# Applications of Bilinear Maps in Cryptography

by

Martin Gagné

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2002

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

# Abstract

It was recently discovered by Joux [30] and Sakai, Ohgishi and Kasahara [47] that bilinear maps could be used to construct cryptographic schemes. Since then, bilinear maps have been used in applications as varied as identity-based encryption, short signatures and one-round tripartite key agreement.

This thesis explains the notion of bilinear maps and surveys the applications of bilinear maps in the three main fields of cryptography: encryption, signature and key agreement. We also show how these maps can be constructed using the Weil and Tate pairings in elliptic curves.

# Acknowledgements

First, I would like to thank my supervisor, Alfred Menezes, for the suggestion of this topic and for his great patience in assisting me in the writing of this thesis. I would also like to thanks Fiona McAlister for helping me to solve all sorts of administrative problems I encountered when writing this thesis and during my Master's degree.

Great thanks also to all my friends inside and outside of the C&O department for their support while I was writing this thesis.

I would also like to thank the Department of Combinatorics and Optimization and the Natural Science and Engineering Research Council of Canada for their financial support during my studies.

Last but certainly not least, I would like to thank my family for their love and support.

# Contents

# Chapter 1

# Introduction

Bilinear maps were first introduced in cryptography in 1993 in the form of the Weil pairing by Menezes, Okamoto and Vanstone [40] in an attack against the discrete logarithm problem in supersingular elliptic curves. This attack was extended a year later by Frey and Rück using the Tate pairing. At that time, the existence of such a bilinear map had negative connotations.

In 2000, Sakai, Ohgishi and Kasahara [47] and Joux [30] independently found that bilinear maps could be used in constructive ways to build new cryptographic schemes. These two papers were followed by many others that used bilinear maps to construct identity-based encryption schemes [5, 26, 29, 34], schemes for short signatures [5, 6], identity-based signature schemes [10, 28, 44, 47], tripartite key agreement schemes [1, 30, 47] and many other applications.

The goal of this thesis is to survey the main applications of bilinear maps and show how these bilinear maps can be implemented.

In Chapter 2, we define the concept of bilinear maps, introduce the bilinear

Diffie-Hellman problem, and relate this problem to others already used in cryptography.

In Chapter 3, 4 and 5, we survey the applications of bilinear maps in the 3 main areas of cryptography, namely encryption, signature and key agreement.

Finally, in Chapter 6, we present a way to implement bilinear maps using the Weil and Tate pairing on supersingular elliptic curves.

# Chapter 2

# Bilinear Maps and the Bilinear Diffie-Hellman Assumption

Most of the concepts in this section come from a paper by Boneh and Franklin [5].

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of equal order $n$. A function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is called a bilinear map if it satisfies the following properties:

Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$[1].

Non-Degeneracy: For a given point $Q \in G_1$, $\hat{e}(Q, R) = 1_{\mathbb{G}_2}$ for all $R \in \mathbb{G}_1$ if and only if $Q = 1_{\mathbb{G}_1}$. From that, we can find that if $P$ is a generator of $\mathbb{G}_1$, then $\hat{e}(P, P)$ is a generator for $\mathbb{G}_2$.

Also, for practical reasons, we require the following:

Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

[1]The only known bilinear maps meeting all the requirements for use in cryptography are the Weil and Tate pairings on abelian varieties. For consistency of notation with Chapter 6 on the implementation of such maps, we denote $\mathbb{G}_1$ additively and $\mathbb{G}_2$ multiplicatively.

We note that any such bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is necessarily symmetric, i.e. $\hat{e}(Q, R) = \hat{e}(R, Q)$ since, taking $r, s \in \mathbb{Z}$ such that $Q = rP$ and $R = sP$ for some generator $P$ of $\mathbb{G}_1$,

$$\hat{e}(Q, R) = \hat{e}(rP, sP) = \hat{e}(P, P)^{rs} = \hat{e}(sP, rP) = \hat{e}(R, Q).$$

The existence of such a bilinear map between groups $\mathbb{G}_1$ and $\mathbb{G}_2$ has two direct consequences on the hardness of some problems in the group $\mathbb{G}_1$.

**The MOV and FR reductions:** Menezes, Okamoto and Vanstone [40] and Frey and Rück [19] show a reduction from the discrete log problem in group $\mathbb{G}_1$ to the discrete log problem in group $\mathbb{G}_2$. The reduction goes as follows. Given an instance $P, Q$ of the discrete log problem in $\mathbb{G}_1$, where $P$ is a point of order $n$, we wish to find $x \in \mathbb{Z}$, $1 \leq x \leq n-1$, such that $Q = xP$. Let $R$ be an element of $\mathbb{G}_1$ such that $g = \hat{e}(R, P)$ has order $n$, and let $h = \hat{e}(R, Q)$. Then, by bilinearity of $\hat{e}$, we have that $h = g^x$. Thus, computing the discrete log of $h$ to the base $g$ in $\mathbb{G}_2$ gives the answer to the initial problem. Hence, the discrete log problem in $\mathbb{G}_1$ is no harder than the discrete log problem in $\mathbb{G}_2$.

**The Decision Diffie-Hellman problem is easy:** The Decision Diffie-Hellman Problem (DDH) in $\mathbb{G}_1$ can be stated as follows: given $\langle P, aP, bP, cP \rangle$ where $P \in \mathbb{G}_1$ and $a, b, c \in \mathbb{Z}$, determine if $c = ab \bmod n$. Joux and Nguyen [31] observe that

$$c = ab \bmod n \quad \Longleftrightarrow \quad \hat{e}(P, cP) = \hat{e}(aP, bP).$$

Therefore, the DDH problem is easy to solve in $\mathbb{G}_1$. (Note that the Computational Diffie-Hellman problem (CDH) can still be hard in $\mathbb{G}_1$ even if the DDH problem is easy.)

**Notation:** From here on, we use $\mathbb{G}^*$ to denote the set $\mathbb{G}^* = \mathbb{G} \setminus \{\mathcal{O}\}$ where $\mathcal{O}$ is the identity element in $\mathbb{G}$, and $\mathbb{Z}_n$ to denote the group $\{0, 1, \ldots, n-1\}$ under addition modulo $n$. We denote by $\mathbb{F}_q$ the finite field with $q$ elements.

## 2.1 Intractability Assumptions

We recall here the Computational Diffie-Hellman (CDH) assumption and introduce a variant, the Bilinear Diffie-Hellman (BDH) assumption which will be needed to prove the security of some of the schemes.

### 2.1.1 The Computational Diffie-Hellman Problem

The *Computational Diffie-Hellman Problem* in a cyclic group $\mathbb{G}$ of order $n$ is:
*Given a generator $g$ of $\mathbb{G}$ and two elements $g^a, g^b \in \mathbb{G}$ for $a$ and $b$ random in $\mathbb{Z}_n$, compute $g^{ab}$.*

We say that a randomized algorithm $\mathcal{IG}$ is a *CDH parameter generator* if:

1. $\mathcal{IG}$ takes as input a security parameter $0 < \lambda \in \mathbb{Z}$ (in unary).
2. $\mathcal{IG}$ runs in polynomial time in $\lambda$.
3. $\mathcal{IG}$ outputs the description of a group $\mathbb{G}$ of prime order $p$.

We denote by $\mathcal{IG}(1^\lambda)$ the set of all possible outcomes of $\mathcal{IG}$ on input $1^\lambda$.

For any probabilistic algorithm $\mathcal{A}$, we define the *CDH advantage of* $\mathcal{A}$ *against the CDH parameter generator* $\mathcal{IG}$ by

$$AdvCDH_{\mathcal{IG},\mathcal{A}}(\lambda) = \Pr\left[h = g^{ab} \middle| \begin{array}{c} \mathbb{G} \leftarrow \mathcal{IG}(1^\lambda), |\mathbb{G}| = p; g \leftarrow \mathbb{G}^*; \\ a, b \leftarrow \mathbb{Z}_p; h \leftarrow \mathcal{A}(g^a, g^b) \end{array}\right].$$

Where $\leftarrow$ means 'is a random element of'.

The *CDH assumption* for $\mathcal{IG}$ is that :
*For every probabilistic polynomial-time algorithm* $\mathcal{A}$*, the function* $AdvCDH_{\mathcal{IG},\mathcal{A}}(\lambda)$ *is negligible in* $\lambda$. [2]

### 2.1.2 The Bilinear Diffie-Hellman Problem

The *Bilinear Diffie-Hellman Problem* in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of equal order $n$ and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear map satisfying the properties described previously, is:
*Given a generator* $P$ *of* $\mathbb{G}_1$ *and three elements* $aP, bP, cP \in \mathbb{G}_1$ *for* $a$*,* $b$ *and* $c$ *random in* $\mathbb{Z}_n$*, compute* $\hat{e}(P, P)^{abc}$.

We say that a randomized algorithm $\mathcal{IG}$ is a *BDH parameter generator* if:

1. $\mathcal{IG}$ takes as input a security parameter $0 < \lambda \in \mathbb{Z}$ (in unary).

[2] Recall that a function $f(\lambda)$ is said to be negligible in $\lambda$ if $f(\lambda) \leq 1/\rho(\lambda)$ for all polynomials $\rho \in \mathbb{Z}[x]$.

2. $\mathcal{IG}$ runs in polynomial time in $\lambda$.

3. $\mathcal{IG}$ outputs the description of a two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order $p$ and the description of a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

We denote by $\mathcal{IG}(1^\lambda)$ the set of all possible outcomes if $\mathcal{IG}$ on input $1^\lambda$.

For any probabilistic algorithm $\mathcal{A}$, we define the *BDH advantage of $\mathcal{A}$ against the BDH parameter generator $\mathcal{IG}$* by

$$AdvBDH_{\mathcal{IG},\mathcal{A}}(\lambda) = \Pr\left[h = \hat{e}(P,P)^{abc} \middle| \begin{array}{c} \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle \leftarrow \mathcal{IG}(1^\lambda), |\mathbb{G}_1| = |\mathbb{G}_2| = p; \\ P \leftarrow \mathbb{G}_1^*; a, b, c \leftarrow \mathbb{Z}_p; h \leftarrow \mathcal{A}(P, aP, bP, cP) \end{array}\right].$$

The *BDH assumption* for $\mathcal{IG}$ is that :
*For every probabilistic polynomial-time algorithm $\mathcal{A}$, the function $AdvBDH_{\mathcal{IG},\mathcal{A}}(\lambda)$ is negligible in $\lambda$.*

### 2.1.3 Relation between the CDH and BDH

It is easy to show that the BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ [3] is no harder than the CDH problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$. If we have an algorithm to solve the CDH problem in $\mathbb{G}_1$ with respect to $P$, then given $aP, bP, cP \in \mathbb{G}_1$, we compute $abP$ using the CDH algorithm. Then $\hat{e}(abP, cP) = \hat{e}(P,P)^{abc}$ is the solution to the BDH problem. On the other hand, if we have an algorithm to solve the CDH problem in $\mathbb{G}_2$ with respect to $g = \hat{e}(P,P)$, then, given $aP, bP, cP \in \mathbb{G}_1$, we compute $g^{ab} = \hat{e}(aP, bP)$ and

[3] From now on, $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order.

$g^c = \hat{e}(cP, P)$, and then we use the CDH algorithm to compute $g^{abc} = \hat{e}(P, P)^{abc}$, the solution to the BDH problem. The problem of finding if the BDH problem is polynomially equivalent to the CDH problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$ is still open.

It is interesting to note that the isomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$ induced by the bilinear map are one-way functions assuming that the DDH problem is hard in $\mathbb{G}_2$. For a point $Q \in \mathbb{G}_1^*$, define the isomorphism $f_Q : \mathbb{G}_1 \to \mathbb{G}_2$ by $f_Q(R) = \hat{e}(R, Q)$ [4]. We show that an efficient algorithm to invert any $f_Q$ would imply an efficient algorithm to solve the DDH problem in $\mathbb{G}_2$ (and for the only known implementation of a bilinear map, the DDH is intractable in $\mathbb{G}_2$).

**Proposition 2.1.1** *Suppose there is a t-time algorithm $\mathcal{A}$ which inverts $f_Q$ for some $Q \in \mathbb{G}_1^*$ with probability $\varepsilon$. Then there is an efficient algorithm $\mathcal{B}$ that solves the DDH problem in $\mathbb{G}_2$ with respect to $g = \hat{e}(P, P)$ in time $4t+\tau$ and with probability $\varepsilon^4$, where $P$ is any generator of $\mathbb{G}_1$ and $\tau$ is the time required to compute the function $\hat{e}$.*

**Proof** Given $g^a, g^b, g^c \in \mathbb{G}_2$, compute $U = f_Q^{-1}(g^a)$, $V = f_Q^{-1}(g^b)$, $W = f_Q^{-1}(g^c)$ and $Y = f_Q^{-1}(g)$. Say $Q = xP$. Note that $U = f_Q^{-1}(g^a)$ implies that $\hat{e}(U, Q) = g^a$, so $\hat{e}(U, xP) = \hat{e}(U, P)^x = \hat{e}(xU, P) = g^a$ so $U = ax^{-1}P$. [5] Similarly, $V = bx^{-1}P$, $W = cx^{-1}P$ and $Y = x^{-1}P$. Therefore, $\hat{e}(U, V) = g^{abx^{-2}}$ and $\hat{e}(W, Y) = g^{cx^{-2}}$. Thus, $g^c = g^{ab}$ if and only if $\hat{e}(U, V) = \hat{e}(W, Y)$. It is easy to see that we need 4 applications of algorithm $\mathcal{A}$ and this new algorithm is always successful if $\mathcal{A}$ gives the correct answer each time. ∎

[4] It is a homomorphism because $\hat{e}$ is bilinear and is is onto because $\hat{e}$ is non-degenerate and $Q \in \mathbb{G}_1$ is a generator of $\mathbb{G}_1$ since $\mathbb{G}_1$ has prime order.

[5] By $x^{-1}$, we mean $x^{-1} \bmod p$.

# Chapter 3

# Encryption Schemes

In this chapter, we present how bilinear maps can be used to construct encryption schemes.

The most significant result in this field is probably the identity-based encryption scheme by Boneh and Franklin [5]. We present this scheme with complete proof of security. We also present results by Lynn [34], Horwitz and Lynn [29] and Gentry and Silverberg [26] who show how the scheme can be modified to include message authentication, or to allow a hierarchy of private key generators.

We also present schemes by Boneh and Franklin [5] and Verheul [51] that use bilinear maps to build encryption schemes with key escrow.

## 3.1 Identity Based Encryption

In traditional public key encryption, public keys are usually generated at random. Then, if Alice wants to send an encrypted message to Bob, she must first obtain

Bob's key from a public directory together with a certificate proving the authenticity of the key. The motivation for identity-based encryption is to eliminate the need for directory and certificates by using the identity of the receiver as the public key (for example, if Alice wants to send an e-mail to Bob at bob@yahoo.com, then she simply encrypts the message using the string "bob@yahoo.com" as the public key).

The idea of identity-based cryptography was first formulated by Shamir [48] in 1984. However, practical schemes for identity-based encryption were not found until recently by Boneh and Franklin [5] and Cocks [13] in 2001. Cocks's scheme is based on the "Quadratic Residuosity Problem" and even though it is reasonably fast, it has a significant message expansion factor. We present here the scheme by Boneh and Franklin, which is based on the BDH problem. This section closely follows [5].

### Definitions

### Identity-Based Encryption Scheme

An *identity-based encryption scheme* consists of four randomized algorithms: Setup, Extract, Encrypt and Decrypt.

Setup: takes as input a security parameter and outputs params and master-key. The system parameters params must include a description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$. The system parameters will be publicly known, while the master-key is known only to the private key generator (PKG).

Extract: takes as input the system parameters params, the master-key and an arbitrary string $\mathsf{ID} \in \{0,1\}^*$ and outputs the private key $d_{\mathsf{ID}}$ corresponding to the

public key ID.

Encrypt: takes as input the system parameters params, a public key ID and a plaintext $M$ and outputs the corresponding ciphertext.

Decrypt: takes as input the system parameters, params, a private key $d_{\mathsf{ID}}$, and a ciphertext $C$ and outputs the corresponding plaintext.

These algorithms are required to satisfy the standard consistency constraints, namely if params is produced by the Setup algorithm, ID is a public key and $d$ is the corresponding private key generated by the algorithm Extract, then

$$\forall M \in \mathcal{M},\ \mathsf{Decrypt}(\mathsf{params}, d, \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}, M)) = M.$$

## Security

The standard notion of security for public key encryption schemes is that of chosen ciphertext security (IND-CCA) defined by Rackoff and Simon in [45]. This definition captures the notion that an adversary should not be able to obtain any information about a ciphertext even if he is given the decryption of any other ciphertext of his choice. However, in our setting, the adversary may also be able to obtain the private key corresponding to some IDs of his choice, other than the one on which he is being tested. The system should remain secure against such an attack. Therefore, the definition of security must be strengthened a little to allow the adversary to obtain the private key corresponding to any IDs except the one on which he is being tested.

The notion of semantic security against adaptive chosen ciphertext attack for an identity-based encryption scheme (IND-ID-CCA) is defined through the following game:

**Setup:** The challenger takes a security parameter $\lambda$ and runs the Setup algorithm. He returns to the adversary the public system parameters params and keeps the master-key to itself.

**Phase 1:** The adversary issues queries $q_1, \ldots, q_m$ where each query is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $d_i$ corresponding to the public key $\mathsf{ID}_i$ and sends it to the adversary.
- Decryption query $\langle \mathsf{ID}_i, C_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $d_i$ corresponding to the public key $\mathsf{ID}_i$, uses this private key to decrypt the ciphertext $C_i$ and returns the resulting plaintext to the adversary.

**Challenge:** The adversary outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and a public key ID on which he wishes to be tested. The only constraint is that ID must not have appeared in any extraction query in Phase 1. The challenger picks a random bit $c \in \{0, 1\}$ and sends $C = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}, M_c)$ as the challenge to the adversary.

**Phase 2:** The adversary issues queries $q_{m+1}, \ldots, q_n$ where each query is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$ where $\mathsf{ID}_i \neq \mathsf{ID}$. The challenger responds as in Phase 1.

– Decryption query $\langle \mathsf{ID}_i, C_i \rangle \neq \langle \mathsf{ID}, C \rangle$. The challenger responds as in Phase 1.

**Guess:** The adversary outputs a guess $c' \in \{0, 1\}$. The adversary wins the game if $c' = c$.

Such an adversary is called an IND-ID-CCA attacker. The advantage of an IND-ID-CCA attacker $\mathcal{A}$ against the scheme is defined to be:

$$Adv_{\mathcal{A}}(\lambda) = \left| \Pr[c = c'] - \frac{1}{2} \right|$$

where the probability is over the random choices made by the challenger and the adversary. We say that an identity-based encryption scheme is semantically secure against adaptive chosen ciphertext attack (IND-ID-CCA) if no polynomially bounded adversary (in $\lambda$) has non-negligible advantage (in $\lambda$) in the game described above.

To prove that the identity-based encryption scheme is IND-ID-CCA, we need a weaker notion of security called one-way encryption [21]. One-way-encryption (OWE) is defined for public-key encryption schemes as follows: the adversary is given a random public key $K_{pub}$ and a ciphertext $C$ which is the encryption of a random plaintext $M$ using $K_{pub}$. The goal of the adversary is to recover the plaintext $M$. We say that a public-key encryption scheme is a one-way encryption scheme if no polynomially bounded adversary has a non-negligible probability of recovering the plaintext.

Again, this definition needs to be strengthened to model the idea that the ad-

versary may obtain some private keys. The notion of one-way identity-based encryption (ID-OWE) is defined through the following game:

**Setup:** The challenger takes a security parameter $\lambda$ and runs the Setup algorithm. He returns to the adversary the public system parameters params and keeps the master-key to itself.

**Phase 1:** The adversary issues private key extraction queries $\mathsf{ID}_1, \ldots, \mathsf{ID}_m$. The challenger responds by running algorithm Extract to generate the private key $d_i$ corresponding to the public key $\mathsf{ID}_i$ and sends it to the adversary.

**Challenge:** The adversary outputs a public key ID different from $\mathsf{ID}_1, \ldots, \mathsf{ID}_m$ on which he wishes to be challenged. The challenger picks a random plaintext $M \in \mathcal{M}$, encrypts it using ID as a public key and sends the resulting ciphertext to the adversary.

**Phase 2:** The adversary issues more private key extraction queries $\mathsf{ID}_{m+1}, \ldots, \mathsf{ID}_n$ different from ID. The challenger responds as in Phase 1.

**Guess:** The adversary outputs a guess $M' \in \mathcal{M}$. The adversary wins if $M' = M$.

Such an adversary is called an ID-OWE attacker. The advantage of an ID-OWE attacker against the scheme is defined to be $\Pr[M' = M]$, where the probability is over the random choices made by the challenger and the adversary. We say that an identity-based encryption scheme is a one-way identity-based encryption scheme (ID-OWE) if no polynomially bounded adversary (in $\lambda$) has non-negligible advantage (in $\lambda$) in the game described above.

### 3.1.1 The Basic Scheme

We first present a basic identity-based encryption scheme called BasicIdent and prove it is ID-OWE in the random oracle model [1].

Setup: Given a BDH parameter generator $\mathcal{IG}$ and a security parameter $\lambda$,

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in G_1^*$.

Step 2: Pick a random $s \in \mathbb{Z}_p^*$ and compute $P_{pub} = sP$.

Step 3: Pick cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some positive integer $n$.

The plaintext space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $C = \mathbb{G}_1^* \times \{0,1\}^n$. The public system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, H_1, H_2 \rangle$. The master-key is $s \in \mathbb{Z}_p^*$.

Extract: Given a string $\mathsf{ID} \in \{0,1\}^*$, the master-key $s$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, H_1, H_2 \rangle$,

Step 1: Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$.

