameters $\langle p, \mathbb{G}_1.\mathbb{G}_2, \hat{e}\rangle$ produced by $\mathcal{G}$ and a random instance $\langle P, sP, s^2P, \cdots, s^qP, bP\rangle = \langle P, P_1^1, P_1^2, \cdots, P_1^q, P_2\rangle$ of the q-BSDH problem for these parameters, i.e. P is random in $\mathbb{G}_1$ and s, b are random in $\mathbb{Z}_p^*$ where p is the order of $\mathbb{G}_1, \mathbb{G}_2$. For any $c \in \mathbb{Z}_p^* \backslash \{-s\}$, let

$\hat{e}(\frac{1}{c+s}P, bP)$ be the solution to this q-BSDH problem. Algorithm $\mathcal{B}$ finds D by interacting with $\mathcal{A}$ as follows:

Setup: Algorithm $\mathcal{B}$ creates the MPub public key $K_{pub} = \langle p, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P^0_{pub}, P^1_{pub}, \cdots, P^q_{pub}, H_2, Q^0_{ID}, Q^1_{ID}, \cdots, Q^{q-1}_{ID} \rangle$ by setting $P^k_{pub} = c^k P$ and $Q^k_{ID} = c^k P_1 = sP^k_{pub}$. Here H_2 is a random oracle controlled by $\mathcal{B}$ as described below. Algorithm $\mathcal{B}$ gives $\mathcal{A}$ the MPub public key K_{pub}. Observe that the (unknown) private key associated to K_{pub} is $d_{ID} = \frac{1}{s+c}P = \frac{1}{c+s}P$.

H_2-queries: At any time algorithm $\mathcal{A}$ may issue queries to the random oracle H_2. To respond to these queries $\mathcal{B}$ maintains a list of tuples called the H_2^{list}. Each entry in the list is a tuple of the form $\langle X_j, H_j \rangle$. Initially the list is empty. To respond to query X_i algorithm $\mathcal{B}$ does the following:

1. If the query X_i already appears on the H_2^{list} in a tuple $\langle X_j, H_j \rangle$ then responds with $H_2(X_i) = H_i$.
2. Otherwise, $\mathcal{B}$ just picks a random string $H_i \in \{0,1\}^n$ and adds the tuple $\langle X_j, H_j \rangle$ to the H_2^{list}. It responds to $\mathcal{A}$ with $H_2(X_i) = H_i$.

Challenge: Algorithm $\mathcal{A}$ outputs two messages m_0, m_1 on which it wishes to be challenged. Algorithm $\mathcal{B}$ picks a random strings R and defines C to be the ciphertext $C = \langle P_2, cP_2, \cdots, c^{q-1}P_2, R \rangle$. Algorithm $\mathcal{A}$ gives C as the challenge to $\mathcal{A}$. Observe that, by definition, the decryption of C is $R \oplus H_2(\hat{e}(P_2, d_{ID})) = R \oplus H_2(D)$.

Guess: Algorithm $\mathcal{A}$ outputs its guess $c' \in \{0,1\}$. At this point $\mathcal{B}$ picks a random tuple $\langle X_j, H_j \rangle$ from the H_2^{list} and outputs X_j as the solution to the given instance of q-BSDH.

Algorithm $\mathcal{B}$ is simulating a real attack environment for algorithm $\mathcal{A}$ (it simulates the challenger and the oracle for H_2). We show that algorithm $\mathcal{B}$ outputs the correct answer D with probability at least $2\epsilon/q_{H_2}$ as required. Let $\mathcal{H}$ be the event that algorithm $\mathcal{A}$ issues a query for $H_2(D)$ at some point during the simulation above (this implies that at the end of the simulation D appears in some tuple on the H_2^{list}). We show that $\mathsf{Pr}[\mathcal{H}] \geq 2\epsilon$.

Claim 3. $\mathsf{Pr}[\mathcal{H}]$ *in the simulation above is equal to* $\mathsf{Pr}[\mathcal{H}]$ *in the real attack.*

Proof of Claim. [7] □

Claim 4. *In the real attack we have* $\mathsf{Pr}[\mathcal{H}] \geq 2\epsilon$.