Step 2: Compute $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ and return $d_{\mathsf{ID}}$.

Encrypt: Given a plaintext $M \in \mathcal{M}$, a public key ID and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, H_1, H_2 \rangle$,

[1] Recall that a random oracle is an idealized random function which, given any input in its domain, returns a random and independent element of its range (but always returns the same element for a given input).

Step 1: Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$.

Step 2: Compute $g = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2^*$.

Step 3: Pick a random $r \in \mathbb{Z}_p^*$.

Step 4: Compute the ciphertext $C = \langle rP, M \oplus H_2(g^r) \rangle$ and return $C$.

Decrypt: Given a ciphertext $\langle U, V \rangle$, a private key $d_{\mathsf{ID}} \in G_1^*$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, H_1, H_2 \rangle$,

Step 1: Compute $g' = \hat{e}(d_{\mathsf{ID}}, U)$.

Step 2: Compute $M = V \oplus H_2(g')$. Return $M$.

We have that

$$\hat{e}(d_{\mathsf{ID}}, U) = \hat{e}(sQ_{\mathsf{ID}}, rP) = \hat{e}(Q_{\mathsf{ID}}, P)^{rs} = \hat{e}(Q_{\mathsf{ID}}, P_{pub})^r = g^r.$$

Therefore, the message returned by the decryption algorithm is

$$M \oplus H_2(g^r) \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = M \oplus H_2(g^r) \oplus H_2(g^r) = M$$

which proves that the scheme is consistent.

## Security

We now show that BasicIdent is a one-way identity-based encryption scheme (ID-OWE) assuming that the BDH assumption holds for $\mathcal{IG}$.

**Theorem 3.1.1** *Let the hash functions $H_1, H_2$ be random oracles. Suppose there is an* ID-OWE *attacker $\mathcal{A}$ that has advantage $\varepsilon$ against the scheme* BasicIdent *which makes at most $q_E > 0$ private key extraction queries and $q_{H_2} > 0$ hash queries to $H_2$. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem in $\mathcal{IG}$ with advantage at least*

$$AdvBDH_{\mathcal{IG},\mathcal{B}}(\lambda) \geq \frac{\varepsilon}{e(1+q_E)\cdot q_{H_2}} - \frac{1}{q_{H_2}\cdot 2^n}$$

*where $e \approx 2.71$ is the base of the natural logarithm. The running time of $\mathcal{B}$ is $O(time(\mathcal{A}))$.*

Note that the values $\varepsilon, q_E, q_{H_2}$ are actually functions of $\lambda$ (because their value certainly change depending on the size of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$), but we drop the parameter $\lambda$ to simplify the notation. Note also that the second term in the bound on $AdvBDH_{\mathcal{IG},\mathcal{B}}(\lambda)$ is negligible in $\lambda$ only if $1/2^n$ is negligible in $\lambda$.

The following public key encryption scheme, that we call BasicPub, is used to prove the theorem. It is described by the three following algorithms:

Setup: Given a BDH parameter generator $\mathcal{IG}$ and a security parameter $\lambda$,

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in G_1^*$.

Step 2: Pick a random $Q_{\mathsf{ID}} \in \mathbb{G}_1$ (so $Q_{\mathsf{ID}}$ is in the group generated by $P$).

Step 3: Pick a random $s \in \mathbb{Z}_p^*$ and compute $P_{pub} = sP$ and $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$.

Step 4: Pick a cryptographic hash function $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some positive integer $n$.

The public key is $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$, the private key is $d_{\mathsf{ID}}$. The plaintext space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$.

Encrypt: Given a plaintext $M \in \mathcal{M}$ and a public key $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$,

Step 1: Compute $g = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2^*$.

Step 2: Pick a random $r \in \mathbb{Z}_p^*$.

Step 3: Compute the ciphertext $C = \langle rP, M \oplus H_2(g^r) \rangle$ and return $C$.

Decrypt: Given a ciphertext $C = \langle U, V \rangle \in \mathcal{C}$, a public key $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$ and the private key $d_{\mathsf{ID}}$,

Step 1: Compute $g' = \hat{e}(d_{\mathsf{ID}}, U)$.

Step 2: Compute $M = V \oplus H_2(g')$. Return $M$.

This completes the description of BasicPub. We now prove Theorem 3.1.1 in two steps. First, we show that an ID-OWE attack on BasicIdent can be converted into a OWE attack on BasicPub, which shows that the private key extraction queries do not help the adversary. Then, we show that an OWE attack on BasicPub can be converted into an algorithm to solve the BDH problem.

**Lemma 3.1.2** *Let $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ be a random oracle. Let $\mathcal{A}$ be an* ID-OWE *adversary that has advantage $\varepsilon$ against* BasicIdent *which makes at most $q_E > 0$ private key extraction queries. Then there is an* OWE *adversary $\mathcal{B}$ that has advantage at least $\frac{\varepsilon}{e(1+q_E)}$ against* BasicPub. *The running time of $\mathcal{B}$ is $O(time(\mathcal{A}))$.*

**Proof** The challenger generates a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$ and a private key $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ using the algorithm Setup of BasicPub. He also picks a random plaintext $M$ and encrypts it using the algorithm Encrypt and the public key $K_{pub}$. He gives $K_{pub}$ and the resulting ciphertext $C = \langle U, V \rangle$ to $\mathcal{B}$.

$\mathcal{B}$ computes its guess for $M$ by interacting with $\mathcal{A}$ as follows

**Setup:** $B$ gives algorithm $\mathcal{A}$ the BasicIdent parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, H_1, H_2 \rangle$, where $\mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, p, P, P_{pub}, H_2$ come from $K_{pub}$ and $H_1$ is a random oracle controlled by $\mathcal{B}$.

**$H_1$-queries:** Algorithm $\mathcal{B}$ maintains a list of tuples $\langle \mathsf{ID}_j, Q_j, b_j, c_j \rangle$ containing information about the previous queries to oracle $H_1$. We call this list $H_1^{list}$. It is initially empty. $\mathcal{B}$ responds to $\mathcal{A}$'s queries to oracle $H_1$ as follows:

1. If the query $\mathsf{ID}_i$ already appears in $H_1^{list}$ in a tuple $\langle \mathsf{ID}_i, Q_i, b_i, c_i \rangle$, return $Q_i$.
2. Otherwise, $\mathcal{B}$ generates a random $coin \in \{0, 1\}$ so that $\Pr[coin = 0] = \delta$ where $\delta = 1 - \frac{1}{q_E + 1}$ (the reason for this choice will be given later).
3. $\mathcal{B}$ picks a random $b \in \mathbb{Z}_p^*$. If $coin = 0$, compute $Q_i = bP \in \mathbb{G}_1$. If $coin = 1$, compute $Q_i = bQ_{\mathsf{ID}} \in \mathbb{G}_1$.
4. $\mathcal{B}$ adds the tuple $\langle ID_i, Q_i, b, coin \rangle$ to $H_1^{list}$ and returns $Q_i$ to $\mathcal{A}$.

Note that in both cases, the distribution of $Q_i$ is uniform in $\mathbb{G}_1^*$ and independent of $\mathcal{A}$'s view.

**Private key extraction queries** Algorithm $\mathcal{B}$ responds to a private key extraction $\mathsf{ID}_i$ issued by $\mathcal{A}$ as follows:

1. If $\mathcal{A}$ had previously issued the query $\mathsf{ID}_i$ to oracle $H_1$, find the tuple $\langle ID_i, Q_i, b_i, coin_i \rangle$ in $H_1^{list}$, otherwise, create such a tuple using the procedure described above and add it to $H_1^{list}$. If $coin_i = 1$, then $\mathcal{B}$ reports failure and terminates. The attack on BasicPub has failed.
2. Otherwise, $coin_i = 0$, so $Q_i = b_i P$. Return $d_i = b_i P_{pub} \in \mathbb{G}_1^*$ to $\mathcal{A}$. Observe that $d_i$ is the private key corresponding to $\mathsf{ID}_i$ since $d_i = b_i P_{pub} = b_i s P = s Q_i$.

**Challenge** Given the public key ID on which $\mathcal{A}$ wishes to be challenged, $\mathcal{B}$ responds as follows:

1. If $\mathcal{A}$ had previously issued the query ID to oracle $H_1$, find the tuple $\langle ID, Q, b, coin \rangle$ in $H_1^{list}$, otherwise, create such a tuple using the procedure described above and add it to $H_1^{list}$. If $coin = 0$, then $\mathcal{B}$ reports failure and terminates. The attack on BasicPub has failed.
2. If $coin = 1$, then $Q = bQ_{\mathsf{ID}}$. Let $C = \langle U, V \rangle$ be the challenge ciphertext given to algorithm $\mathcal{B}$, where $U = rP \in \mathbb{G}_1$ for some $r \in \mathbb{Z}_p$. Return $C' = \langle b^{-1}U, V \rangle$ to $\mathcal{A}$, where $b^{-1}$ is the inverse of $b \bmod p$. $C'$ is a BasicIdent encryption of $M$ under the public key ID since $V$ is the exclusive-or of $M$ with the hash of $\hat{e}(Q_{\mathsf{ID}}, P_{pub})^r$ and

$$\hat{e}(b^{-1}U, d'_{\mathsf{ID}}) = \hat{e}(b^{-1}rP, sbQ_{\mathsf{ID}}) = \hat{e}(P, Q_{\mathsf{ID}})^{rsbb^{-1}}$$

$$= \hat{e}(Q_{\mathsf{ID}}, sP)^r = \hat{e}(Q_{\mathsf{ID}}, P_{pub})^r.$$

Thus, the BasicIdent decryption of $C'$ using $d'_{\mathsf{ID}}$ is the same as the BasicPub decryption of $C$ using $d_{\mathsf{ID}}$.

**Guess** Eventually, algorithm $\mathcal{A}$ outputs its guess $M'$. $\mathcal{B}$ returns $M'$ as its guess for the decryption of $C$.

**Claim:** 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. Further, if $\mathcal{B}$ does not abort, then $\Pr[M = M'] \geq \varepsilon$, where the probability is over the random bits used by $\mathcal{A}, \mathcal{B}$ and the challenger.

**Proof of Claim:** If $\mathcal{B}$ does not abort, then all responses given by the $H_1$ oracle are uniformly and independently distributed in $\mathbb{G}_1^*$, all responses to the private key extraction queries are valid and the challenge ciphertext $C'$ is the encryption of a random plaintext $M \in \mathcal{M}$. Thus, $\mathcal{A}$ view is identical to its view in the real attack. Furthermore, the challenge ciphertext $C'$ given to $\mathcal{A}$ is the BasicIdent encryption of $M$ under the public key ID chosen by $\mathcal{A}$. Hence, by definition of $\mathcal{A}$, it will make the correct guess with probability at least $\varepsilon$. ∎

It remains to calculate the probability that $\mathcal{B}$ does not abort during the simulation. If $\mathcal{A}$ makes $q_E$ private key extraction queries, then the probability that $\mathcal{B}$ does not abort treating one of those queries is $\delta^{q_E}$. The probability that $\mathcal{B}$ does not abort during the challenge step is $(1 - \delta)$. Therefore, the probability that $\mathcal{B}$ does not abort during the simulation is $\delta^{q_E}(1 - \delta)$. The value $\delta = 1 - \frac{1}{q_E+1}$ was chosen

in order to maximize this function. We can find that this probability that $\mathcal{B}$ does not abort is at least $\frac{1}{e(1+q_E)}$ [2]. ∎

The analysis used in the proof of Lemma 3.1.2 uses a similar technique to Coron's analysis of the Full Domain Hash signature scheme [14].

**Lemma 3.1.3** *Let $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ be a random oracle. Let $\mathcal{A}$ be an* OWE *adversary that has advantage $\varepsilon$ against* BasicPub. *Suppose that $\mathcal{A}$ makes at most $q_{H_2} > 0$ queries to $H_2$. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem in $\mathcal{IG}$ with advantage at least $(\varepsilon - \frac{1}{2^n})/q_{H_2}$ and running time $O(time(\mathcal{A}))$.*

**Proof** Algorithm $\mathcal{B}$ is given as input the BDH parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ produced by $\mathcal{IG}$ and a random instance $\langle P, aP, bP, cP \rangle = \langle P, P_1, P_2, P_3 \rangle$ of the BDH problem, i.e. $P$ is random in $\mathbb{G}_1^*$ and $\langle a, b, c \rangle$ are random in $\mathbb{Z}_p^*$ where $p$ is the order of $\mathbb{G}_1, \mathbb{G}_2$. Let $D = \hat{e}(P,P)^{abc}$. $\mathcal{B}$ calculates $D$ by interacting with $\mathcal{A}$ as follows.

**Setup:** $\mathcal{B}$ sets $P_{pub} = P_1$ and $Q_{\mathsf{ID}} = P_2$ and sends the BasicPub public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2 \rangle$ to algorithm $\mathcal{A}$. $H_2$ is a random oracle controlled by $\mathcal{B}$.

**Challenge:** $\mathcal{B}$ picks a random string $R \in \{0,1\}^n$, defines the challenge ciphertext to be $C = \langle P_3, R \rangle$ and sends $C$ to $\mathcal{A}$. The private key associated to $K_{pub}$ is $d_{\mathsf{ID}} = aQ_{\mathsf{ID}} = abP$. Therefore, the decryption of $C$ is $M = R \oplus H_2(\hat{e}(P_3, d_{\mathsf{ID}})) = R \oplus H_2(D)$. Note that both $d_{\mathsf{ID}}$ and $M$ are unknown to $\mathcal{B}$.

[2] To see this, note that $(1 - \frac{1}{n+1})^n \geq \frac{1}{e}$ and $\lim_{n\rightarrow\infty} (1 - \frac{1}{n+1})^n = \frac{1}{e}$.

**$H_2$-queries:** Algorithm $\mathcal{B}$ maintains a list of tuples $\langle X_j, H_j \rangle$ containing information about the previous queries to oracle $H_2$. We call this list $H_2^{list}$. It is initially empty. $\mathcal{B}$ responds to $\mathcal{A}$'s queries to oracle $H_2$ as follows:

1. If the query $X_i$ already appears in $H_2^{list}$ in a tuple $\langle X_i, H_i \rangle$, return $H_i$.
2. Otherwise, $\mathcal{B}$ picks a random string $H_i \in \{0,1\}^n$, adds the tuple $\langle X_i, H_i \rangle$ to $H_2^{list}$ and returns $H_i$.

**Guess:** Eventually, $\mathcal{A}$ outputs a guess $M'$ as the decryption of $C$. This guess is ignored and $\mathcal{B}$ picks a random tuple $\langle X_j, H_j \rangle$ from $H_2^{list}$ and outputs $X_j$ as its guess to the solution of the given instance of the BDH problem.

Again, it is easy to see that $\mathcal{A}$'s view is identical to its view in the real attack. The setup is as in the real attack since $a$ and $b$ are random in $\mathbb{Z}_p^*$, so is the challenge since $c$ is random in $\mathbb{Z}_p^*$ and the resulting encrypted message is a random plaintext since it is the exclusive-or of two random strings in $\{0,1\}^n$. Thus, $\Pr[M' = M] \geq \varepsilon$. It remains to calculate the probability that $\mathcal{B}$ outputs the correct result.

Let $\mathcal{H}$ be the event that at the end of the simulation $D$ appears in a tuple on $H_2^{list}$. Let $\delta = \Pr[\mathcal{H}]$. Note that if $D$ does not appear in $H_2^{list}$, then the decryption of $C$ is independent of $\mathcal{A}$'s view (since $H_2(D)$ is a random string in $\{0,1\}^n$ independent of $\mathcal{A}$'s view). Thus, $\Pr[M' = M \mid \neg\mathcal{H}] \leq 1/2^n$. Therefore, we have

$$\begin{aligned}
\varepsilon &\leq Pr[M' = M] = Pr[M' = M \mid \mathcal{H}] \cdot Pr[\mathcal{H}] + Pr[M' = M \mid \neg\mathcal{H}] \cdot Pr[\neg\mathcal{H}] \\
&\leq Pr[\mathcal{H}] + Pr[M' = M \mid \neg\mathcal{H}] \cdot Pr[\neg\mathcal{H}] \leq \delta + \frac{1}{2^n}(1-\delta) \\
&\Longrightarrow \delta - \frac{\delta}{2^n} \geq \varepsilon - \frac{1}{2^n}.
\end{aligned}$$

Therefore, we get $\Pr[\mathcal{H}] = \delta > \delta(1 - 1/2^n) \geq \varepsilon - 1/2^n$. Since we pick a random element from $H_2^{list}$, it follows that the probability that $\mathcal{B}$ produces the right answer is at least $\Pr[\mathcal{H}]/q_{H_2} \geq (\varepsilon - 1/2^n)/q_{H_2}$. ∎

Note that, in the proof of the preceding lemma, if $\mathcal{A}$ answers correctly, then $R \oplus M' = H_2(D)$. So $\mathcal{B}$ could scan through $H_2^{list}$, pick a random tuple $\langle X_j, H_j \rangle$ such that $H_j = H_2(D)$ and output $X_j$ instead of picking a random tuple in all of $H_2^{list}$. Suppose further that $n$ (the bitsize of the image of $H_2$) is large enough so that $2^{n/2}$ represents an infeasible amount of computation. Then, if we knew that when $\mathcal{A}$ makes less than $2^{n/2}$ calls to $H_2$, the probability that more than $k$ of these calls result in the same hash value is a negligible function $f(n)$, then we would obtain that $\mathcal{B}$ produces the right answer with probability at least $(\varepsilon - 1/2^n - f(n))/k$.

**Proof of Theorem 3.1.1.** The result follows directly from Lemmas 3.1.2 and 3.1.3. Composing the reductions we get that if there exists an ID-OWE adversary $\mathcal{A}$ that has advantage $\varepsilon$ against the scheme BasicIdent, then there is an algorithm $\mathcal{B}$ that solves the BDH problem for $\mathcal{IG}$ with advantage at least

$$\frac{\varepsilon}{e(1 + q_E) \cdot q_{H_2}} - \frac{1}{q_{H_2} \cdot 2^n},$$

as required. ∎

### 3.1.2 The Full Scheme

We use a transformation due to Fujisaki and Okamoto [21] to convert the scheme BasicIdent into an IND-ID-CCA scheme. Let $\mathcal{E}$ be a probabilistic encryption scheme

and let $\mathcal{E}_{pk}(M;r)$ denote the encryption of $M$ using the random bits $r$ under the public key $pk$. Fujisaki-Okamoto define the hybrid scheme $\mathcal{E}^{hy}$ as:

$$\mathcal{E}^{hy}_{pk}(M) = \langle \mathcal{E}_{pk}(\sigma; H(\sigma, M)), H'(\sigma) \oplus M \rangle$$

where $\sigma$ is generated at random and $H, H'$ are cryptographic hash functions. Fujisaki-Okamoto show that if $\mathcal{E}$ is a one-way encryption scheme, then $\mathcal{E}^{hy}$ is a chosen ciphertext secure system (IND-CCA) in the random oracle model (assuming $\mathcal{E}$ satisfies some natural constraints.)

We apply this transformation to BasicIdent and show that the resulting scheme, which we call FullIdent, is IND-ID-CCA. Here is the scheme FullIdent:

**Setup:** As in the BasicIdent scheme. In addition, pick cryptographic hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_p^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$. The plaintext space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$.

**Extract:** As in the BasicIdent scheme.

**Encrypt:** Given a plaintext $M \in \mathcal{M}$, a public key ID and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$,

Step 1: Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$.

Step 2: Pick a random $\sigma \in \{0,1\}^n$ and compute $r = H_3(\sigma, M)$.

Step 3: Compute $g = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2^*$.

Step 4: Compute the ciphertext $C = \langle rP, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma) \rangle$ and return $C$.

**Decrypt:** Given a ciphertext $\langle U, V, W \rangle$, a private key $d_{\mathsf{ID}} \in G_1^*$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$,

Step 1: If $U \notin \mathbb{G}_1^*$ reject the ciphertext.

Step 2: Compute $g' = \hat{e}(d_{\mathsf{ID}}, U)$.

Step 3: Compute $\sigma = V \oplus H_2(g')$.

Step 4: Compute $M = W \oplus H_4(\sigma)$.

Step 5: Compute $r = H_3(\sigma, M)$. If $U \neq rP$ reject the ciphertext.

Step 6: Return $M$.

Consistency can be shown in a similar way as with BasicIdent.

It is interesting to note that, for added security, it is possible to distribute the private key of the PKG among different sites using techniques of threshold cryptography [25]. See [5] for more details.

## Security

To prove the security of FullIdent, we need the following theorem due to Fujisaki and Okamoto (Theorem 14 in [21]). We state their theorem as it applies to the scheme BasicPub. Let BasicPub$^{hy}$ be the scheme obtained by applying the Fujisaki-Okamoto transformation to BasicPub.

**Theorem 3.1.4** *Let the hash function $H_3, H_4$ be random oracles and let $\mathcal{A}$ be a $t$-time adversary on* BasicPub$^{hy}$ *that achieves advantage $\varepsilon$. Suppose that $\mathcal{A}$ makes at most $q_D$ decryption queries and at most $Q_{H_3}, Q_{H_4}$ queries to the hash functions*

*$H_3$ and $H_4$ respectively. Then there is an adversary $\mathcal{B}$ on* BasicPub *with:*

$$time(\mathcal{B}) = FO_{time}(t, q_{H_4}, q_{H_3}) = t + O((q_{H_3} + q_{H_4}) \cdot n)$$
$$Adv(\mathcal{B}) = FO_{adv}(\varepsilon, q_{H_4}, q_{H_3}, q_D) = \frac{1}{2(q_{H_3} + q_{H_4})}[(\varepsilon + 1)(1 - 2/p)^{q_D} - 1]$$

*where $p$ is the order of the group in which* BasicPub$^{hy}$ *is implemented.*

The following theorem shows that FullIdent is IND-ID-CCA assuming that the BDH assumption holds for $\mathcal{IG}$.

**Theorem 3.1.5** *Let the hash functions $H_1, H_2, H_3, H_4$ be random oracles. Let $\mathcal{A}$ be a $t$-time* IND-ID-CCA *adversary on* FullIdent *that achieves advantage $\varepsilon$ and which makes at most $q_E > 0$ private key extraction queries, at most $q_D > 0$ 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 BDH algorithm $\mathcal{B}$ in $\mathcal{IG}$ with:*

$$time_{\mathcal{B}}(\lambda) \leq O(FO_{time}(t, q_{H_4}, q_{H_3}))$$
$$Adv BDH_{\mathcal{IG},\mathcal{B}}(\lambda) \geq \left( FO_{adv}\left( \frac{\varepsilon}{e(1 + q_E + q_D)}, q_{H_4}, q_{H_3}, q_D \right) - \frac{1}{2^n} \right) \Big/ q_{H_2}.$$

We first need the following lemma, which shows that an IND-ID-CCA attacker on FullIdent can be transformed into an IND-CCA attacker on BasicPub$^{hy}$ (so again, this shows that the private key extraction queries do not help the adversary).

**Lemma 3.1.6** *Let $\mathcal{A}$ be an* IND-ID-CCA *adversary that has advantage $\varepsilon$ against the scheme* FullIdent *which makes at most $q_E > 0$ private key extraction queries and at*

*most $q_D > 0$ decryption queries. Then there is an* IND-CCA *adversary $\mathcal{B}$ that has advantage at least $\frac{\varepsilon}{e(1+q_E+q_D)}$ against* $\mathsf{BasicPub}^{hy}$. *Its running time is $O(time(\mathcal{A}))$.*

**Proof** The challenger obtains a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, Q_{\mathsf{ID}}, H_2, H_3, H_4 \rangle$ and a private key $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ by running the Setup algorithm of $\mathsf{BasicPub}^{hy}$ and gives $K_{pub}$ to $\mathcal{B}$.

$\mathcal{B}$ mounts an attack against $\mathsf{BasicPub}^{hy}$ on the key $K_{pub}$ by interacting with $\mathcal{A}$ as follows:

**Setup:** $\mathcal{B}$ gives to $\mathcal{A}$ the FullIdent parameters $\langle \mathbb{G}_1, \mathbb{G}_2, p, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$ where $\mathbb{G}_1, \mathbb{G}_2, p, n, P, P_{pub}, H_2, H_3, H_4$ come from $K_{pub}$ and $H_1$ is a random oracle controlled by $\mathcal{B}$

**$H_1$-queries** Algorithm $\mathcal{B}$ maintains a list of tuples $\langle \mathsf{ID}_j, Q_j, b_j, c_j \rangle$ containing information about the previous queries to oracle $H_1$. We call this list $H_1^{list}$. It is initially empty. $\mathcal{B}$ responds to $\mathcal{A}$'s queries to oracle $H_1$ as follows:

1. If the query $\mathsf{ID}_i$ already appears in $H_1^{list}$ in a tuple $\langle \mathsf{ID}_i, Q_i, b_i, c_i \rangle$, return $Q_i$.
2. Otherwise, $\mathcal{B}$ generates a random $coin \in \{0,1\}$ so that $\Pr[coin = 0] = \delta$ where $\delta = 1 - \frac{1}{1+q_E+q_D}$ (the reason for this choice will be given later).
3. $\mathcal{B}$ picks a random $b \in \mathbb{Z}_p^*$. If $coin = 0$, compute $Q_i = bP \in \mathbb{G}_1$. If $coin = 1$, compute $Q_i = bQ_{\mathsf{ID}} \in \mathbb{G}_1$.
4. $\mathcal{B}$ adds the tuple $\langle ID_i, Q_i, b, coin \rangle$ to $H_1^{list}$ and returns $Q_i$.

Note that in both cases, the distribution of $Q_i$ is uniform in $\mathbb{G}_1^*$ and independent of $\mathcal{A}$'s view.

**Phase 1: Private key queries.** Algorithm $\mathcal{B}$ responds to a private key extraction $\mathsf{ID}_i$ issued by $\mathcal{A}$ as follows:

1. If $\mathcal{A}$ had previously issued the query $\mathsf{ID}_i$ to oracle $H_1$, find the tuple $\langle ID_i, Q_i, b_i, coin_i \rangle$ in $H_1^{list}$, otherwise, create such a tuple using the procedure described above and add it to $H_1^{list}$. If $coin_i = 1$, then $\mathcal{B}$ reports failure and terminates. The attack on $\mathsf{BasicPub}^{hy}$ has failed.
2. Otherwise, $coin_i = 0$, so $Q_i = b_i P$. Return $d_i = b_i P_{pub} \in \mathbb{G}_1^*$ to $\mathcal{A}$. Observe that $d_i$ is the private key corresponding to $\mathsf{ID}_i$ since $d_i = b_i P_{pub} = b_i s P = s Q_i$.

**Phase 1: Decryption queries.** Let $\langle \mathsf{ID}_i, C_i \rangle$ be a decryption query issued by $\mathcal{A}$. Say $C_i = \langle U_i, V_i, W_i \rangle$. Algorithm $\mathcal{B}$ responds to this query as follows:

1. If $\mathcal{A}$ had previously issued the query $\mathsf{ID}_i$ to oracle $H_1$, find the tuple $\langle ID_i, Q_i, b_i, coin_i \rangle$ in $H_1^{list}$, otherwise, create such a tuple using the procedure described above and add it to $H_1^{list}$.
2. If $coin_i = 0$, then $\mathcal{B}$ runs the algorithm for private key extraction to obtain the private key corresponding to $\mathsf{ID}_i$. Then, it uses the private key to decrypt the ciphertext $C_i$ and returns the corresponding plaintext to $\mathcal{A}$.
3. If $coin_i = 1$, then $Q_i = b_i Q_{\mathsf{ID}}$. Recall that $U_i \in \mathbb{G}_1$. Set $C_i' = \langle b_i U_i, V_i, W_i \rangle$. Then the $\mathsf{FullIdent}$ decryption of $C_i$ using $d_i = s Q_i$ (the unknown $\mathsf{FullIdent}$ private decryption key corresponding to $\mathsf{ID}_i$) is the

same as the $\mathsf{BasicPub}^{hy}$ decryption of $C_i'$ using $d_{\mathsf{ID}}$ since

$$\hat{e}(b_iU_i, d_{\mathsf{ID}}) = \hat{e}(U_i, sQ_{\mathsf{ID}})^{b_i} = \hat{e}(U_i, sb_iQ_{\mathsf{ID}}) = \hat{e}(U_i, sQ_i) = \hat{e}(U_i, d_i).$$

So $\mathcal{B}$ can issue the decryption query $\langle C_i' \rangle$ to the challenger and give the challenger's response to $\mathcal{A}$.

**Challenge** Given the public key $\mathsf{ID}$ and the two messages $M_0, M_1$ on which the challenger wishes to be challenged,

1. $\mathcal{B}$ gives $M_0, M_1$ to the challenger as the messages on which it wishes to be tested. The challenger responds with a ciphertext $C = \langle U, V, W \rangle$ such that $C$ is the $\mathsf{BasicPub}^{hy}$ encryption of $M_c$ for a random $c \in \{0, 1\}$.

2. If $\mathcal{A}$ had previously issued the query $\mathsf{ID}$ to oracle $H_1$, find the tuple $\langle ID, Q, b, coin \rangle$ in $H_1^{list}$, otherwise, create such a tuple using the procedure described above and add it to $H_1^{list}$. If $coin = 0$ then $\mathcal{B}$ reports failure and terminates. The attack on $\mathsf{BasicPub}^{hy}$ has failed.

3. If $coin = 1$, then $Q = bQ_{\mathsf{ID}}$. Return $C' = \langle b^{-1}U, V, W \rangle$ to $\mathcal{A}$, where $b^{-1}$ is the inverse of $b \bmod p$. As in the proof of lemma 3.1.2, we can find that $C'$ is the $\mathsf{FullIdent}$ encryption of $M_c$ under the public key $\mathsf{ID}$, as required.

**Phase 2: Private key queries.** As in Phase 1.

**Phase 2: Decryption queries.** As in Phase 1. Note that $\mathcal{B}$ is not allowed to issue to the challenger the decryption query $\langle U, V, W \rangle$, but this is not a problem

since this happens if and only if $\mathcal{A}$ issues the decryption query $\langle b^{-1}U, V, W\rangle = C'$, which, by definition, it is not allowed to do.

**Guess:** Eventually, algorithm $\mathcal{A}$ produces a guess $c'$ for $c$. $\mathcal{B}$ outputs $c'$ as its guess for $c$.

**Claim:** 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. Further, if $\mathcal{B}$ does not abort, then $|\Pr[c = c'] - \frac{1}{2}| \geq \varepsilon$, where the probability is over the random bits used by $\mathcal{A}, \mathcal{B}$ and the challenger.

**Proof of Claim:** If $\mathcal{B}$ does not abort, the responses to the $H_1$-queries are uniformly distributed in $\mathbb{G}_1^*$ as in the real attack, the responses to the decryption and private key extraction queries are valid and the challenge ciphertext $C'$ given to $\mathcal{A}$ is the FullIdent encryption of $M_c$ for a random $c \in \{0, 1\}$. Thus, by definition of algorithm $\mathcal{A}$, we have $|\Pr[c = c'] - \frac{1}{2}| \geq \varepsilon$. The result then follows from the fact that $\mathcal{B}$ outputs the correct answer if and only if $\mathcal{A}$'s answer is correct. ∎

It remains to bound the probability that $\mathcal{B}$ aborts during the simulation. If $\mathcal{B}$ aborts, it can be only for one of the following (these are the only events that can cause $\mathcal{B}$ to abort)

1. $\mathcal{A}$ issues a bad private key extraction query in Phase 1 or 2.
2. $\mathcal{A}$ chooses a bad ID to be challenged on.

We define two corresponding events:

$\mathcal{E}_1$ is the event that $\mathcal{A}$ issues a private key extraction query that causes $\mathcal{B}$ to abort.

$\mathcal{E}_2$ is the event that $\mathcal{A}$ asks to be challenged on an $\mathsf{ID}$ that causes $\mathcal{B}$ to abort.

**Claim:** $\Pr[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2] \geq \delta^{q_E+q_D}(1-\delta)$.

**Proof of Claim:** We prove the claim by induction on the number of private key extraction and decryption queries issued by the adversary. let $\mathcal{E}^{0\ldots i}$ be the event that $\mathcal{E}_1 \vee \mathcal{E}_2$ happens after $\mathcal{A}$ issues at most $i$ queries and let $\mathcal{E}^i$ be the event that $\mathcal{E}_1 \vee \mathcal{E}_2$ happens for the first time when $\mathcal{A}$ issues the $i^{th}$ query. We prove by induction that $\Pr[\neg\mathcal{E}^{0\ldots i}] \geq \delta^i(1-\delta)$.

If $i = 0$, then the only reason for which $\mathcal{B}$ could abort is if $\mathcal{E}_2$ occurs. Thus $\Pr[\neg\mathcal{E}^{0\ldots 0}] \geq 1-\delta$. Now, suppose that the claim holds for $i-1$. Then

$$\begin{aligned} Pr[\neg\mathcal{E}^{0\ldots i}] &= Pr[\neg\mathcal{E}^{0\ldots i} \mid \neg\mathcal{E}^{0\ldots i-1}]Pr[\neg\mathcal{E}^{0\ldots i-1}] \\ &= Pr[\neg\mathcal{E}^{i} \mid \neg\mathcal{E}^{0\ldots i-1}]Pr[\neg\mathcal{E}^{0\ldots i-1}] \\ &\geq Pr[\neg\mathcal{E}^{i} \mid \neg\mathcal{E}^{0\ldots i-1}]\delta^{i-1}(1-\delta). \end{aligned}$$

Hence, it suffices to show that $\Pr[\neg\mathcal{E}^i \mid \neg\mathcal{E}^{0\ldots i-1}] = q_i \geq \delta$, i.e. we bound the probability that the $i^{th}$ query does not cause $\mathcal{B}$ to abort knowing that the first $i-1$ did not. The $i^{th}$ query is either a private key extraction for $\langle \mathsf{ID}_i \rangle$ or a decryption query for $\langle \mathsf{ID}_i, C_i \rangle$ where $C_i = \langle U_i, V_i, W_i \rangle$.

Let $H_1(\mathsf{ID}_i) = Q_i$ and let $\langle \mathsf{ID}_i, Q_i, b_i, coin_i \rangle$ be the corresponding tuple in $H_1^{list}$. Note that if $coin_i = 0$, $\mathcal{E}_1$ cannot happen since $\mathcal{B}$ is able to calculate the private key corresponding to $\mathsf{ID}_i$. There are two cases to consider.

Case 1: $\mathsf{ID}_i$ is not equal to the public key $\mathsf{ID}$ on which $\mathcal{A}$ is being challenged. Then

$q_i \geq \Pr[coin_i = 0] = \delta$.

Case 2: $\mathsf{ID}_i = \mathsf{ID}$. Then the $i^{th}$ query is not a private key extraction query since such an extraction query is not allowed, by definition. So it must be a decryption query $\langle \mathsf{ID}, C_i \rangle$. Therefore, $q_i = 1$ since $\mathcal{B}$ never aborts in a decryption query.

Hence, in both cases, $q_i \geq \delta$, so $\Pr[\neg \mathcal{E}^{0 \ldots i}] \geq \delta^i(1-\delta)$. The claim follows from the fact that

$$\begin{aligned} Pr[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2] &= Pr[\neg\mathcal{E}^{0\ldots i}] \text{ for some } i,\ 0 \leq i \leq (q_D + q_E) \\ &\geq \delta^i(1-\delta) \text{ for some } i,\ 0 \leq i \leq (q_D + q_E) \\ &\geq \delta^{q_E+q_D}(1-\delta) \quad \blacksquare \end{aligned}$$

As in the proof of Lemma 3.1.2, $\delta$ was chosen to maximize $\delta^{q_E+q_D}(1-\delta)$ and the probability that $\mathcal{B}$ does not abort is at least $\frac{1}{e(1+q_E+q_D)}$. Hence, $\mathcal{B}$ advantage is at least $\frac{\varepsilon}{e(1+q_E+q_D)}$. $\blacksquare$

**Proof of Theorem 3.1.5** The result follows directly from Lemma 3.1.6, Theorem 3.1.4 and Lemma 3.1.3. Composing the reductions we get that if there exists an ID-OWE adversary $\mathcal{A}$ that has advantage $\varepsilon$ against the scheme BasicIdent, then there is an algorithm $\mathcal{B}$ that solves the BDH problem for $\mathcal{IG}$ with advantage at least

$$\left( FO_{adv}\left( \frac{\varepsilon}{e(1+q_E+q_D)}, q_{H_4}, q_{H_3}, q_D \right) - \frac{1}{2^n} \right) \Big/ q_{H_2}.$$

and which runs in time $O(FO_{time}(t, q_{H_4}, q_{H_3}))$, as required. $\blacksquare$

## 3.2 Authenticated Identity-Based Encryption

The scheme by Boneh and Franklin can be modified to include message authentication without any expansion in the length of the ciphertext. In this section, we present the scheme by Lynn [34] for authenticated identity-based encryption. It is based on an idea presented by Sakai, Ohgishi and Kasahara in [47].

The level of security achieved is the same as that in a private conversation, i.e. secure authenticated communication without the ability to prove to a third party that any information was ever exchanged. Such a scheme would be ideal for applications such as email exchange.

It is interesting to note that authenticated encryption is obtained from the Boneh-Franklin scheme with no additional computational cost. Also, since the authentication code is the ciphertext itself, proving integrity is equivalent to proving ciphertext unforgeability.

We mention that in [36], Malone-Lee claims to give an identity-based signcryption scheme, i.e. a scheme in which the ciphertexts are authenticated and non-repudiable. However, he does not give any formal proof of security. That scheme is not presented here.

### 3.2.1 Definitions

#### Authenticated Identity-Based Scheme

An *authenticated identity-based encryption scheme* consists of four randomized algorithms: Setup, Extract, Aut-Encrypt and Aut-Decrypt.

Setup: takes as input a security parameter and outputs params and master-key. The system parameters params must include a description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$. The system parameters will be publicly known, while the master-key is known only to the PKG.

Extract: takes as input the system parameters params, the master-key and an arbitrary string $\mathsf{ID} \in \{0,1\}^*$ and returns the private decryption key $d$ corresponding to the public key ID.

Aut-Encrypt: takes as input the system parameters params, a private key $d$ (the sender's), a public key ID (the receiver's) and a message $M$ and outputs a ciphertext.

Aut-Decrypt: takes as input the system parameters, params, a private key $d$ (the receiver's), a public key ID (the sender's) and a ciphertext $C$ and outputs the corresponding plaintext.

These algorithms are required to satisfy the standard consistency constraints, namely if params is produced by the Setup algorithm, $\mathsf{ID}_A, \mathsf{ID}_B$ are two public keys and $d_A, d_B$ are the corresponding private keys generated by the algorithm Extract, then

$$\forall M \in \mathcal{M},\ \textsf{Aut-Decrypt}(\textsf{params}, d_B, \mathsf{ID}_A, \textsf{Aut-Encrypt}(\textsf{params}, d_A, \mathsf{ID}_B, M)) = M.$$

## Security

We must again modify slightly the definition of semantic security against chosen ciphertext attack to allow the adversary to make encryption queries since he can no longer encrypt a plaintext by himself unless he is given access to a private key.

We define the notion of semantic security against adaptive chosen ciphertext attack for an authenticated identity-based scheme (IND-AID-CCA) through the following game:

**Setup:** The challenger takes a security parameter $\lambda$ and runs the Setup algorithm. He returns to the adversary the public system parameters params and keeps the master-key to himself.

**Phase 1:** The adversary issues queries $q_1, \ldots, q_m$ where each query is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $d_i$ corresponding to the public key $\mathsf{ID}_i$ and sends it to the adversary.
- Encryption query $\langle \mathsf{ID}_{A_i}, \mathsf{ID}_{B_i}, M_i \rangle$. The challenger responds by running algorithm Extract to generate the private keys $d_{A_i}$ corresponding to the public keys $\mathsf{ID}_{A_i}$, uses this private key to encrypt the plaintext $M_i$ and returns the resulting ciphertext to the adversary.
- Decryption query $\langle \mathsf{ID}_{A_i}, \mathsf{ID}_{B_i}, C_i \rangle$. The challenger responds by running algorithm Extract to generate the private keys $d_{B_i}$ corresponding to the public keys $\mathsf{ID}_{B_i}$, uses this private key to decrypt the ciphertext $C_i$ and returns the resulting plaintext to the adversary.

**Challenge:** The adversary outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and two public keys $\mathsf{ID}_A, \mathsf{ID}_B$ on which he wishes to be tested. The only constraint is $\mathsf{ID}_A, \mathsf{ID}_B$ must not have appeared in any extraction query in Phase 1. The challenger picks a random bit $c \in \{0,1\}$ and sends $C = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}_A, \mathsf{ID}_B, M_c)$ as the challenge to the adversary.

**Phase 2:** The adversary issues queries $q_{m+1}, \ldots, q_n$ where each query is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$ where $\mathsf{ID}_i \notin \{\mathsf{ID}_A, \mathsf{ID}_B\}$. The challenger responds as in Phase 1.
- Encryption query $\langle \mathsf{ID}_{A_i}, \mathsf{ID}_{B_i}, M_i \rangle$. The challenger responds as in Phase 1.
- Decryption query $\langle \mathsf{ID}_{A_i}, \mathsf{ID}_{B_i}, C_i \rangle \neq \langle \mathsf{ID}_A, \mathsf{ID}_B, C \rangle$. The challenger responds as in Phase 1.

**Guess:** The adversary outputs a guess $c' \in \{0,1\}$. The adversary wins the game if $c' = c$.

Such an adversary is called an IND-AID-CCA attacker. The advantage of an IND-AID-CCA attacker against the scheme is defined to be:

$$Adv_{\mathcal{A}}(\lambda) = \left| \Pr[c = c'] - \frac{1}{2} \right|$$

where the probability is over the random choices made by the challenger and the adversary. We say that an identity-based encryption scheme is semantically secure against adaptive chosen ciphertext attack (IND-AID-CCA) if no polynomially

bounded adversary (in $\lambda$) has non-negligible advantage (in $\lambda$) in the game described above.

## Integrity

As mentioned before, in this scheme, the ciphertext is the authentication code of the message. Therefore, to ensure integrity, it must be infeasible to forge ciphertexts.

We define the notion of security against ciphertext forgery (AID-CUF) through the following game:

**Setup:** The challenger takes a security parameter $\lambda$ and runs the Setup algorithm. He returns the adversary the public system parameters params and keeps the master-key to itself.

**Query Phase:** The adversary issues queries $q_1, \ldots, q_m$ where each query is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $d_i$ corresponding to the public key $\mathsf{ID}_i$ and sends it to the adversary.
- Encryption query $\langle \mathsf{ID}_{A_i}, \mathsf{ID}_{B_i}, M_i \rangle$. The challenger responds by running algorithm Extract to generate the private keys $d_{A_i}$ corresponding to the public keys $\mathsf{ID}_{A_i}$, uses this private key to encrypt the plaintext $M_i$ and returns the resulting ciphertext to the adversary.
- Decryption query $\langle \mathsf{ID}_{A_i}, \mathsf{ID}_{B_i}, C_i \rangle$. The challenger responds by running algorithm Extract to generate the private keys $d_{B_i}$ corresponding to the

public keys $\mathsf{ID}_{B_i}$, uses this private key to decrypt the ciphertext $C_i$ and returns the resulting plaintext to the adversary.