Proof of Claim. [7] □

To complete the proof of Lemma 2 observe that by Claims 3 and 4 we know that $\mathsf{Pr}[\mathcal{H}] \geq 2\epsilon$ in the simulation above. Hence, at the end of the simulation, D appears in some tuple on the H_2^{list} with probability at least 2ϵ. It follows that $\mathcal{B}$ produces the correct answer with probability at least $2\epsilon/q_{H_2}$ as required.

Proof of Theorem 1. By Lemma 1 an IND-ID-CCA2 adversary on IBE-SK implies an IND-CCA2 adversary on MPub^{hy}. By Theorem 2 an IND-CCA2

adversary on MPub^{hy} implies an IND-CPA adversary on MPub. By Lemma 2 an IND-CPA adversary on MPub implies an algorithm for q-BSDH. Composing all these reductions gives the required bounds.

5 Conclusion

In this paper, we gave a novel IBE scheme for multiple public-key. In our IBE-SK scheme, we can set a single private key that maps multiple public keys (ID's), such that the private key can decrypt all ciphertexts generated from different public keys (ID's).

Acknowledgement. The authors would like to thank the anonymous reviewers of *Pairing 2007* for their helpful comments on this work.

References

1. 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)
2. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM conference on Computers and Communication Security, pp. 62–73 (1993)
3. Benaloh, J., de Mare, M.: One-way accumulators: a decentralized al- ternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
4. Boneh, D., Boyen, X.: Efficient selective-id secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
7. 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)
8. 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)
9. Cheon, J.H.: Security analysis of the Strong Diffie-Hellman problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)
10. Canetti, R., Halevi, S., Katz, J.: A forward-secure public key encryption scheme. In: Biham, E. (ed.) Advances in Cryptology – EUROCRPYT 2003. LNCS, vol. 2656, Springer, Heidelberg (2003)

11. Chen, L., Harrison, K., Smart, N.P., Soldera, D.: Applications of Multiple Trust Authorities in pairing based Cryptosystems. In: Davida, G.I., Frankel, Y., Rees, O. (eds.) InfraSec 2002. LNCS, vol. 2437, pp. 260–275. Springer, Heidelberg (2002)
12. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and applications to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
13. Chatterjee, S., Sarkar, P.: Constant Size Ciphertext HIBE in the Augmented Selective-ID Model and its Extensions http://eprint.iacr.org/2007/084
14. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
15. Gangishetti, R., Gorantla, M.C., Das, M.L., Saxena, A., Gulati, V.P.: An Efficient Secure Key Issuing Protocol in ID-Based Cryptosystems. In: Proceedings of the International Conference on Information Technology: Coding and Computin (ITCC'05), vol. 1, pp. 674–678. IEEE Computer Society, Los Alamitos (2005)
16. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
17. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y., Zheng, Y. (eds.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
18. Horwitz, J., Lynn, B.: Towards hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
19. Kumar, K.P., Shailaja, G., Saxena, A.: Secure and Efficient Threshold Key Issuing Protocol for ID-based Cryptosystems http://eprint.iacr.org/2006/245
20. Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., Yoo, S.: Secure Key Issuing in ID-based Cryptography. In: proceedings of the Second Australian Information Security Workshop-AISW 2004, ACS Conferences in Research and Practice in Information Technology, vol. 32, pp. 69–74 (2004)
21. Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Trans E85-A(2), 481–484 (2002)
22. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Tran. on Info. Th. 39, 1639–1646 (1993)
23. Nguyen, L.: Accumulators from Bilinear Pairings and Applications. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
24. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
25. Sui, A., Chow, S.S.M., Hui, L.C.K., Yiu, S.M., Chow, K.P., Tsang, W.W., Chong, C.F., Pun, K.H., Chan, H.W.: Seperable and Anonymous Identity-Based Key Issuing without Secure Channel. In: Proc. of the 11th International Comference on Parallel and Distributed Systems (ICPADS 2005), vol. 2, pp. 275–279 (2005)
26. Waters, B.: Efficient Identity-Based Encryption without Random Oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Author Index