**Guess:** The adversary outputs two public keys, $\mathsf{ID}_A, \mathsf{ID}_B$ and a string of bits $C'$, his attempt to forge a valid ciphertext from sender $\mathsf{ID}_A$ to receiver $\mathsf{ID}_B$ such that $\mathsf{ID}_A$ and $\mathsf{ID}_B$ did not appear in any extraction query and $C'$ is different from all the output of the encryption queries issued with $\mathsf{ID}_A$ and $\mathsf{ID}_B$. The adversary wins if $C'$ is a valid ciphertext.

We call such an adversary an AID-CUF attacker. We define the advantage of an AID-CUF attacker to be $\Pr[C' \text{ is a valid ciphertext}]$. We say that an authenticated identity-based scheme is secure against ciphertext forgery (AID-CUF) if no polynomially bounded adversary (in $\lambda$) has non-negligible advantage (in $\lambda$) in the game above.

### 3.2.2 The Scheme

In this section, we give the description of an authenticated identity-based encryption scheme and state the theorems about its security. We direct the reader to the paper [34] for complete proofs of security.

Here is the description of the scheme, which we call AutIdent.

Setup: Given a security parameter $\lambda$

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in G_1^*$.

Step 2: Pick a random $s \in \mathbb{Z}_p^*$.

Step 3: Pick cryptographic hash functions $H_1 : \mathbb{Z}_p \times \mathbb{G}_2 \rightarrow \{0,1\}^n$, $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_3 : \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$ for some positive integer $n$.

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $C = \mathbb{Z}_p \times \{0,1\}^n \times \{0,1\}^n$. The public system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2, H_3, H_4 \rangle$. The master-key is $s \in \mathbb{Z}_p^*$.

Extract: Given a string $\mathsf{ID} \in \{0,1\}^*$, the master-key $s$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2, H_3, H_4 \rangle$,

Step 1: Compute $Q_{\mathsf{ID}} = H_2(\mathsf{ID}) \in \mathbb{G}_1^*$.

Step 2: Compute $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. Return $d_{\mathsf{ID}}$.

Encrypt: Given a plaintext $M \in \mathcal{M}$, a private key $d_{\mathsf{ID}_A}$, a public key $\mathsf{ID}_B$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2, H_3, H_4 \rangle$,

Step 1: Pick a random $\sigma \in \{0,1\}^n$.

Step 2: Compute $r = H_3(\sigma, M)$.

Step 4: Compute $g = \hat{e}(d_{\mathsf{ID}_A}, H_2(\mathsf{ID}_B)) \in \mathbb{G}_2^*$.

Step 5: Compute the ciphertext $C = \langle r, \sigma \oplus H_1(r, g), M \oplus H_4(\sigma) \rangle$ and return $C$.

Decrypt: Given a ciphertext $\langle U, V, W \rangle$, a public key $\mathsf{ID}_A$, a private key $d_{\mathsf{ID}_B}$ system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2, H_3, H_4 \rangle$,

Step 2: Compute $g = \hat{e}(H_2(\mathsf{ID}_A), d_{\mathsf{ID}_B})$.

Step 3: Compute $\sigma = V \oplus H_1(U, g)$.

Step 4: Compute $M = W \oplus H_4(\sigma)$.

Step 5: Compute $r = H_3(\sigma, M)$. If $U \neq r$ reject the ciphertext.

Step 6: Return $M$.

Consistency follows from

$$\begin{aligned}\hat{e}(H_2(\mathsf{ID}_A), d_{\mathsf{ID}_B}) &= \hat{e}(H_2(\mathsf{ID}_A), sH_2(\mathsf{ID}_B)) \\ &= \hat{e}(sH_2(\mathsf{ID}_A), H_2(\mathsf{ID}_B)) = \hat{e}(d_{\mathsf{ID}_A}, H_2(\mathsf{ID}_B)).\end{aligned}$$

In step 5 of the encryption algorithm, $M \oplus H_4(\sigma)$ can be replaced by the encryption of $M$ using any semantically secure symmetric encryption scheme with key $H_4(\sigma)$. Step 4 of the decryption algorithm would then have to be modified accordingly.

**Theorem 3.2.1** *Let the hash functions $H_1, H_2, H_3, H_4$ be random oracles. Suppose $\mathcal{A}$ is an AID-CUF attacker that can forge an AutIdent ciphertext with advantage $\varepsilon$ and makes at most $q$ $H_2$-queries and at most $q_D$ decryption queries. Then there exists an algorithm $\mathcal{B}$ that solves the BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ which runs in time $O(time(\mathcal{A}))$ and has advantage at least $\varepsilon / q_D \binom{q}{2}^2$.*

**Theorem 3.2.2** *Let the hash function $H_1, H_2, H_3, H_4$ be random oracles. Suppose $\mathcal{A}$ is an IND-AID-CCA attacker with advantage $\varepsilon$ against the scheme AutIdent and whose number of $H_1$-queries is bounded by $q_1$ and whose number of $H_2$-queries*

*is bounded by $q_2$. Furthermore, suppose that the scheme is AID-CUF. Then there exists an algorithm $\mathcal{B}$ that solves the BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ which runs in time $O(time(\mathcal{A}))$ and has advantage at least $\varepsilon / q_1 \binom{q_2}{2}^2$.*

Note that the scheme has the property that, for any 2 public keys $\mathsf{ID}_A, \mathsf{ID}_B$ and any plaintext $M \in \mathcal{M}$,

$$\mathsf{Encrypt}(\mathsf{params}, d_A, \mathsf{ID}_B, M) = \mathsf{Encrypt}(\mathsf{params}, d_B, \mathsf{ID}_A, M)$$

where $d_A, d_B$ are the private key corresponding to $\mathsf{ID}_A$ and $\mathsf{ID}_B$ respectively. This is why it is impossible for the receiver to prove to a third party that the exchange of information ever took place.

## 3.3 Hierarchical Identity-Based Encryption

A disadvantage of the Boneh-Franklin identity-based encryption scheme is that, in a large network, the private key generator would have a quite burdensome job. One solution to this problem is to allow a hierarchy of PKGs in which PKGs have to compute private keys only to the entities immediately below them in the hierarchy.

In this section, we describe two hierarchical identity-based encryption schemes. The first one, by Horwitz and Lynn [29], is 2-level hierarchical identity-based encryption scheme that in addition allows key escrow at each level (the PKG can decrypt messages produced by all the users below them in the hierarchy). The second scheme, by Gentry and Silverberg [26], allows for an arbitrary number of levels (however, the size of the ciphertext grows with the number of levels) and

encrypted messages can be decrypted only by the PKG immediately above the user who produced it.

### 3.3.1 Definitions

## Hierarchical Identity-Based Encryption Scheme

A *primitive ID* (PID) is an arbitrary string, i.e. an element of $\{0,1\}^*$.

An *address* is an $l$-tuple of PID's. An address $\langle \mathsf{ID}_1, \ldots, \mathsf{ID}_i \rangle$ is said to be a *prefix* of address $\langle \mathsf{ID'}_1, \ldots, \mathsf{ID'}_j \rangle$ if $i \leq j$ and $\mathsf{ID'}_a = \mathsf{ID}_a$ for all $1 \leq a \leq i$.

An *$l$-level hierarchical identity-based scheme* ($l$-HIDE) consists of $l+3$ randomized algorithms: Root Setup, Lower-level Setup, Extract$_i$ for $1 \leq i \leq l$, Encrypt and Decrypt.

Root Setup: takes as input a security parameter and outputs params and root secret. The system parameters params must include a description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$. The system parameters will be publicly known, while the root secret is known only to the root PKG.

Lower-level Setup: takes as input the system parameters params and outputs a lower-level secret. This algorithm is necessary only if the scheme requires the lower-level entities to have such a lower-level secret.

Extract$_i$ (for $1 \leq i \leq l$) : takes as input the system parameters params, an $i$-tuple of PID's $(\mathsf{ID}_1, \ldots, \mathsf{ID}_i)$ and a level-$(i-1)$ private key $\mathsf{mk}_{\langle \mathsf{ID}_1, \ldots, \mathsf{ID}_{i-1} \rangle}$ and outputs a level-$i$ private key $\mathsf{mk}_{\langle \mathsf{ID}_1, \ldots, \mathsf{ID}_i \rangle}$.

Encrypt: takes as input the system parameters params, an address and a plaintext $M \in \mathcal{M}$ and outputs the corresponding ciphertext.

Decrypt: takes as input the system parameters params, a secret key $\mathsf{mk}_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_l\rangle}$ and a ciphertext $C \in \mathcal{C}$ and outputs the corresponding plaintext.

## Security

The notion of semantic security against adaptive chosen ciphertext attack for a hierarchical identity-based encryption scheme (IND-HID-CCA) is defined through the following game:

**Setup:** The challenger takes a security parameter $\lambda$ and runs the Setup algorithm. He returns to the adversary the public system parameters params and keeps the root secret to itself.

**Phase 1:** The adversary issues queries $q_1, \ldots, q_m$ where each query is one of:

- Extraction query $\langle \mathsf{PID\text{-}tuple}_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $\mathsf{mk}_{\mathsf{PID\text{-}tuple}}$ corresponding to $\mathsf{PID\text{-}tuple}_i$ and returns it to the adversary.
- Decryption query $\langle \mathsf{PID\text{-}tuple}_i, C_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $\mathsf{mk}_{\mathsf{PID\text{-}tuple}}$ corresponding to $\mathsf{PID\text{-}tuple}_i$, uses this private key to decrypt the ciphertext $C_i$ and returns the resulting plaintext to the adversary.

**Challenge:** The adversary outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and a PID-tuple on which he wishes to be tested. The only constraints are that

neither this PID-tuple nor any of its ancestors appeared in any extraction query in Phase 1. The challenger picks a random bit $c \in \{0,1\}$ and sends $C = \mathsf{Encrypt}(\mathsf{params}, \mathsf{PID\text{-}tuple}, M_c)$ as the challenge to the adversary.

**Phase 2:** The adversary issues queries $q_{m+1}, \ldots, q_n$ where each query is one of:

- Extraction query $\langle \mathsf{PID\text{-}tuple}_i \rangle$ where $\mathsf{PID\text{-}tuple}_i \neq \mathsf{PID\text{-}tuple}$ or any of its ancestors. The challenger responds as in Phase 1.
- Decryption query $\langle \mathsf{PID\text{-}tuple}_i, C_i \rangle \neq \langle \mathsf{PID\text{-}tuple}, C \rangle$. The challenger responds as in Phase 1.

**Guess:** The adversary outputs a guess $c' \in \{0,1\}$. The adversary wins the game if $c' = c$.

Such an adversary is called an IND-HID-CCA attacker. The advantage of an IND-HID-CCA attacker against the scheme is defined to be:

$$Adv_{\mathcal{A}}(\lambda) = \left| \Pr[c = c'] - \frac{1}{2} \right|$$

where the probability is over the random choices made by the challenger and the adversary. We say that an identity-based encryption scheme is semantically secure against adaptive chosen ciphertext attack (IND-HID-CCA) if no polynomially bounded adversary (in $\lambda$) has non-negligible advantage (in $\lambda$) in the game described above.

The notion of one-way hierarchical identity-based encryption is defined through the following game:

**Setup:** The challenger takes a security parameter $\lambda$ and runs the Setup algorithm. He returns to the adversary the public system parameters params and keeps the root secret to itself.

**Phase 1:** The adversary issues private key extraction queries $\mathsf{PID\text{-}tuple}_1, \ldots, \mathsf{PID\text{-}tuple}_m$. The challenger responds by running algorithm Extract to generate the private key $\mathsf{mk}_{\mathsf{PID\text{-}tuple}}$ corresponding to the public key $\mathsf{PID\text{-}tuple}_i$ and sends it to the adversary.

**Challenge:** The adversary outputs a PID-tuple on which he wishes to be challenged. The only constraints are that neither this PID-tuple nor any of its ancestors appeared in any extraction query in Phase 1. The challenger picks a random plaintext $M \in \mathcal{M}$, encrypts it using PID-tuple as a public key and sends the resulting ciphertext to the adversary.

**Phase 2:** The adversary issues more private key extraction queries $\mathsf{PID\text{-}tuple}_{m+1}, \ldots, \mathsf{PID\text{-}tuple}_n$ different from the challenge PID-tuple. The challenger responds as in Phase 1.

**Guess:** The adversary outputs a guess $M' \in \mathcal{M}$. The adversary wins if $M' = M$.

Such an adversary is called an HID-OWE attacker. The advantage of an HID-OWE attacker against the scheme is defined to be $\Pr[M' = M]$, where the probability is over the random choices made by the challenger and the adversary. We say that a hierarchical identity-based encryption scheme is a one-way hierarchical identity-based encryption scheme (HID-OWE) if no polynomially bounded adversary (in $\lambda$) has non-negligible advantage (in $\lambda$) in the game described above.

### 3.3.2 A 2-HIDE Domain-Collusion Resistant Scheme with Escrow at Each Level

The following scheme, given by Horwitz and Lynn [29], is a 2-HIDE scheme such that the root secret can be used as an escrow key to decrypt messages produced by any user. The PKGs on the intermediate level can also decrypt the messages produced by the users below them (we call these intermediate PKGs, together with all the users below them *domains*). We also note that only the addresses at the lowest level can be used as public keys for encryption. The scheme is secure only as long as there is limited collusion between the users under a given intermediate PGK. We call the scheme Esc-2-HIDE

Let $m$ be the maximal amount of collusion we are willing to tolerate among users of the same domain.

Root Setup: Given a BDH parameter generator $\mathbb{G}$ and a security parameter $\lambda$,

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in G_1^*$.

Step 2: Pick a random $s \in \mathbb{Z}_p^*$ and set $P_{pub} = sP$.

Step 3: Pick cryptographic hash functions $H_1 : \{0,1\}^* \to {\mathbb{G}_1^*}^{m+1}$, $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ and $H_3 : \mathbb{G}_2 \to \{0,1\}^n$ for some positive integer $n$.

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$. The public system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2, H_3 \rangle$. The root secret is $\mathsf{mk}_\epsilon = s$.

$\mathsf{Extract}_1$: Given a prefix address $\langle \mathsf{ID}_1 \rangle$, the root secret $s$ and the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2, H_3 \rangle$,

Step 1: Compute $\langle P_0, \ldots, P_m \rangle = H_1(\mathsf{ID}_1) \in {\mathbb{G}_1^*}^{m+1}$

Step 2: For $0 \leq i \leq m$, compute $sP_i$ and return $\mathsf{mk}_{\langle \mathsf{ID}_1 \rangle} = \langle sP_0, \ldots, sP_m \rangle$ (the level-1 private key).

$\mathsf{Extract}_2$: Given an address $\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle$, the level-1 address $\mathsf{mk}_{\langle \mathsf{ID}_1 \rangle} = \langle P'_0, \ldots, P'_m \rangle$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2, H_3 \rangle$,

Step 1: Compute $d = H_2(\mathsf{ID}_1 || \mathsf{ID}_2)$.

Step 2: Compute $d_{\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle} = \sum_{i=0}^{m} d^i P'_i \in \mathbb{G}_1$ and return $\mathsf{mk}_{\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle} = d_{\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle}$ (the level-2 private key).

$\mathsf{Encrypt}$: Given a plaintext $M \in \mathcal{M}$, an address $\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2, H_3 \rangle$,

Step 1: Compute $\langle P_0, \ldots, P_m \rangle = H_1(\mathsf{ID}_1)$ and $d = H_2(\mathsf{ID}_1 || \mathsf{ID}_2)$.

Step 2: Compute $P' = \sum_{i=0}^{m} d^i P_i$.

Step 3: Compute $g = \hat{e}(P', P_{pub})$.

Step 4: Pick a random $r \in \mathbb{Z}_p^*$.

Step 5: Compute the ciphertext $C = \langle rP, M \oplus H_3(g^r) \rangle$ and return $C$.

$\mathsf{Decrypt}$: Given a ciphertext $C = \langle U, V \rangle$, a level-2 private key $d_{\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2, H_3 \rangle$,

Step 1: Compute $g' = \hat{e}(d_{\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle}, U)$.

Step 2: Compute $M = V \oplus H_3(g')$ and return $M$.

Consistency follows from

$$\hat{e}(d_{\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle}, U) = \hat{e}\left(\sum_{i=0}^{m} d^i s P_i, rP\right) = \hat{e}\left(\sum_{i=0}^{m} d^i P_i, P\right)^{rs} = \hat{e}\left(\sum_{i=0}^{m} d^i P_i, sP\right)^{r} = \hat{e}\left(\sum_{i=0}^{m} d^i P_i, P_{pub}\right)^{r}.$$

It is also easy to see that the root PKG can decrypt all the messages using the following algorithm:

Escrow Decrypt: Given a ciphertext $C = \langle U, V \rangle$, a level-1 private key $s$, an address $\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2, H_3 \rangle$,

Step 1: Compute $\langle P_0, \ldots, P_m \rangle = H_1(\mathsf{ID}_1)$ and $d = H_2(\mathsf{ID}_1 || \mathsf{ID}_2)$.

Step 2: Compute $d_{\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle} = \sum_{i=0}^{m} d^i s P_i$. This is the secret key corresponding to the address $\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle$.

Step 3: Compute $g' = \hat{e}(d_{\langle \mathsf{ID}_1, \mathsf{ID}_2 \rangle}, U)$.

Step 4: Compute $M = V \oplus H_3(g')$ and return $M$.

**Theorem 3.3.1** *Let the hash function $H_1, H_2, H_3$ be random oracles. Let $\mathcal{A}$ is an* HID-OWE *attacker that makes at most $n$* $\mathsf{Extract}_2$ *queries in each domain. Suppose that $\mathcal{A}$ has advantage $\varepsilon$ against the scheme* Esc-2-HIDE*. Then there exists an algorithm $\mathcal{B}$ that can solve the BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with advantage at least $\varepsilon / \left(2(q_{K_1} + 2q_{K_2}) q_{H_1} \binom{q_{H_2}}{n} e\right)$, where $q_{K_i}$ is a bound on the number of* $\mathsf{Extract}_i$ *queries, $q_{H_i}$ is a bound on the number of $H_i$ queries and $e$ is the base of the natural logarithm.*

We can then apply the transformation by Fujisaki and Okamoto [21] to obtain an IND-HID-CCA scheme.

We add that in [29], Horwitz and Lynn present a 2-HIDE scheme with key escrow at each level that would be resistant to any amount of collusion at the lower level if we knew how to construct a function with certain algebraic properties.

### 3.3.3 A Full HIDE Scheme

This scheme, called BasicHIDE is due to Gentry and Silverberg [26]. It supports an arbitrary number of levels but the length of the ciphertext, as well as computational complexity of the Encrypt and Decrypt algorithms grows linearly with the number of levels. We also note that any address (at any level) can be used as a public key for encryption.

Root Setup: Given a BDH parameter generator $\mathbb{G}$ and a security parameter $\lambda$,

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P_0 \in G_1^*$.

Step 2: Pick a random $s_\epsilon \in \mathbb{Z}_p^*$ and set $Q_\epsilon = s_\epsilon P_\epsilon$.

Step 3: Pick cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$, $H_2 : G_2^* \to \{0,1\}^n$ for some positive integer $n$.

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = G_1^{*l} \times \{0,1\}^n$ where $l$ is the level of the receiver. The public system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_\epsilon, Q_\epsilon, H_1, H_2 \rangle$. The root secret is $s_\epsilon$.

Lower-level Setup: Given the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_\epsilon, Q_\epsilon, H_1, H_2 \rangle$ , each entity $E_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle}$ other than the root picks a random $s_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle} \in \mathbb{Z}_p^*$ and computes $Q_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle} = s_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle} P_\epsilon$ which it keeps secret.

Extract$_i$: Given the PID tuple $\langle \mathsf{ID}_1, \ldots, \mathsf{ID}_i \rangle$ of one of its children, its private key $\mathsf{mk}_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1} \rangle} = \langle S_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1} \rangle}, Q_\epsilon, Q_{\langle \mathsf{ID}_1 \rangle}, \ldots, Q_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1} \rangle} \rangle$, its secret value $s_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1} \rangle} \in \mathbb{Z}_p^*$ [3] and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2 \rangle$, PKG $E_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1} \rangle}$ computes the private key as follows:

Step 1: Compute $P_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle} = H_1(\mathsf{ID}_1, \ldots, \mathsf{ID}_i) \in \mathbb{G}_1$.

Step 2: Compute $S_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle} = S_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1} \rangle} + s_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1} \rangle} P_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle}$.

Step 3: Returns $\langle S_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle}, Q_\epsilon, Q_{\langle \mathsf{ID}_1 \rangle}, \ldots, Q_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1} \rangle} \rangle$.

The private key corresponding to $\langle \mathsf{ID}_1, \ldots, \mathsf{ID}_i \rangle$ is $\langle S_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle}, Q_\epsilon, Q_{\langle \mathsf{ID}_1 \rangle}, \ldots, Q_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle} \rangle$ (the user $\langle \mathsf{ID}_1, \ldots, \mathsf{ID}_i \rangle$ already knows $Q_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle}$).

Encrypt: Given a plaintext $M \in \mathcal{M}$, a PID-tuple $\langle \mathsf{ID}_1, \ldots, \mathsf{ID}_l \rangle$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2 \rangle$,

Step 1: Compute $P_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i \rangle} = H_1(\mathsf{ID}_1, \ldots, \mathsf{ID}_i) \in \mathbb{G}_1$ for $1 \leq i \leq l$.

Step 2: Compute $g = \hat{e}(Q_\epsilon, P_{\langle \mathsf{ID}_1 \rangle})$.

Step 3: Pick a random $r \in \mathbb{Z}_p^*$.

Step 4: Compute the ciphertext $C = \langle rP_\epsilon, rP_{\langle \mathsf{ID}_1,\mathsf{ID}_2 \rangle}, \ldots, rP_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_l \rangle}, M \oplus H_2(g^r) \rangle$ and return $C$.

[3] For definiteness, if $i = 1$, then the private key is $\mathsf{mk}_\epsilon = \langle S_\epsilon, Q_\epsilon \rangle$ where $S_\epsilon$ is the identity element in $\mathbb{G}_1$ and the secret value is $s_\epsilon$.

Decrypt: Given a ciphertext $C = \langle U_0, U_2, \ldots, U_l, V \rangle$, a private key $\langle S_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_l\rangle}, Q_\epsilon, Q_{\langle \mathsf{ID}_1\rangle}, \ldots, Q_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_l\rangle}\rangle$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2, \rangle$, compute

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_l\rangle})}{\prod_{i=2}^{l} \hat{e}(Q_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1}\rangle}, U_i)}\right) = M$$

and return $M$.

Consistency follows from

$$\begin{aligned}
\frac{\hat{e}(U_0, S_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_l\rangle})}{\prod_{i=2}^{l} \hat{e}(Q_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1}\rangle}, U_i)} &= \frac{\hat{e}(rP_\epsilon, s_\epsilon P_{\langle \mathsf{ID}_1\rangle} + \sum_{i=2}^{l} s_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1}\rangle} P_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i\rangle})}{\prod_{i=2}^{l} \hat{e}(s_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1}\rangle} P_\epsilon, rP_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i\rangle})} \\
&= \hat{e}(P_\epsilon, P_{\langle \mathsf{ID}_1\rangle})^{rs_\epsilon} \frac{\prod_{i=2}^{l} \hat{e}(P_\epsilon, P_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i\rangle})^{rs_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1}\rangle}}}{\prod_{i=2}^{l} \hat{e}(P_\epsilon, P_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_i\rangle})^{rs_{\langle \mathsf{ID}_1,\ldots,\mathsf{ID}_{i-1}\rangle}}} \\
&= \hat{e}(r_\epsilon P_\epsilon, P_{\langle \mathsf{ID}_1\rangle})^r \\
&= \hat{e}(Q_\epsilon, P_{\langle \mathsf{ID}_1\rangle})^r.
\end{aligned}$$

**Theorem 3.3.2** *Let the hash functions $H_1, H_2$ be random oracles. Suppose there is an HID-OWE attacker $\mathcal{A}$ that has advantage $\varepsilon$ in targeting a note in level $t$ in the scheme BasicHIDE for some $t$ and that makes at most $q_{H_2} > 0$ $H_2$ queries and at most $q_E > 0$ extraction queries. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ with advantage at least $(\varepsilon(\frac{t}{e(q_E+t)})^t - \frac{1}{2^n})/q_{H_2}$ and running time $O(time(\mathcal{A}))$.*

Again, by applying the Fujisaki-Okamoto transformation [21], we get an IND-HID-CCA scheme.

It is interesting to note that if there is only one level, then this scheme is identical to the Boneh-Franklin scheme presented in section 3.1.

## 3.4 Escrow Encryption Schemes

Using a bilinear map, it is also possible to design El Gamal-like encryption schemes which offer key escrow. We present two such schemes. The first one, by Boneh and Franklin [5] offers global escrow, the second, by Verheul [51] has simple escrow.

### 3.4.1 Encryption Scheme with Global Escrow

**Setup:** Given a BDH parameter generator $\mathcal{IG}$ and a security parameter $\lambda$

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in G_1^*$.

Step 2: Pick a random $s \in \mathbb{Z}_p^*$ and compute $Q = sP$.

Step 3: Pick a cryptographic hash function $H : \mathbb{G}_2 \to \{0,1\}^n$ for some positive integer $n$.

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$. The system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, Q, H \rangle$. The escrow key is $s \in \mathbb{Z}_p^*$.

**KeyGen:** Given the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, Q, H \rangle$, a user generates his public/private key pair by picking a random $x \in \mathbb{Z}_p^*$ and computing $P_{pub} = xP$. His public key is $P_{pub}$ and his private key is $x$.

**Encrypt:** Given a message $M \in \mathcal{M}$, a public key $P_{pub}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, Q, H \rangle$,

Step 1: Pick a random $r \in \mathbb{Z}_p^*$.

Step 2: Compute $g = \hat{e}(P_{pub}, Q)$.

Step 3: Compute the ciphertext $C = \langle rP, M \oplus H(g^r) \rangle$ and return $C$.

**Decrypt:** Given a ciphertext $C = \langle U, V \rangle$, a private key $x$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, Q, H \rangle$,

Step 1: If $U \notin \mathbb{G}_1$, reject the ciphertext.

Step 2: Compute $g' = \hat{e}(U, xQ)$.

Step 3: Compute $M = V \oplus H(g')$ and return $M$.

**Escrow-decrypt:** Given a ciphertext $C = \langle U, V \rangle$, a public key $P_{pub}$, the escrow key $s$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, Q, H \rangle$,

Step 1: If $U \notin \mathbb{G}_1$, reject the ciphertext.

Step 2: Compute $g' = \hat{e}(U, sP_{pub})$.

Step 3: Compute $M = V \oplus H(g')$ and return $M$.

Consistency for these algorithms is easy to check. It follows from the fact that

$$\begin{aligned}\hat{e}(P_{pub}, Q)^r = \hat{e}(xP, Q)^r = \hat{e}(P, sP)^{rx} = \hat{e}(rP, xsP) = \hat{e}(U, xQ) \\ = \hat{e}(rP, sxP) = \hat{e}(U, sP_{pub}).\end{aligned}$$

A standard argument shows that this scheme is semantically secure in the random oracle model if the BDH assumption holds for $\mathcal{IG}$.

### 3.4.2 Encryption Scheme with Simple Escrow

**Setup:** Given a BDH parameter generator $\mathcal{IG}$ and a security parameter $\lambda$

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in G_1^*$.

Step 2: Pick a cryptographic hash function $H : \mathbb{G}_2 \to \{0,1\}^n$ for some positive integer $n$.

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_2 \times \{0,1\}^n$. The system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H \rangle$.

**Keygen:** Given the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H \rangle$, a user generates his public/private key pair by picking a random $x \in \mathbb{Z}_p^*$ and computing $P_{esc} = xP$ and $y = \hat{e}(P, P_{esc})$. His public key is $y$, his private key is $x$ and the escrow key is $P_{esc}$.

**Encrypt:** Given a plaintext $M \in \mathcal{M}$, a public key $y$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H \rangle$,

Step 1: Pick a random $r \in \mathbb{Z}_p^*$.

Step 2: Compute the ciphertext $C = \langle rP, M \oplus H(y^r) \rangle$ and return $C$.

**Decrypt:** Given a ciphertext $C = \langle U, V \rangle$, the private key $x$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H \rangle$,

Step 1: If $U \notin \mathbb{G}_1$, reject the ciphertext.

Step 2: Compute $g' = \hat{e}(U, P)^x$.

Step 3: Compute $M = V \oplus H(g)$ and return $M$.

**Escrow-decrypt:** Given a ciphertext $C = \langle U, V \rangle$, the escrow key $P_{esc}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H \rangle$,

Step 1: If $U \notin \mathbb{G}_1$, reject the ciphertext.

Step 2: Compute $g' = \hat{e}(U, P_{esc})$.

Step 3: Compute $M = V \oplus H(g')$ and return $M$.

Consistency is again easy to check. It follows from the fact that

$$y^r = g^{rx} = \hat{e}(P, P)^{rx} = \hat{e}(rP, P)^x = \hat{e}(U, P)^x$$
$$= \hat{e}(rP, xP) = \hat{e}(U, P_{esc})$$

Note that here, since the user has access to both the escrow key and the secret key, he can use either decryption algorithms. This scheme is useful if the user wants to use the same secret key for encryption and signature, since the knowledge of the escrow key enables to decrypt messages, but not to sign. Also, we note that in this scheme, the user has explicit control as to whom he gives his escrow key.

# Chapter 4

# Signature Schemes

Bilinear maps have been used to build signature schemes that produce short signatures, and to build identity-based signature schemes. We present two signature schemes that provide short signatures and give complete proofs of their security. We also present one of the many identity-based signature schemes that were proposed and state the theorem about the security of the scheme.

We mention that in [52], Verheul shows how bilinear maps can be used to construct self-blindable credential certificates, and in [35], Lysyanskaya presents a unique signature scheme using bilinear maps. These schemes are not presented here.

## 4.1 Short Signatures

Short digital signatures are always desirable. They are necessary in situations in which humans are asked to manually key in the signature or when working in low-

bandwidth communication environments. They are also useful in general to reduce the communication complexity of any transmission.

We present two signature schemes that provide signatures whose length is approximately 160 bits but that still provide a level of security comparable to 320-bit DSA or ECDSA signatures, or 1000-bit RSA signatures. The first scheme is obtained by modifying the identity-based encryption scheme presented in Section 3.1. The second scheme was designed by Boneh, Lynn and Shacham [6] using an idea of Okamoto and Pointcheval [43]. This second scheme has the advantage of being simpler to implement and it has a tighter security reduction.

### 4.1.1 Definitions

#### Signature Scheme

A *signature scheme* consists of four randomized algorithms: Setup, KeyGen, Sign and Verify.

Setup: takes as input a security parameter and outputs params, the public system parameters of the signature scheme.

KeyGen: takes as input the system parameters params, and returns a public key and a private key.

Sign: takes as input the system parameters params, a private key and a message $M$ and outputs a signature on $M$.

Verify: takes as input the system parameters, params, a public key, and a message-signature pair and outputs valid or invalid.

These algorithms are required to satisfy the standard consistency constraints, namely if params is produced by the Setup algorithm, $K_{pub}$ and $x$ are respectively a public and private key produced by the KeyGen algorithm, then

$$\forall M \in \mathcal{M},\ \mathsf{Verify}(\mathsf{params}, K_{pub}, \langle M, \mathsf{Sign}(\mathsf{params}, x, M)\rangle) = \mathsf{valid}.$$

## Security

Security against adaptive chosen message attack is the standard notion of security for signature schemes. It is defined through the following game between a challenger and an adversary:

**Setup:** The challenger chooses a security parameter $\lambda$ and runs the Setup algorithm. He gives the public system parameters to the adversary and keeps the private key to himself.

**Query Phase:** The adversary issues signing queries $M_1, \ldots, M_n$ where $M_i \in \{0,1\}^*$. These queries can be made adaptively. The challenger responds by running the Sign algorithm with his private key and returns the resulting signature to the adversary.

**Guess:** The adversary outputs a message-signature pair $\langle M, \sigma\rangle$ where $M$ is different from all queries issued in the preceding phase. The adversary wins if $\sigma$ is a valid signature of $M$.

The advantage of an adversary $\mathcal{A}$ against a signature scheme is defined to be the probability that $\mathcal{A}$ produces a valid message-signature pair in the game described

above. A signature scheme is said to be secure against adaptive chosen message attack if no polynomially bounded adversary (in $\lambda$) has non-negligible advantage (in $\lambda$) in this game .

### 4.1.2 Signature Scheme from Identity-Based Encryption

Moni Naor observed that an IBE scheme could be transformed into a public key signature scheme as follows. The public key for the signature scheme are the system parameters of the IBE scheme. The private key for the signature scheme is the master key of the IBE scheme. The signature on a message $M$ is the IBE decryption key for ID $= M$. To verify a signature, choose a random plaintext $M'$ in the plaintext space of the IBE scheme, encrypt it using ID $= M$ as the public key, decrypt the resulting ciphertext using the given signature on $M$ as the decryption key and check that the resulting plaintext is the original plaintext $M'$. If the IBE scheme is IND-ID-CCA, then the signature scheme is secure against adaptive chosen message attack. We show that in fact, it is sufficient to have an ID-OWE IBE scheme to obtain a signature scheme secure against adaptive chosen message attack by proving that the signature scheme obtained by applying the transformation above to the scheme BasicIdent yields a signature scheme secure against adaptive chosen ciphertext attack. The security proof would be similar for any other ID-OWE IBE scheme.

We call this signature scheme BasicSig.

Setup: Given a BDH parameter generator $\mathcal{IG}$ and a security parameter $\lambda$,

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map

$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in G_1^*$.

Step 2: Pick cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some positive integer $n$.

The system parameters are $\mathsf{params} = \langle .\mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2 \rangle$.

KeyGen: Given system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2 \rangle$,

Step 1: Pick a random $x \in \mathbb{Z}_p^*$.

Step 2: Compute $K_{pub} = xP$.

The public key is $K_{pub}$, the private key is $x \in \mathbb{Z}_p^*$.

Sign: Given a message $M \in \{0,1\}^*$, a private key $x$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2 \rangle$,

Step 1: Compute $Q = H_1(M) \in \mathbb{G}_1^*$.

Step 2: Compute $\sigma = xQ$.

The signature on $M$ is $\sigma$.

Verify: Given a message-signature pair $\langle M, \sigma \rangle$, a public key $K_{pub}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2 \rangle$,

Step 1: Compute $Q = H_1(M) \in \mathbb{G}_1^*$.

Step 2: Pick a random $r \in \mathbb{Z}_p^*$.

Step 3: Compute $g = \hat{e}(Q, K_{pub}) \in \mathbb{G}_2^*$.

Step 4: Pick a random $M' \in \{0,1\}^n$.

Step 5: Compute $U = rP$ and $V = M' \oplus H_2(g^r)$.

Step 6: Compute $g' = \hat{e}(\sigma, U)$.

Step 7: Compute $M'' = V \oplus H_2(g')$.

Step 8: If $M'' = M'$ output valid; else output invalid.

Observe that the signature on any message $M \in \{0,1\}^*$ is a unique element of $\mathbb{G}_1$. We will see in Chapter 6 that the elements of $\mathbb{G}_1$ can have a 160 bit representation if $\mathbb{G}_1$ is chosen carefully.

**Theorem 4.1.1** *Let the hash functions $H_1, H_2$ be random oracles. Suppose there is an algorithm $\mathcal{A}$ that can forge a BasicSig signature with probability $\varepsilon$. Then there exists an ID-OWE attacker which has advantage at least $\varepsilon$ against the scheme BasicIdent.*

**Proof** First, the challenger runs the BasicIdent Setup algorithm to get system parameters params $= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, P_{pub}, H_1, H_2 \rangle$ and a master key $s$. The challenger then gives params to $\mathcal{B}$ and keeps $s$ to himself.

Algorithm $\mathcal{B}$ mounts its attack against BasicIdent by interacting with $\mathcal{A}$ as follows.

**Setup:** $\mathcal{B}$ gives $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, n, P, H_1, H_2 \rangle$ to $\mathcal{A}$ as the system parameters and $K_{pub} = P_{pub}$ as the public key of the signature scheme.

**Query Phase:** Let $M_i$ be a signing query issued by algorithm $\mathcal{A}$. $\mathcal{B}$ responds to this query by relaying the key extraction query $\langle M_i \rangle$ to the challenger. The

challenger responds by returning the private key $d_M$ corresponding to the public key $M_i$. $\mathcal{B}$ returns $d_M$ to $\mathcal{A}$.

**Guess:** Eventually, algorithm $\mathcal{A}$ produces a message-signature pair $\langle M, \sigma \rangle$ where $M$ is not equal to any signing query previously issued. Algorithm $\mathcal{B}$ gives $M$ to the challenger as the public key on which it wishes to be challenged. The challenger returns a ciphertext $C = \langle U, V \rangle$ which is the encryption of a random plaintext under the public key $M$. $\mathcal{B}$ runs the BasicIdent decryption algorithm on $C$ using $\sigma$ as the private key and returns the plaintext obtained as its guess for the decryption of $C$.

It is easy to see that $\mathcal{A}$'s view is identical to that in the real attack since all the responses to signing queries are valid signatures. Therefore, $\mathcal{A}$ produces a valid message-signature pair with probability $\varepsilon$. Given the private key corresponding to the public key $M$, $\mathcal{B}$ will obviously be able to decrypt the ciphertext correctly. Hence, $\mathcal{B}$ has advantage at least $\varepsilon$ against the scheme BasicIdent. ∎

We note that the HIBE scheme presented in Section 3.3 can also be transformed into a signature scheme using the same idea.

### 4.1.3 The Gap Diffie-Hellman Signature Scheme

This signature scheme was described implicitly by Okamoto and Pointcheval in [43]. However, the proof of security and implementation details come from a paper by Boneh, Lynn and Shacham [6].

This scheme can be implemented in any *Gap Diffie-Hellman group* (GDH

group), i.e. in any group in which the Decision Diffie-Hellman problem is easy to solve, but the Computational Diffie-Hellman problem is intractable.

In [4], Boldyreva shows that this scheme can be modified to construct threshold signature, multisignature and blind signature schemes. Lynn and Shacham also mentioned in the Rump session of Crypto 2002 that it can also be modified to construct aggregate signature and ring signature schemes.

Setup: Given a BDH parameter generator $\mathcal{IG}$ and a security parameter $\lambda$,

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in \mathbb{G}_1^*$.

Step 2: Pick a cryptographic hash function $H : \{0,1\}^* \to \mathbb{G}_1^*$.

The system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P, H \rangle$.

KeyGen: Given the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P, H \rangle$, pick a random $x \in \mathbb{Z}_p^*$ and compute $K_{pub} = xP$. The public key is $K_{pub}$ and the private key is $x$.

Sign: Given a message $M \in \{0,1\}^*$, a private key $x$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P, H \rangle$,

Step 1: Compute $R = H(M)$.

Step 2: Compute $\sigma = xR$.

The signature on $M$ is $\sigma$.

Verify: Given a message-signature pair $\langle M, \sigma \rangle$, a public key $K_{pub}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P, H \rangle$,

Step 1: Compute $R = H(M)$.

Step 2: Compute $g_1 = \hat{e}(P, \sigma)$ and $g_2 = \hat{e}(K_{pub}, R)$.

Step 3: If $g_1 = g_2$, return valid; else return invalid.

Consistency is easy to check: if the scheme is executed correctly, then

$$\hat{e}(P, \sigma) = \hat{e}(P, xH(M)) = \hat{e}(xP, H(M)) = \hat{e}(K_{pub}, H(M))$$

In a general GDH group, the verifier would check that $\langle P, K_{pub}, H(M), \sigma \rangle$ is a Diffie-Hellman tuple.

The signature on a message $M \in \{0,1\}^*$ is a unique element of $\mathbb{G}_1$. We will see in Chapter 6 that the elements of $\mathbb{G}_1$ can have a 160-bit representation if $\mathbb{G}_1$ is chosen carefully.

## Proof of Security

**Theorem 4.1.2** *Let the hash function $H$ be a random oracle. Suppose there is an algorithm $\mathcal{A}$ that makes at most $q_S$ queries to the signing oracle and has advantage $\varepsilon$ against the scheme above. Then there exists an algorithm $\mathcal{B}$ that solves the CDH problem in the groups $\mathbb{G}_1$ generated by $\mathcal{IG}$ with advantage at least $\frac{\varepsilon - 1/(p-1)}{e(q_S+1)}$ and runs in time $O(time(\mathcal{A}))$.*

**Proof** $\mathcal{B}$ is given as input the BDH parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p \rangle$ with $|\mathbb{G}_1| = p$ and an instance of the CDH problem in $\mathbb{G}_1$, $\langle P, aP, bP \rangle = \langle P, P_1, P_2 \rangle$, where $P$ is a random generator of $\mathbb{G}_1$ and $a, b$ are random elements of $\mathbb{Z}_p$.

$\mathcal{B}$ computes $abP$ by interacting with $\mathcal{A}$ as follows:

**Setup:** $\mathcal{B}$ gives algorithm $\mathcal{A}$ the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, H \rangle$ and public key $K_{pub} = P_1$, where $H$ is a random oracle controlled by $\mathcal{B}$.

***H*-queries:** Algorithm $\mathcal{B}$ maintains a list of tuples $\langle M_j, R_j, r_j, c_j \rangle$ containing information about the previous queries to oracle $H$. We call this list $H^{list}$. It is initially empty. $\mathcal{B}$ responds to $\mathcal{A}$'s $H$-query $M_i$ as follows:

1. If the query $M_i$ already appears in $H^{list}$ in a tuple $\langle M_i, R_i, r_i, c_i \rangle$, return $R_i$.
2. Otherwise, $\mathcal{B}$ generates a random $coin \in \{0, 1\}$ so that $\Pr[coin = 0] = \delta$ where $\delta = 1 - \frac{1}{q_S+1}$ (the reason for this choice will be given later).
3. $\mathcal{B}$ picks a random $r \in \mathbb{Z}_p^*$. If $coin = 0$, compute $R_i = rP \in \mathbb{G}_1$. If $coin = 1$, compute $R_i = r_i P + P_2 \in \mathbb{G}_1$.
4. $\mathcal{B}$ adds the tuple $\langle M_i, R_i, r, coin \rangle$ to $H^{list}$ and returns $R_i$.

Note that in both cases, the distribution of $R_i$ is uniform in $\mathbb{G}_1^*$ and independent of $\mathcal{A}$'s view.

**Signing queries:** Algorithm $\mathcal{B}$ responds to a signing query $M_i$ issued by $\mathcal{A}$ as follows:

1. If $\mathcal{A}$ had previously issued the query $M_i$ to oracle $H$, find the tuple $\langle M_i, R_i, b_i, coin_i \rangle$ in $H^{list}$, otherwise, create such a tuple using the procedure described above and add it to $H^{list}$. If $coin_i = 1$, then $\mathcal{B}$ reports failure and terminates. The attempt to solve the CDH problem has failed.

2. Otherwise, $coin_i = 0$, so $R_i = r_i P$. Return $\sigma_i = r_i P_1 \in \mathbb{G}_1^*$ to $\mathcal{A}$. Observe that $\sigma_i$ is a valid signature on $M_i$ since $\langle P, P_1, H(M_i), \sigma_i \rangle = \langle P, aP, r_i P, r_i aP \rangle$ is a valid Diffie-Hellman tuple.

**Guess:** Eventually, $\mathcal{A}$ produces a valid message-signature pair $\langle M', \sigma' \rangle$. If $M'$ does not appear in any element of $H^{list}$ or if $M'$ appears in a tuple $\langle M', R', r', c' \rangle$ of $H^{list}$ with $c' = 0$, then $\mathcal{B}$ reports failure and terminates. The attempt to solve the CDH problem has failed. Otherwise, if $M'$ appears in a tuple $\langle M', R', r', c' \rangle$ of $H^{list}$ with $c' = 1$, then $\mathcal{B}$ outputs $\sigma' - r' P_1$.

If algorithm $\mathcal{A}$ successfully forges a signature on a message $M'$ which appears in a tuple $\langle M', R', r', c' \rangle$ of $H^{list}$ with $c' = 1$, then $\langle P, P_1, H(M'), \sigma' \rangle = \langle P, aP, (r' + b)P, \sigma' \rangle$ is a valid Diffie-Hellman tuple, so $\sigma - r' P_1 = a(r' + b)P - r' aP = abP$.

Also, it is easy to see that if $\mathcal{B}$ does not abort when simulating $\mathcal{A}$, then algorithm $\mathcal{A}$'s view is identical to its view in the real attack since the distribution of the output of $H$ is uniform in $\mathbb{G}^*$ and the signatures returned by the signing oracle are valid. So, if $\mathcal{B}$ does not abort when simulating $\mathcal{A}$, $\Pr[\mathcal{A}$ outputs a valid message-signature pair$] = \varepsilon$.

But even if $\mathcal{A}$ outputs a valid message-signature pair, $\mathcal{B}$ will succeed only if $M'$ is in $H^{list}$ and if the corresponding tuple has $c' = 1$. The probability $\mathcal{A}$ outputs a valid message-signature pair with $M'$ in $H^{list}$ is at least $\varepsilon - 1/(p-1)$ because if $M'$ is not in $H^{list}$ then the signature on $M$ is independent from $\mathcal{A}$'s view (since $H(M')$ is a random element of $\mathbb{G}_1^*$). Given that $M'$ is in $H^{list}$, the probability that $c' = 1$ is $(1 - \delta)$. Therefore, the probability that $\mathcal{B}$ succeeds given that it does not abort when simulating $\mathcal{A}$ is $(\varepsilon - 1/(p-1))(1 - \delta)$.

It remains to calculate the probability that $\mathcal{B}$ does not abort during the simulation. For each signing query, the probability that $\mathcal{B}$ does not abort is $\delta$. Since $\mathcal{A}$ makes at most $q_S$ such queries, the probability that $\mathcal{B}$ does not abort when simulating $\mathcal{A}$ is at least $\delta^{q_S}$.

Therefore, the probability that $\mathcal{B}$ succeeds is $\delta^{q_S}(1-\delta)(\varepsilon - 1/(p-1))$. The value $\delta(\lambda) = 1 - \frac{1}{q_S+1}$ was chosen in order to maximize $\delta^{q_S}(1-\delta)$. Similarly as in lemma 3.1.2, $(1-\frac{1}{q_S+1})^{q_S} \geq \frac{1}{e}$. Hence, the probability that $\mathcal{B}$ succeeds in solving the CDH problem is at least $\frac{\varepsilon - 1/(p-1)}{e(q_S+1)}$. ∎

## 4.2 Identity-Based Signature Schemes

Bilinear maps can also be used to build identity-based signatures. The concept of identity-based signatures was also presented by Shamir in [48]. However, contrary to identity-based encryption, satisfactory identity-based signature schemes have been found not long after ([18, 17]).

We present in this section an identity-based signature scheme by Cha and Cheon [10]. We note that other identity-based signature schemes using bilinear maps were also presented by Paterson [44] and Hess [28]. However, no proof of security is provided for the scheme in [44]. In [28], two identity-based signature schemes are presented, a proof of security is provided for the first one in the case of fixed $\mathsf{ID}$, only a heuristic proof is provided for the second one, which was later broken by Cheon [12].

### 4.2.1 Definitions

#### Signature Scheme

An *identity-based signature scheme* (IBS scheme) consists of four randomized algorithms: Setup, Extract, Sign and Verify.

Setup: takes as input a security parameter and outputs params and master-key. The system parameters will be publicly known, while the master-key is known only to the PKG.

Extract: takes as input the system parameters params, a public key ID and the master key, and returns the private key $d_{\mathsf{ID}}$ corresponding to ID.

Sign: takes as input the system parameters params, a private key $d_{\mathsf{ID}}$ and a message $M$ and outputs a signature on $M$.

Verify: takes as input the system parameters, params, a public key ID, and a message-signature pair and outputs valid or invalid.

These algorithms are required to satisfy the standard consistency constraints, namely if params is produced by the Setup algorithm, ID and $d_{\mathsf{ID}}$ are respectively a public key and the corresponding private key produced by the Extract algorithm, then

$$\forall M \in \mathcal{M},\ \mathsf{Verify}(\mathsf{params}, \mathsf{ID}, \langle M, \mathsf{Sign}(\mathsf{params}, d_{\mathsf{ID}}, M)\rangle) = \mathsf{valid}.$$

## Security

Again, we have to strengthen a bit the security definition to allow the adversary to obtain the private key corresponding to public keys of his choice. The security of an identity-based signature scheme is defined through the following game between a challenger and an adversary:

**Setup:** The challenger chooses a security parameter $\lambda$ and runs the Setup algorithm. He gives the public system parameters to the adversary and keeps the private key to himself.

**Query Phase:** The adversary issues queries $q_1, \ldots, q_m$ where each query is one of:

- Extraction query $\langle \mathsf{ID}_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $d_{\mathsf{ID}_i}$ corresponding to the public key $\mathsf{ID}_i$ and sends it to the adversary.
- Signing query $\langle \mathsf{ID}_i, M_i \rangle$. The challenger responds by running algorithm Extract to generate the private key $d_{\mathsf{ID}_i}$ corresponding to the public key $\mathsf{ID}_i$, uses this private key to sign the message $M_i$ and returns the resulting signature to the adversary.

**Guess:** The adversary outputs $\langle \mathsf{ID}, M, \sigma \rangle$ where ID is a public key, $M$ is a message and $\sigma$ is a signature, such that the adversary did not issue an extraction query on ID and did not issue a signing query on $\langle \mathsf{ID}, M \rangle$. The adversary wins if $\sigma$ is a valid signature on $M$ for user ID.

The advantage of an adversary $\mathcal{A}$ against a signature scheme is defined to be the probability that $\mathcal{A}$ produces a valid message-signature pair for some ID in the game

described above. An identity-based signature scheme is said to be secure against adaptive chosen message attack if no polynomially bounded adversary (in $\lambda$) has non-negligible advantage (in $\lambda$) in the game described above.

### 4.2.2 The Scheme

Setup: Given a BDH parameter generator $\mathcal{IG}$ and a security parameter $\lambda$,

Step 1: Run $\mathcal{IG}$ with input $1^\lambda$ to get two groups $\mathbb{G}_1, \mathbb{G}_2$ and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $p$ be the order of $\mathbb{G}_1$ and $\mathbb{G}_2$. Pick an arbitrary generator $P \in \mathbb{G}_1^*$.

Step 2: Pick a random $s \in \mathbb{Z}_p^*$ and compute $P_{pub} = sP$.

Step 3: Pick cryptographic hash functions $H_1 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \mathbb{G}_1$.

The public system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P, P_{pub}, H_1, H_2 \rangle$.

Extract: Given a string $\mathsf{ID} \in \{0,1\}^*$, the master key $s$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P, P_{pub}, H_1, H_2 \rangle$,

Step 1: Compute $Q_{\mathsf{ID}} = H_2(\mathsf{ID})$.

Step 2: Compute $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ and return $d_{\mathsf{ID}}$.

Sign: Given a message $M \in \{0,1\}^*$, a private key $d_{\mathsf{ID}}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P, P_{pub}, H_1, H_2 \rangle$,

Step 1: Compute $Q_{\mathsf{ID}} = H_2(\mathsf{ID})$.

Step 2: Pick a random $r \in \mathbb{Z}_p^*$ and compute $U = rQ_{\mathsf{ID}}$.

Step 3: Compute $h = H_1(M, U)$.

Step 4: Compute $V = (r + h \bmod p)d_{\mathsf{ID}}$.

The signature on message $M$ is $\langle U, V \rangle$.

Verify: Given a message $M \in \{0,1\}^*$, a signature $\langle U, V \rangle$, a public key $\mathsf{ID}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P, P_{pub}, H_1, H_2 \rangle$,

Step 1: Compute $h = H_1(M, U)$.

Step 2: Compute $Q_{\mathsf{ID}} = H_2(\mathsf{ID})$.

Step 3: Compute $Q = U + hQ_{\mathsf{ID}}$.

Step 4: Compute $g = \hat{e}(P, V)$ and $g' = \hat{e}(P_{pub}, Q)$.

Step 5: If $g = g'$, output valid; else output invalid.

Consistency is easily proved as follows. If $\langle U, V \rangle$ is a valid signature on $M$, then $U = rQ_{\mathsf{ID}}$ and $V = (r+h)d_{\mathsf{ID}}$, where $Q_{\mathsf{ID}} = H_2(\mathsf{ID})$, $h = H_1(M, U)$ and $r \in \mathbb{Z}_p^*$. So,

$$\hat{e}(P, V) = \hat{e}(P, s(r+h)Q_{\mathsf{ID}}) = \hat{e}(sP, (r+h)Q_{\mathsf{ID}}) = \hat{e}(P_{pub}, U + hQ_{\mathsf{ID}}).$$

In a general GDH group, the verifier would check that $\langle P, P_{pub}, Q, V \rangle$ is a Diffie-Hellman tuple.

**Theorem 4.2.1** *Let the hash functions $H_1, H_2$ be random oracles. Suppose there is an algorithm $\mathcal{A}$ which queries oracles $H_1$, $H_2$ and the signing oracle at most $q_{H_1}$, $q_{H_2}$ and $q_S$ times respectively and that can forge an IDsig signature in time $t$ and*

*with probability* $\varepsilon \geq 10(q_S+1)(q_S+q_{H_1})q_{H_2}/(p-1)$ *Then there is an algorithm* $\mathcal{B}$ *that solves the CDH problem in* $\mathbb{G}_1$ *with advantage* $\varepsilon' \geq 1/9$ *and in time* $t' \leq \frac{23q_{H_1}q_{H_2}t}{\varepsilon(1-\frac{1}{p})}$.

# Chapter 5

# Key Agreement Schemes

In this section, we present Joux's famous one round tripartite key agreement protocol [30]. We note that this protocol, being non authenticated, suffers from the same shortcomings as the original Diffie-Hellman protocol. Therefore, we also present an authenticated tripartite key agreement protocol [1] which seems to prevent most of the security problems of Joux's protocol.

We also note that Smart presents in [50] a two party identity-based authenticated key agreement protocol. The advantage of an identity-based key agreement protocol is that it eliminates the need for a trusted fourth party when running the protocol (a fourth party is usually necessary to obtain authenticated protocols). That protocol is not presented here.

## 5.1 Joux's One Round Tripartite Key Agreement Protocol

In [30], Joux proposed a very simple protocol with which three parties, $A$, $B$ and $C$, can establish a secret session key. The protocol goes as follows:

Let $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, P \rangle$ be BDH parameters.

- $A$ picks a random $a \in \mathbb{Z}_p$, computes $aP$ and sends $aP$ to $B$ and $C$.
- $B$ picks a random $b \in \mathbb{Z}_p$, computes $bP$ and sends $bP$ to $A$ and $C$.
- $C$ picks a random $c \in \mathbb{Z}_p$, computes $cP$ and sends $cP$ to $A$ and $B$.

The order in which these steps are executed is not important. After these three steps are executed, $A$ computes $K_A = \hat{e}(bP, cP)^a$, $B$ computes $K_B = \hat{e}(aP, cP)^b$ and $C$ computes $K_C = \hat{e}(aP, bP)^c$. It is easy to see that, by bilinearity of $\hat{e}$, all these are equal to $K_{ABC} = \hat{e}(P, P)^{abc}$. It is easy to see that a passive adversary would have to solve an instance of the BDH problem to compute the shared secret obtained by running the protocol.

### 5.1.1 Man-in-the-Middle Attack on Joux's Protocol

Suppose that $D$ is able to intercept $A$'s communications with $B$ and $C$, impersonating $A$ to $B$ and $C$ and impersonating $B$ and $C$ to $A$. Then $D$ can execute the following man-in-the-middle attack on Joux's protocol:

1. $D$ picks random $\delta_1, \delta_2, \delta_3 \in \mathbb{Z}_p$.

2. $D$ intercepts $aP$ from $A$ and forwards $\delta_1 P$ to $B$ and $C$.

3. $D$ intercepts $bP$ from $B$ and forwards $\delta_2 P$ to $A$.

4. $D$ intercepts $cP$ from $C$ and forwards $\delta_3 P$ to $A$.

At the end of this attack, $D$ has agreed a key $\hat{e}(P,P)^{\delta_1 bc}$ with $B$ and $C$, and has agreed a key $\hat{e}(P,P)^{a\delta_2\delta_3}$ with $A$. In subsequent communications, $D$ can keep impersonating $A$ to $B$, $C$, and $B$, $C$ to $A$ using these keys.

This attack can easily be extended when the adversary has total control of the network, so that $D$ can share a separate session key with each user and can impersonate any user to any other user.

## 5.2 Authenticated Tripartite Key Agreement Protocol

The one round tripartite key agreement protocol can be modified to include authentication so that, among other things, the man-in-the-middle attack described above no longer works.

In [1], Al-Riyami and Paterson present four authenticated three party key agreement protocols. These protocols are modeled on the MTI protocols [37] and on the MQV protocol [32]. In this section, we describe the protocol modeled on the MQV protocol and give some heuristic evidence of its security.

### 5.2.1 Security Goals and Desired Attributes

The definition of secure authenticated two party key agreement protocol that is now generally accepted is that given by Canetti and Krawczyk in [8]. However, describing this notion of security and extending it to three party would be quite lengthy and the protocol we describe most likely would not satisfy it. Therefore, we only give informal definitions of the security goals and desirable security attributes for an authenticated key agreement protocol.

Let $A$, $B$ and $C$ be honest entities. We say that a key agreement protocol provides *implicit key authentication* (of $B$, $C$ to $A$) if entity $A$ is assured that no entities other than $B$ and $C$ can compute the shared secret after the execution of the protocol. Note that this does not mean that $A$ is assured that $B$ and $C$ actually computed this shared secret. A key agreement protocol that provides implicit key authentication to all parties is called an *authenticated key agreement (AK)* protocol.

We say that a key agreement protocol provides *key confirmation* (of $B$, $C$ to $A$) if, after the execution of the protocol, entity $A$ is assured that entities $B$ and $C$ have correctly computed the shared secret. If both implicit key authentication and key confirmation (of $B$, $C$ to $A$) are provided, then we say that the key agreement protocol provides *explicit key authentication* (of $B$, $C$ to $A$). A key agreement protocol that provides explicit key authentication to all parties is called an *authenticated key agreement with key confirmation (AKC)* protocol.

In addition to implicit key authentication and key confirmation, a number of desirable security attributes of AK and AKC protocols have been found:

**known-key security:** Each time the protocol is run, the resulting shared secret

should be different. The knowledge of the shared secret produced by previous runs of the protocol should not enable an adversary to prevent the protocol from achieving its goals.

**forward secrecy:** If the long-term private key of one or more entities are compromised, the secrecy of the shared secret produced by honest entities in previous runs of the protocol should not be affected.

**no key compromise impersonation:** Clearly, the knowledge of the long-term private key of one of the entities enables an adversary to impersonate this entity. However, it should not be possible for this adversary to impersonate any other entity to $A$ (actually, it should not be possible for this adversary to impersonate any other entity at all).

**no unknown key-share:** It should not be possible for an adversary to convince two of the three parties, say $A$ and $B$ that they share their secret key with an entity $D$ when they actually share it with entity $C$, while entity $C$ (correctly) believes the key is shared with $A$ and $B$. [1]

**key control:** It should not be possible for any entity to force the shared secret to a preselected value.

Further discussion on these attributes can be found in Chapter 12 of [41].

[1] In this situation, whenever entities $A$ or $B$ would receive a message from $C$, they would believe the message came from $D$.

### 5.2.2 The Protocol

We now describe one of the protocols given by Al-Riyami and Paterson (TAK-4 in [1]) and give some evidence indicating it should satisfy the security properties given above.

The protocol requires a certification authority (CA) to provide certificates which binds users' identities to long-term keys. The certificate for user $A$ has the form:

$$Cert_A = (\mathcal{I}_A, \mu_A, P, sign_{\mathrm{CA}}(\mathcal{I}_A \| \mu_A \| P))$$

where $\|$ denotes string concatenation, $\mathcal{I}_A$ is the identity string of $A$ and $sign_{\mathrm{CA}}(S)$ denotes the CA's signature on string $S$. $\mu_A = xP$ is $A$'s long-term public key and $x \in \mathbb{Z}_p$ is $A$'s long-term secret key. Element $P$ is included to specify which element is used to construct $\mu_A$ and the short-term public values. Similarly, $Cert_B$ and $Cert_C$ are the certificates for $B$ and $C$, with $\mu_B = yP$ and $\mu_C = zP$ as their long-term public keys.

Let $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ be BDH parameters and let $H : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p$ be a cryptographic hash function.

- $A$ picks a random $a \in \mathbb{Z}_p$, computes $aP$ and sends $\langle aP, Cert_A \rangle$ to $B$ and $C$.
- $B$ picks a random $b \in \mathbb{Z}_p$, computes $bP$ and sends $\langle bP, Cert_B \rangle$ to $A$ and $C$.
- $C$ picks a random $c \in \mathbb{Z}_p$, computes $cP$ and sends $\langle cP, Cert_C \rangle$ to $A$ and $B$.

The order in which these steps are executed is not important. After these three steps are executed,

- $A$ computes $K_A = \hat{e}(bP + H(bP, yP)yP, cP + H(cP, zP)zP)^{a+H(aP,xP)x}$.
- $B$ computes $K_B = \hat{e}(aP + H(aP, xP)xP, cP + H(cP, zP)zP)^{b+H(bP,yP)y}$.
- $C$ computes $K_C = \hat{e}(aP + H(aP, xP)xP, bP + H(bP, yP)yP)^{c+H(cP,zP)z}$.

It is easy to find that, by bilinearity of $\hat{e}$, these are all equal to

$$K_{ABC} = \hat{e}(P, P)^{(a+H(aP,xP)x)(b+H(bP,yP)y)(c+H(cP,zP)z)}.$$

Typically, one would then use a key derivation function $H' : \mathbb{G}_2 \to \{0,1\}^n$ for some $n$ (usually a hash function) and each user would then use $H'(K_{ABC})$ as their session key.

### Security

First, since the protocol is very closely related to the MQV protocol, we would expect it to offer the same security and have the same shortcomings as the original MQV protocol.

Note that without the knowledge of both the long-term and short-term secret key of a user, it does not seem possible for a fourth party to compute the value of the shared secret. This gives us implicit key authentication.

The protocol also seems to offer perfect forward security, since the shared secret includes the component $\hat{e}(P, P)^{abc}$. So even if the adversary can obtain all three long-term secret keys, we expect he would still have to solve an instance of the BDH problem to obtain the shared secrets obtained in previous runs of the protocol.

Key compromise impersonation attacks also seem unlikely since the knowledge of the long-term key is required to impersonate a user.

The unknown key share attack against the MOV scheme does not seem to work for this protocol because of the use of the hash function to compute the shared secret.

We refer the reader to [1] for arguments that most known attacks on key agreement schemes do not seem to apply to this scheme, which provides some additional evidence about the security of the protocol.

Explicit key authentication can easily be added to this protocol by using standard key derivation and MAC techniques. The resulting protocol would then require 6 broadcasts over 2 rounds (each party simultaneously broadcasts its MAC after computing the shared secret) or 5 broadcasts over 3 rounds ($C$ waits until he receives short-term and long-term public keys from $A$ and $B$, computes the shared secret and then sends his short-term and long-term public key, together with his MAC in one broadcast, then $A$ and $B$ compute the shared secret and simultaneously broadcast their MAC).

# Chapter 6

# Implementation of Bilinear Maps : the Weil and Tate Pairings

In this chapter, we show how bilinear maps can be efficiently implemented using the Weil or the Tate pairings in supersingular elliptic curves.

We first present a quick introduction to elliptic curves, then define the Weil and Tate pairings and show how they can be modified to become bilinear maps. Most of the results presented in the introduction to elliptic curves come from [39] and [16]. Elementary proofs of most of these results can be found in [16] or [11]. A reader with a good background in algebraic geometry may prefer to consult [49].

## 6.1 Introduction to Elliptic Curves

**Definition** Let $\mathbb{F}$ be a field with algebraic closure $\overline{\mathbb{F}}$.

- An *elliptic curve* over $\mathbb{F}$ is given by a Weierstrass equation

$$E/\mathbb{F} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

  where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ such that there are no singular points (a point $(x_0, y_0) \in \overline{\mathbb{F}}$ is said to be singular if, for $F(x, y) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6$, $F(x_0, y_0) = 0$, $\frac{\partial F}{\partial x}(x_0, y_0) = 0$ and $\frac{\partial F}{\partial x}(x_0, y_0) = 0$). The set of points of an elliptic curve is the set of solutions to its defining equation together with the point at infinity $\mathcal{O}_E$.

- For an extension $\mathbb{K}$ of $\mathbb{F}$, a finite point $(a, b)$ on the curve $E$ is called a $\mathbb{K}$-*rational point* if $(a, b) \in \mathbb{K}^2$. The point at infinity is $\mathbb{K}$-rational for any field $\mathbb{K}$. The set of all $\mathbb{K}$-rational points is denoted by $E(\mathbb{K})$.

- We denote $E(\overline{\mathbb{F}})$ by $E$. [1]

- The points $P \in E \setminus \{\mathcal{O}_E\}$ are called *finite points*.

From now on, since we are only interested in elliptic curves defined over finite fields, we will have $\mathbb{F} = \mathbb{F}_q$ ($q = p^m$ for some prime $p$)

### 6.1.1 The Group Law and Group Structure

Let $E$ be an elliptic curve defined over some finite field $\mathbb{F}_q$ ($q = p^m$). We can define a group law on $E$ as follows. Let $P$ and $Q$ be points on $E$. Let $l_1$ be the line through $P$ and $Q$ (if $P = Q$, then $l_1$ is taken to be the tangent line to $E$ at $P$; if

[1] So we use the symbol $E$ to denote the elliptic curve, its defining equation and $E(\overline{\mathbb{F}})$.

one of $P$ or $Q$ is $\mathcal{O}_E$, then $l_1$ is the vertical line through the other point; if $P = Q = \mathcal{O}_E$, $l_1$ is the 'line at infinity'). Then $l_1$ intersects $E$ at one other point, say $R$ (points tangent to the line are counted twice and the line at infinity has a triple intersection at the point at infinity). Let $l_2$ be the line through $R$ and $\mathcal{O}_E$. Then $l_2$ intersects $E$ at a third point, which we define to be $P + Q$.

It can be shown that $(E, +)$ is an abelian group, i.e. the operation above is associative, commutative, has a neutral element ($\mathcal{O}_E$) and every point has an inverse. Algebraic formulas can easily be derived to compute this group law [16]. Further, for any extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$, if $P$ and $Q$ are $\mathbb{F}_{q^k}$-rational points, then so is $P + Q$. Therefore, $(E(\mathbb{F}_{q^k}), +)$ is an abelian group for any extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$.

The following two theorems give us the possible orders of the groups of points of an elliptic curve defined over a finite field.

**Theorem 6.1.1** (Hasse's Theorem) *If $E$ is an elliptic curve defined over $\mathbb{F}_q$, then*

$$\#E(\mathbb{F}_q) = q + 1 - t, \text{ where } |t| \leq 2\sqrt{q}.$$

**Theorem 6.1.2** *Let $q = p^m$. Then there exists an elliptic curve $E/\mathbb{F}_q$ where $\#E(\mathbb{F}_q) = q + 1 - t$ if and only if one of the following holds:*

1. *$t^2 \leq 4q$ and $p \nmid t$*

2. *$m$ is odd and*

   *(a) $t = 0$*

   *or (b) $t^2 = 2q$ and $p = 2$*

   *or (c) $t^2 = 3q$ and $p = 3$*

*3. $m$ is even and*

*(a)* $t^2 = 4q$

*or (b)* $t^2 = q$ *and* $p \not\equiv 1 \pmod 3$

*or (c)* $t = 0$ *and* $p \not\equiv 1 \pmod 4$.

The order of an elliptic curve defined over a finite field can be computed in polynomial time.

**Definition** We say that an elliptic curve $E$ defined over $\mathbb{F}_q$ is *supersingular* if $p \mid t$ where $q = p^m$ and $t = q + 1 - \#E(\mathbb{F}_q)$. If $p \nmid t$, then we say $E$ is *non-supersingular*.

From the previous theorem, we get the following.

**Corollary 6.1.3** *Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ with $\#E(\mathbb{F}_q) = q+1-t$. Then $E$ is supersingular if and only if $t^2 = 0, q, 2q, 3q$ or $4q$.*

If we know the order of an elliptic curve $E$ over $\mathbb{F}_q$, then it is possible to compute its order over any finite extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ using the following theorem.

**Theorem 6.1.4** *Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ with $t = q+1-\#E(\mathbb{F}_q)$. Let $\alpha$ and $\beta$ be the complex roots of the polynomial $T^2 - tT + q \in \mathbb{Z}[T]$. Then $\#E(\mathbb{F}_{q^k}) = q^k + 1 - \alpha^k - \beta^k$.*

The next theorem gives the group type of the group of points of an elliptic curve $E(\mathbb{F}_q)$.

**Theorem 6.1.5** *Let $E$ be an elliptic curve defined over $\mathbb{F}_q$. Then $E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$, where $n_2 \mid n_1$ and $n_2 \mid q - 1$.*

Given the order of the group of points of an elliptic curve $E(\mathbb{F}_q)$, it is possible to compute its group type in polynomial time. It is especially easy if the elliptic curve is supersingular, in which case we can use the following result.

**Theorem 6.1.6** *Let $E$ be a supersingular elliptic curve defined over $\mathbb{F}_q$ and let $t = q + 1 - \#E(\mathbb{F}_q)$.*

1. *If $t^2 = q, 2q$ or $3q$, then $E(\mathbb{F}_q)$ is cyclic.*
2. *If $t^2 = 4q$, then $E(\mathbb{F}_q) \cong \mathbb{Z}_{\sqrt{q}-1} \oplus \mathbb{Z}_{\sqrt{q}-1}$ if $t = 2\sqrt{q}$ and $E(\mathbb{F}_q) \cong \mathbb{Z}_{\sqrt{q}+1} \oplus \mathbb{Z}_{\sqrt{q}+1}$ if $t = -2\sqrt{q}$.*
3. *If $t = 0$ and $q \not\equiv 3 \pmod 4$, then $E(\mathbb{F}_q)$ is cyclic. If $t = 0$ and $q \equiv 3 \pmod 4$, then either $E(\mathbb{F}_q)$ is cyclic or $E(\mathbb{F}_q) \cong \mathbb{Z}_{(q+1)/2} \oplus \mathbb{Z}_2$*

**Definition** Let $E$ be an elliptic curve defined over $\mathbb{F}_q$.

- The *order* of a point $P \in E$ is the least positive non-zero integer $n$ such that $nP = \mathcal{O}_E$. If $P$ is a $\mathbb{F}_{q^k}$-rational point, then the order of $P$ is a divisor of $\#E(\mathbb{F}_{q^k})$.
- A point $P \in E$ is called an *$n$-torsion* point if $nP = \mathcal{O}_E$.
- We denote the subgroup of $n$-torsion points of $E$ (or $E(\mathbb{F}_{q^k})$) by $E[n]$ (or $E(\mathbb{F}_{q^k})[n]$).

**Theorem 6.1.7** *Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ and let $n$ be a positive integer such that $\gcd(q, n) = 1$. Then $E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n$.*

### 6.1.2 Function Field of an Elliptic Curve

**Definition** Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ and let $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ be its defining equation. Let $F(x, y) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 \in \mathbb{F}_q[x, y]$.

- The *coordinate ring* $\mathbb{F}_q[E]$ of $E$ over $\mathbb{F}_q$ is the integral domain $\mathbb{F}_q[E] = \mathbb{F}_q[x, y]/(F)$. Similarly, we define $\overline{\mathbb{F}}_q[E] = \overline{\mathbb{F}}_q[x, y]/(F)$, the coordinate ring of $E$ over $\overline{\mathbb{F}}_q$. [2] The elements of $\overline{\mathbb{F}}_q[E]$ are called *regular functions*.

- The *function field* $\mathbb{F}_q(E)$ of $E$ over $\mathbb{F}_q$ is the field of fractions of $\mathbb{F}_q[E]$. Similarly, the function field $\overline{\mathbb{F}}_q(E)$ of $E$ over $\overline{\mathbb{F}}_q$ is the field of fractions of $\overline{\mathbb{F}}_q[E]$. The elements of $\overline{\mathbb{F}}_q(E)$ are called *rational functions*.

Note that for each $l \in \overline{\mathbb{F}}_q[E]$, we can repeatedly replace occurrences of $y^2$ by $y^2 - F(x, y)$ to obtain a representation $l(x, y) = v(x) + yw(x)$ where $v(x), w(w) \in \overline{\mathbb{F}}_q[x]$.

**Definition** Let $l \in \overline{\mathbb{F}}_q[E]$ be a regular function, $l(x, y) = v(x) + yw(x)$. We define the *degree* of $l$ by $\deg(l) = \max(2\deg_x(v), 3 + 2\deg_x(w))$.

Let $f \in \overline{\mathbb{F}}_q(E)$ be a rational function and $P = (x_0, y_0) \in E$ be a finite point. Then $f$ is said to be *defined* or *regular* at $P$ if there exists a representation $f = g/h$, $g, h \in \mathbb{F}_q[E]$ such that $h(x_0, y_0) \neq 0$ ($h(x_0, y_0)$ is evaluated like a polynomial). From now on, if $P = (x_0, y_0)$, we denote $h(x_0, y_0)$ simply by $h(P)$. If $f$ is regular at $P$,

[2] Using the fact that neither partial derivative of $F$ vanishes at any point of $E$, we can prove that $F$ is an irreducible polynomial in $\overline{\mathbb{F}}_q[x, y]$, which is why $\mathbb{F}_q[E]$ and $\overline{\mathbb{F}}_q[E]$ are integral domains.

we put $f(P) = g(P)/h(P)$. If $f(P) = 0$, then $f$ is said to have a *zero* at $P$. If $f$ is not defined at $P$, then $f$ is said to have a *pole* at $P$, and we write $f(P) = \infty$. We define the value of $f$ at the point $\mathcal{O}_E$ as follows. Let $f = g/h$ with $g, h \in \overline{\mathbb{F}}_q[E]$. If $\deg(g) < \deg(h)$, then $f(\mathcal{O}_E) = 0$. If $\deg(g) > \deg(h)$, then $f(\mathcal{O}_E) = \infty$. If $\deg(g) = \deg(h)$, then $f(\mathcal{O}_E) = a/b$, where $a$ is the highest degree term in $g$ and $b$ is the highest degree term in $h$.

For each point $P \in E$, there exists a rational function $u \in \overline{\mathbb{F}}_q(E)$, $u(P) = 0$, such that for any $f \in \overline{\mathbb{F}}_q(E)$, we can write $f = u^d s$ where $s \in \overline{\mathbb{F}}_q(E)$, $d \in \mathbb{Z}$, $s(P) \neq 0, \infty$. Such a $u$ is called a *uniformizing parameter* for $P$.

Let $f \in \overline{\mathbb{F}}_q(E)$ and $P \in E$. Write $f = u^d s$ where $u$ is a uniformizing parameter at $P$ and $s(P) \neq 0, \infty$. The *order of $f$ at $P$* is defined to be $d$ and we write $\mathrm{ord}_P(f) = d$. The point $P$ is a zero of $f$ if and only if $\mathrm{ord}_P(f) > 0$ and we define the *multiplicity* of that zero to be $\mathrm{ord}_P(f)$. The point $P$ is a pole of $f$ if and only if $\mathrm{ord}_P(f) < 0$ and we define the *multiplicity* of that pole to be $-\mathrm{ord}_P(f)$. $\mathrm{ord}_P(f) = 0$ if and only if $f$ is defined at $P$.

**Theorem 6.1.8** *Let $f \in \overline{\mathbb{F}}_q(E)$. Then $f$ has finitely many zeros and poles on $E$. Furthermore,*

$$\sum_{P \in E} \mathrm{ord}_P(f) = 0.$$

### 6.1.3 Divisors

**Definition** Let $E$ be an elliptic curve defined over $\mathbb{F}_q$.

- The group of divisors $\mathrm{Div}(E)$ is the free abelian group generated by the points

of $E$.

$$\mathrm{Div}(E) = \left\{ \sum_{P \in E} m_p \langle P \rangle : m_P = 0 \text{ for all but finitely many } P \in E \right\}.^{3}$$

- The *degree* of a divisor $D = \sum_{P \in E} m_P \langle P \rangle$ is defined to be $\deg(D) = \sum_{P \in E} m_P$.
- The *support* of a divisor $D = \sum_{P \in E} m_P \langle P \rangle$ is defined to be $\mathrm{Supp}(D) = \{P \in E \mid m_P \neq 0\}$.
- The subgroup of $\mathrm{Div}(E)$ of divisors of degree zero is denoted by $\mathrm{Div}^0(E)$.
- Let $r \in \overline{\mathbb{F}}_q^*$ be a rational function. We define $\mathrm{div}(r) = \sum_{P \in E} \mathrm{ord}_P(r) \langle P \rangle$. This is a divisor since every rational function has finitely many zeros and poles. A divisor $D$ is called *principal* if $D = \mathrm{div}(r)$ for some $r \in \overline{\mathbb{F}}_q^*$.
- The subgroup of $\mathrm{Div}(E)$ of principal divisors is denoted by $\mathrm{Prin}(E)$. [4]
- Two divisors $D_1$ and $D_2$ are called *equivalent* if $D_1 - D_2 \in Prin(E)$. We write $D_1 \sim D_2$.

From theorem 6.1.8, we get that $\mathrm{Prin}(E) \subset \mathrm{Div}^0(E)$. The following result is very useful to identify principal divisors.

---

[3] These sums should be seen as purely formal sums and are not to be confused with the addition on the elliptic curve.

[4] $\mathrm{Prin}(E)$ is a subgroup of $\mathrm{Div}(E)$ since, for all $r_1, r_2 \in K(E)$, $\mathrm{div}(r_1) + \mathrm{div}(r_2) = \mathrm{div}(r_1 r_2)$, $-\mathrm{div}(r_1) = \mathrm{div}(1/r_1)$ and $0 = \mathrm{div}(1)$.

**Theorem 6.1.9** *Let $D = \sum_{P \in E} m_P \langle P \rangle$ be a divisor. Then $D$ is principal if and only if $\sum_{P \in E} m_P = 0$ and $\sum_{P \in E} m_P P = \mathcal{O}_E$.*

From this result, it is easy to show that for every divisor $D \in \mathrm{Div}^0(E)$, there is a unique point $Q \in E$ such that $D \sim \langle Q \rangle - \langle \mathcal{O}_E \rangle$ (if $D = \sum_{P \in E} m_P \langle P \rangle$, then $Q = \sum_{P \in E} m_P P$). Using this, we see that the group of points of $E$ is isomorphic to $\mathrm{Div}^0(E)/\mathrm{Prin}(E)$.

Let $D = \sum_{P \in E} m_P \langle P \rangle$ be a divisor and let $f \in \overline{\mathbb{F}}_q(E)^*$ be a rational function such that $\mathrm{Supp}(D) \cap \mathrm{Supp}(div(f)) = \emptyset$. Then we define the value of $f$ at $D$ to be

$$f(D) = \prod_{P \in \mathrm{Supp}(D)} f(P)^{m_P}.$$

## 6.2 Bilinear Pairing on Elliptic curves

### 6.2.1 The Weil Pairing

Let $E$ be an elliptic curve defined over $\mathbb{F}_q$. Let $m$ be a positive integer coprime to $q$ and let $\mu_m \subset \overline{\mathbb{F}}_q$ be the group of $m^{th}$ roots of unity ($\mu_m \subset \mathbb{F}_{q^k}$ provided that $m \mid (q^k - 1)$).

Let $P, Q \in E[m]$. Let $D_1, D_2 \in \mathrm{Div}^0(E)$ be such that $D_1 \sim \langle P \rangle - \langle \mathcal{O}_E \rangle$, $D_2 \sim \langle Q \rangle - \langle \mathcal{O}_E \rangle$ and $\mathrm{Supp}(D_1) \cap \mathrm{Supp}(D_2) = \emptyset$. Then, by Theorem 6.1.9, $mD_1$ and $mD_2$ are principal divisors. Let $f_{D_1}, f_{D_2} \in \overline{\mathbb{F}}_q(E)$ be such that $div(f_{D_1}) = mD_1$ and $div(f_{D_2}) = mD_2$.

The *Weil pairing* is the function $e_m : E[m] \times E[m] \to \mu_m$ defined by $e_m(P, Q) = f_{D_1}(D_2)/f_{D_2}(D_1)$.

The Weil pairing satisfies the following properties:

1. *Well-Defined*: The value of $e_m(P,Q)$ is independent of the choice of $D_1, D_2, f_{D_1}, f_{D_2}$.

2. *Identity*: For all $P \in E[m]$, $e_m(P,P) = 1$.

3. *Non-Degeneracy*: For $P \in E[m]$, $e_m(P,Q) = 1$ for all $Q \in E[m]$ if and only if $P = \mathcal{O}_E$.

4. *Bilinearity*: For all $P, Q, R \in E[m]$, $e_m(P+Q,R) = e_m(P,R)e_m(Q,R)$ and $e_m(P,Q+R) = e_m(P,Q)e_m(P,R)$.

5. *Alternation*: For all $P, Q \in E[m]$, $e_m(P,Q) = e_m(Q,P)^{-1}$.

6. If $E[m] \subset E(\mathbb{F}_{q^n})$, then $e_m(P,Q) \in \mathbb{F}_{q^n}$ for all $P, Q \in E[m]$ (i.e. $\mu_m \subset \mathbb{F}_{q^n}$).

### 6.2.2 The Tate Pairing

Let $E$ be an elliptic curve defined over $\mathbb{F}_q$. Let $m$ be a positive integer coprime to $q$ and let $k$ be a positive integer such that $m \mid (q^k - 1)$.

Let $P \in E[m]$ and $Q \in E$. Let $D_1, D_2 \in \mathrm{Div}^0(E)$ be such that $D_1 \sim \langle P \rangle - \langle \mathcal{O}_E \rangle$, $D_2 \sim \langle Q \rangle - \langle \mathcal{O}_E \rangle$ and $\mathrm{Supp}(D_1) \cap \mathrm{Supp}(D_2) = \emptyset$. Then, by Theorem 6.1.9, $mD_1$ is a principal divisor. Let $f_{D_1} \in \overline{\mathbb{F}}_q(E)$ be such that $div(f_{D_1}) = mD_1$.

The *Tate pairing* is the function $\langle \cdot, \cdot \rangle : E(\mathbb{F}_{q^k})[m] \times E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k}) \rightarrow \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^m$ defined by $\langle P, Q \rangle = f_{D_1}(D_2)$. The quotient group $E/mE$ can be thought of as the set of equivalence classes of $E$ under the equivalence relation $P \equiv Q$ if and only if there exists $R \in E$ such that $P = Q + mR$. A similar interpretation

can be used for $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^m$. We note that if $m^2 \| \#E(\mathbb{F}_{q^k})$, then $E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$ is isomorphic to $E[m]$.

The Tate pairing satisfies the following properties:

1. *Well-Defined*: $\langle \mathcal{O}_E, Q \rangle \in (\mathbb{F}_{q^k}^*)^m$ for all $Q \in E(\mathbb{F}_{q^k})$ and, for $P \in E(\mathbb{F}_{q^k})[m]$, $\langle P, Q \rangle \in (\mathbb{F}_{q^k}^*)^m$ for all $Q \in mE$. Further, the value of $\langle P, Q \rangle$ is independent of the choice of $D_1, D_2, f_{D_1}$.

2. *Non-Degeneracy*: For $P \in E[m]$, $\langle P, Q \rangle = 1$ for all $Q \in E$ if and only if $P = \mathcal{O}_E$.

3. *Bilinearity*: For all $P, Q, R \in E[m]$, $\langle P + Q, R \rangle = \langle P, R \rangle \langle Q, R \rangle$ and $\langle P, Q + R \rangle = \langle P, Q \rangle \langle P, R \rangle$.

Note that $\langle P, Q \rangle^m = \langle mP, Q \rangle = \langle \mathcal{O}_e, Q \rangle \in (F_{q^k}^*)^m$, therefore, $\langle P, Q \rangle$ is an $m^{th}$ root of unity, up to multiplication by an $m^{th}$ power.

It is interesting to note that when $m^2 \| \#E(\mathbb{F}_{q^k})$, the Weil pairing is related to the Tate pairing by the equation

$$e_m(P, Q) = \langle P, Q \rangle / \langle Q, P \rangle \text{ for } P, Q \in E[m] \text{ if } P \text{ and } Q \text{ are independent.}$$

### 6.2.3 Computing the Weil and Tate Pairings

Let $E$ be an elliptic curve defined over $\mathbb{F}_q$.

The following result will be used repeatedly to compute the Weil and Tate pairings. We define $g_{U,V} : E \to \overline{\mathbb{F}}_q$ to be the line through the points $U$ and $V$ of $E$[5]. If $U = V$ then $g_{U,V}$ is the tangent to E at $U$, and if either one of $U$ or $V$ is the

[5] If the line through $U$ and $V$ has equation $ax + by + c = 0$, then $g_{U,V} = ax + by + c$.

point at infinity $\mathcal{O}_E$, $g_{U,V}$ is the vertical line through the other point. We write $g_U$ as a shorthand for $g_{U,-U}$.

**Theorem 6.2.1** *Let $P$ be a point on $E$ and let $f_{P,c} \in \overline{\mathbb{F}}_q$ be such that $div(f_{P,c}) = c\langle P\rangle - \langle cP\rangle - (c-1)\langle \mathcal{O}_E\rangle$ [6] for a positive integer $c$. Then, for all $a, b \in \mathbb{Z}$,*

$$div(f_{P,a+b}) = div(f_{P,a}) + div(f_{P,b}) + div(g_{aP,bP}) - div(g_{(a+b)P})$$

*and therefore, $f_{P,a+b}(Q) = f_{P,a}(Q) \cdot f_{P,b}(Q) \cdot g_{aP,bP}(Q)/g_{(a+b)P}(Q)$ provided that $Q \notin Supp(div(f_{P,a})) \cup Supp(div(f_{P,b}))$ and $g_{aP,bP}(Q), g_{(a+b)P}(Q) \in \overline{\mathbb{F}}_q^*$.*

Note that $div(f_{P,0}) = div(f_{P,1}) = 0$, so $f_{P,0}(Q) = f_{P,1}(Q) = 1$. Also, from the theorem above, $f_{P,a+1}(Q) = f_{P,a}(Q) \cdot g_{aP,P}(Q)/g_{(a+1)P}(Q)$ (because $f_{P,1}(Q) = 1$) and $f_{P,2a}(Q) = f_{P,a}(Q)^2 \cdot g_{aP,aP}(Q)/g_{2aP}(Q)$.

Using these formulas, together with the standard double-and-add strategy, we get the following algorithm, due to Miller [42], to compute $f_{P,c}(Q)$ for any positive integer $c$.

Let $(b_t \cdots b_1 b_0)_2$ be the binary representation of $c$.

Set $f = 1$ and $V = P$.

For $i = t - 1$ down to 0,

Set $f = f^2 \cdot g_{V,V}(Q)/g_{2V}(Q)$ and $V = 2V$.

If $b_i = 1$, then set $f = f \cdot g_{V,P}(Q)/g_{V+P}(Q)$ and $V = V + P$

Return $f$.

[6] From Theorem 6.1.9, this is a principal divisor.

**Computing the Weil Pairing**

Let $E$ be an elliptic curve defined over $\mathbb{F}_q$, let $m$ be a positive integer with $\gcd(m,q) = 1$ and let $P, Q \in E[m]$. We show how to compute $e_m(P,Q)$ for $P, Q \neq \mathcal{O}_E$ (if $P$ or $Q$ is $\mathcal{O}_E$, $e_m(P,Q) = 1$).

Pick points $T, U \in E \setminus \{\mathcal{O}_E\}$ at random such that $P + T \neq U, U + Q$ and $T \neq U, U + Q$. Let $D_1 = \langle P + T\rangle - \langle T\rangle$ and $D_2 = \langle Q + U\rangle - \langle U\rangle$. Then $D_1 \sim \langle P\rangle - \langle \mathcal{O}_E\rangle$ and $D_2 \sim \langle Q\rangle - \langle \mathcal{O}_E\rangle$. We want to find $f_{D_1}, f_{D_2} \in \overline{\mathbb{F}}_q$ such that $div(f_{D_1}) = mD_1$ and $div(f_{D_2}) = mD_2$ and compute $f_{D_1}(D_2)/f_{D_2}(D_1)$. Note that $m\langle P + T\rangle - m\langle T\rangle = div(f_{P+T,m}) - div(f_{T,m})$. Thus,

$$e_m(P,Q) = f_{D_1}(D_2)/f_{D_2}(D_1) = \frac{f_{P+T,m}(D_2) f_{U,m}(D_1)}{f_{T,m}(D_2) f_{Q+U,m}(D_1)}.$$

Use Miller's algorithm to compute $f_{P+T,m}(Q+U)$, $f_{T,m}(Q+U)$, $f_{P+T,m}(U)$, $f_{T,m}(U)$, $f_{Q+U,m}(P+T)$, $f_{Q+U,m}(T)$, $f_{U,m}(P+T)$ and $f_{U,m}(T)$. Then,

$$\begin{aligned}\frac{f_{P+T,m}(Q+U), f_{T,m}(U) f_{Q+U,m}(T) f_{U,m}(P+T)}{f_{T,m}(Q+U) f_{P+T,m}(U) f_{Q+U,m}(P+T) f_{U,m}(T)} &= \frac{f_{P+T,m}(D_2) f_{U,m}(D_1)}{f_{T,m}(D_2) f_{Q+U,m}(D_1)} \\ &= e_m(P,Q).\end{aligned}$$

Note that $f_{P+T,m}(Q+U)$ and $f_{P+T,m}(U)$ can be computed in the same invocation of Miller's algorithm, similarly for $f_{T,m}(Q+U)$ and $f_{T,m}(U)$, $f_{Q+U,m}(P+T)$ and $f_{Q+U,m}(T)$, $f_{U,m}(P+T)$ and $f_{U,m}(T)$. So we need 4 invocations of Miller's algorithm to compute $e_m(P,Q)$.

**Computing the Tate Pairing**

The computation of the Tate pairing is much simpler. Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ and let $m$ be a positive integer coprime to $q$. Let $k$ be a positive integer such that $m \mid (q^k - 1)$ and let $P \in E(\mathbb{F}_{q^k})[m]$ and $Q \in E(\mathbb{F}_{q^k})$. We show how to compute $\langle P, Q \rangle$ for $P, Q \neq \mathcal{O}_E$ (if $P$ or $Q$ is $\mathcal{O}_E$, $\langle P, Q \rangle = 1$).

Pick a point $U \in E(\mathbb{F}_{q^k}) \setminus \{\mathcal{O}_E\}$ such that $P \neq U, Q + U$ and $U \neq -Q$. Let $D_1 = \langle P \rangle - \langle \mathcal{O}_E \rangle$ and $D_2 = \langle Q + U \rangle - \langle U \rangle \sim \langle Q \rangle - \langle \mathcal{O}_E \rangle$. Note that since $mP = \mathcal{O}_E$, $f_{P,m} = m\langle P \rangle - \langle mP \rangle - (m-1)\langle \mathcal{O}_E \rangle = m\langle P \rangle - m\langle \mathcal{O}_E \rangle = mD_1$. So $\langle P, Q \rangle = f_{P,m}(D_2) = \frac{f_{P,m}(Q+U)}{f_{P,m}(U)}$. This can be computed with only one invocation of Miller's algorithm.

Since the Tate pairing is defined only up to a multiple of an $m^{th}$ power and since most applications in cryptography require a unique value, it is necessary to exponentiate the value of the Tate pairing to the power $(q^k - 1)/m$ to eliminate all $m^{th}$ powers. We write $t_m(P, Q) = \langle P, Q \rangle^{(q^k-1)/m}$.

### 6.2.4 Why Choose Supersingular Curves ?

In cryptography, we generally choose elliptic curves whose group of points is cyclic or close to cyclic, and such that $\#E(\mathbb{F}_q)$ is divisible by a large prime, and then use the subgroup of $E(\mathbb{F}_q)$ of prime order to implement cryptographic schemes.

Let $E$ be such an elliptic curve defined over $\mathbb{F}_q$ and let $\mathbb{G} \subset E(\mathbb{F}_q)$ be a cyclic subgroup of prime order $m$. The computation of the Weil pairing $e_m$ and the Tate pairing $t_m$ take place in an extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ with $k$ such that $m \mid (q^k - 1)$.

**Definition** The smallest $k$ such that $m \mid (q^k - 1)$ is called the *security multiplier*

or *embedding degree* of the subgroup of order $m$.

Note that the security multiplier of $\mathbb{G}$ is equal to the order of $q$ modulo $m$. Therefore, in general, we expect $k \approx m$ which means that computing the Weil or Tate pairing is completely impractical since the representation of elements of $\mathbb{F}_{q^k}$ with such a large $k$ have size exponential in $\log q$. [7]

However, we can show that supersingular elliptic curves have surprisingly small security multipliers. First, we separate supersingular elliptic curves into 6 classes, depending on the value $t = q + 1 - \#E(\mathbb{F}_q)$ and their group structure:

1. $t = 0$ and $E(\mathbb{F}_q)$ is cyclic.
2. $t = 0$ and $E(\mathbb{F}_q) \cong \mathbb{Z}_{(q+1)/2} \oplus \mathbb{Z}_2$ (and $q \equiv 3 \pmod 4$).
3. $t^2 = q$ (and $q$ is a square).
4. $t^2 = 2q$ (and $\mathbb{F}_q$ has characteristic 2 and $q$ is not a square).
5. $t^2 = 3q$ (and $\mathbb{F}_q$ has characteristic 3 and $q$ is not a square).
6. $t^2 = 4q$ (and $q$ is a square).

We know that $E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$ with $n_2 \mid n_1$. Using Theorems 6.1.4 and 6.1.6, we can find the smallest integer $k$ such that $E[n_1] \subset E(\mathbb{F}_{q^k})$. [8] These integers are listed in Table 6.1.

[7] By Theorem 6.1.1, $\#E(\mathbb{F}_q) \approx q$ and by choice, $m \approx \#E(\mathbb{F}_q)$, so $k \approx q$.

[8] By property 6 of the Weil pairing, this implies that $n_1 \mid (q^k - 1)$.

| Class of curve | $t$ | Group structure | $n_1$ | k |
|---|---|---|---|---|
| 1. | 0 | cyclic | $q+1$ | 2 |
| 2. | 0 | $\mathbb{Z}_{(q+1)/2} \oplus \mathbb{Z}_2$ | $(q+1)/2$ | 2 |
| 3. | $\pm\sqrt{q}$ | cyclic | $q+1 \mp \sqrt{(q)}$ | 3 |
| 4. | $\pm\sqrt{2q}$ | cyclic | $q+1 \mp \sqrt{(2q)}$ | 4 |
| 5. | $\pm\sqrt{3q}$ | cyclic | $q+1 \mp \sqrt{(3q)}$ | 6 |
| 6. | $\pm 2\sqrt{q}$ | $\mathbb{Z}_{\sqrt{q}\mp 1} \oplus \mathbb{Z}_{\sqrt{q}\mp 1}$ | $\sqrt{q} \mp 1$ | 1 |

Table 6.1: Maximum Security Multipliers for Supersingular Elliptic Curves.

We usually choose $\mathbb{G}$ to be a subgroup of prime order with $|\mathbb{G}| \approx n_1$, so it is very likely that the security multiplier of $\mathbb{G}$ is equal to those values of $k$, and not less.

As noted in Chapter 2, the existence of a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{F}^*_{q^k}$ implies that the discrete logarithm problem in $\mathbb{G}$ is no harder than the discrete logarithm problem in $\mathbb{F}^*_{q^k}$. Therefore, we need to have $q^k \approx 2^{1000}$ to make sure that the discrete logarithm problem is intractable in $\mathbb{F}^*_{q^k}$. However, it is desirable to have $q$ as small as possible to speed up the computations in $\mathbb{G}$ (and to get small signatures with the scheme described in Chapter 4). So we do not want $k$ to be too small either. Therefore, the curves of type 5 seem to be the 'best' elliptic curves for practical applications.

### 6.2.5 Modifying the Weil and Tate Pairings to Obtain Bilinear Maps

The Weil and Tate pairings as we presented them are not suitable bilinear maps: property 2 of the Weil pairing implies that $e_m(P, Q) = 1$ whenever $P$ and $Q$ are in

the same cyclic subgroup of $E[m]$. As for the Tate pairing, the value of $t_m(P,Q)$ when $P$ and $Q$ are dependent may or may not be 1, depending on the curve. We show a way to modify the Weil and Tate pairings to get bilinear maps.

It can be shown that if $P, Q \in E[m]$ are independent points of order $m$, then $e_m(P,Q)$ is a primitive $m^{th}$ root of unity in $\mathbb{F}_{q^k}$. Since $e_m(P,Q) = \langle P,Q\rangle/\langle Q,P\rangle$ (when $m^2 \| \#E(\mathbb{F}_{q^k})$), it follows that $t_m(P,Q)$ must be a primitive $m^{th}$ root of unity as well. If we could find an efficiently computable group automorphism $\Phi : E[m] \rightarrow E[m]$ such that for some $P \in E[m]$ of order $m$, $P$ and $\Phi(P)$ are independent, then, if we let $\mathbb{G}$ be the group of points generated by $P$ and $\mu_m$ be the group of $m^{th}$ roots of unity in $\mathbb{F}_{q^k}^*$, the functions $\tilde{e}_m : \mathbb{G} \times \mathbb{G} \rightarrow \mu_m$ and $\tilde{t}_m : \mathbb{G} \times \mathbb{G} \rightarrow \mu_m$ defined by $\tilde{e}_m(P,Q) = e_m(P, \Phi(Q))$ and $\tilde{t}_m(P,Q) = t_m(P,\Phi(Q))$ respectively would be bilinear maps.

**Definition** Let $E_1$ and $E_2$ be elliptic curves defined over $\mathbb{F}_q$.

- A *morphism* or *rational map* from $E_1$ to $E_2$ is a map of the form $\Phi : E_1 \rightarrow E_2$, $\Phi = [g,h]$ where $g, h \in \overline{\mathbb{F}}_q(E_1)$ such that

  1. for all $P \in E_1$, if $g$ and $h$ are both defined at $p$, then $\Phi(P) = (g(P), h(P)) \in E_2$.
  2. if $g$ or $h$ is not defined at $P$, then we define $\Phi(P) = \mathcal{O}_{E_2}$.

  If $g, h \in \mathbb{F}_{q^k}(E_1)$ for some $k$, then $\Phi$ is said to be *defined over* $\mathbb{F}_{q^k}$.

- An *isogeny* is a morphism $\Phi : E_1 \rightarrow E_2$ satisfying $\Phi(\mathcal{O}_{E_1}) = \mathcal{O}_{E_2}$.

- An *endomorphism* on $E_1$ is an isogeny $\Phi : E_1 \rightarrow E_1$.

- An isogeny $\Phi : E_1 \to E_2$ is said to be an *isomorphism* if there exists an isogeny $\Psi : E_2 \to E_1$ such that $\Psi \circ \Phi$ and $\Phi \circ \Psi$ are identity maps on $E_1$ and $E_2$ respectively.

- An endomorphism on $E_1$ is called an *automorphism* of $E_1$ if it is also an isomorphism.

**Theorem 6.2.2** *Let $E_1, E_2$ be elliptic curves defined over $\mathbb{F}_q$. If $\Phi : E_1 \to E_2$ is an isogeny, then for all $P, Q \in E_1$, $\Phi(P + Q) = \Phi(P) + \Phi(Q)$ (so $\Phi$ is a group homomorphism). Furthermore, if $\Phi$ is an isomorphism, then it induces a group isomorphism between the group of points of $E_1$ and $E_2$.*

For a supersingular curves $E$ defined over $\mathbb{F}_q$ with security multiplier greater than 1, it is usually easy to find an automorphism $\Phi$ such that, for most points $P \in E(\mathbb{F}_q)$, $\Phi(P) \notin E(\mathbb{F}_q)$. Therefore, $P$ and $\Phi(P)$ are usually independent and have the same order since $\Phi$ is an isomorphism. Table 6.2 lists a few cryptographically interesting supersingular elliptic curves with their curve types and Table 6.3 give a suitable automorphism for each of those curves.

The curve $E_{4,1}$ over $\mathbb{F}_{3^{97}}$ has a cyclic subgroup of prime order $p$, where $p$ is a 151 bit prime, large enough to make the discrete logarithm problem intractable in the subgroup, and the discrete logarithm problem is also intractable in the field $\mathbb{F}_{3^{6 \cdot 97}}$, so this curve would be a perfect choice.

We now show that the modified Weil pairing is a bilinear map. Let $E$ be one of the elliptic curves defined over $\mathbb{F}_q$ described in table 6.2. Let $P \in E(\mathbb{F}_q)$ be a point of prime order $p$ and let $\mathbb{G}$ be the cyclic group generated by $P$. Let $\mu_p \subset \overline{\mathbb{F}}_q$

| Curve equation | Underlying field | Order | Type |
|---|---|---|---|
| $E_1 : y^2 = x^3 + 1$ | $\mathbb{F}_p$, $p$ prime<br>$p \equiv 2 \pmod 3$ | $p+1$ | 1 or 2 |
| $E_2 : y^2 = x^3 + x$ | $\mathbb{F}_p$, $p$ prime<br>$p > 3$, $p \equiv 3 \pmod 4$ | $p+1$ | 1 or 2 |
| $E_{3,b} : y^2 + y = x^3 + x + b$,<br>$b \in \{0, 1\}$ | $\mathbb{F}_{2^l}$, $l$ prime | $2^l \pm 2^{\frac{l+1}{2}} + 1$ | 4 |
| $E_{4,b} : y^2 = x^3 - x + b$,<br>$b \in \{-1, 1\}$ | $\mathbb{F}_{3^l}$, $l$ prime | $3^l \pm 3^{\frac{l+1}{2}} + 1$ | 5 |

Table 6.2: Cryptographically Interesting Elliptic Curves.

| Curve | Automorphism | Conditions |
|---|---|---|
| $E_1$ | $\Phi(x, y) = (\zeta x, y)$ | $1 \neq \zeta \in \mathbb{F}_{p^2}$<br>$\zeta^2 + \zeta + 1 = 0$ |
| $E_2$ | $\Phi(x, y) = (-x, iy)$ | $i \in \mathbb{F}_{p^2}$, $i^2 = -1$ |
| $E_{3,b}$ | $\Phi(x, y) = (x + s^2, y + sx + t)$ | $s, t \in F_{2^{4l}}$<br>$s^4 + s = 0$<br>$t^2 + t + s^6 + s^2 = 0$ |
| $E_{4,b}$ | $\Phi(x, y) = (-x + r_b, iy)$ | $r_b, i \in \mathbb{F}_{3^{6l}}$<br>$r_b^3 - r_b - b = 0$, $i^2 = -1$ |

Table 6.3: Automorphisms of Elliptic Curves.

be the group of $p^{th}$ roots of unity ($\mu_p \subset \mathbb{F}_{q^k}$ for some $k \leq 6$). Then, the function $\tilde{e}_p : \mathbb{G} \times \mathbb{G} \to \mu_p$ defined by $\tilde{e}_p(Q, R) = e_p(Q, \Phi(R))$ is a bilinear map since it is:

**Bilinear:** For all $Q, R, S \in \mathbb{G}$, $\tilde{e}_p(Q, R+S) = e_p(Q, \Phi(R+S)) = e_p(Q, \Phi(R) + \Phi(S)) = e_p(Q, \Phi(R))e_p(Q, \Phi(S)) = \tilde{e}_p(Q, R)\tilde{e}_p(Q, S)$ and $\tilde{e}_p(Q+R, S) = e_p(Q+R, \Phi(S)) = e_p(Q, \Phi(S))e_p(R, \Phi(S)) = \tilde{e}_p(Q, S)\tilde{e}_p(R, S)$.

**Non-Degenerate:** If $Q$ is a generator for $\mathbb{G}$, then $\tilde{e}_p(Q, Q) = e_p(Q, \Phi(Q)) \neq 1$ since $Q$ and $\Phi(Q)$ are independent.

**Computable:** Because $\Phi$ and $e_p$ are efficiently computable.

### 6.2.6 Other Abelian Varieties Curves

Elliptic curves are abelian varieties of dimension 1. Since it is possible to implement cryptographic schemes in the jacobian of higher dimension abelian varieties, and since the Weil and Tate pairings can also be implemented in those curves (and have similar properties), it is natural to ask if some of those curves might be better for the implementation of bilinear maps. We know that the maximal security multiplier for supersingular elliptic curves is 6. One might hope to do better by using other supersingular abelian varieties.

First, note that since an abelian variety $A$ of dimension $g$ defined over $\mathbb{F}_q$ has a jacobian of size approximately $q^g$, the appropriate security parameter is $\frac{k}{g}$ [9], where $k$ is the smallest positive integer such that the Tate pairing can map $Jac_{\mathbb{F}_q}(A)$ into $\mathbb{F}_{q^k}$. We will call this $k$ the *embedding degree* of $A(\mathbb{F}_q)$.

[9] Since we want the security parameter to be the ratio between the length of the representation of an element of $\mathbb{G}_2$ and the length of the representation of an element of $\mathbb{G}_1$.

Galbraith [22] and Rubin and Silverberg [46] studied the problem of finding an upper bound on the embedding degree of supersingular abelian varieties. Precise upper bounds on the embedding degree of supersingular abelian varieties were found in [46] and are given in Table 6.4 for all possible defining fields $\mathbb{F}_q$ ($q = p^m$, $p$ prime) and for dimensions 1 through 6 (it is not useful to find the bounds for higher dimensions since experimental results (Gaudry [24]) suggest these curves are unlikely to be used for cryptography). The symbol * indicates that there is no simple supersingular abelian variety of dimension $g$ over $\mathbb{F}_q$

| g | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $q$ a square | 3 | 6 | 9 | 15 | 11 | 21 |
| $q$ not a square, $p > 11$ | 2 | 6 | * | 12 | * | 18 |
| $q$ not a square, $p = 2$ | 4 | 12 | * | 20 | * | 36 |
| $q$ not a square, $p = 3$ | 6 | 4 | 18 | 30 | * | 42 |
| $q$ not a square, $p = 5$ | 2 | 6 | * | 15 | * | 18 |
| $q$ not a square, $p = 7$ | 2 | 6 | 14 | 12 | * | 42 |
| $q$ not a square, $p = 11$ | 2 | 6 | * | 12 | 22 | 18 |

Table 6.4: Upper Bound of the Embedding Degree for Supersingular Curves.

Note that the maximal security parameter is $7.5 = 30/4$ and is attained only over fields of characteristic 3. Since this is not much better than what we can attain with supersingular elliptic curves, one might argue that this small increase in the security parameter is not worth the added complexity of dealing with curves of higher dimension. However, this knowledge of higher dimension curves can be used to obtain a shorter representation of points for some elliptic curves, which effectively increases the security parameter to 7.5 (see [46] for full details).

### 6.2.7 Non-Supersingular Elliptic Curves with Low Security Multiplier

We have seen that in general, for non-singular elliptic curves, the security multiplier is too large to allow efficient computations. Recent publications ([3, 15]) explore the possibility that elliptic curves with subgroups with low security multiplier may be found efficiently. [15] gives an example of a non-supersingular elliptic curve that has a cyclic subgroup with security multiplier 11, and [3] found one with security multiplier 12. Both articles give methods to find such elliptic curves relatively efficiently.

One might hope to be able to modify the Weil or Tate pairings similarly as in Section 6.2.5 to obtain a bilinear map. Unfortunately, the following result denies that possibility.

**Theorem 6.2.3** *Let $E$ be a non-supersingular elliptic curve defined over $\mathbb{F}_q$. Let $\Phi$ be an endomorphism on $E$. Then, $\Phi$ is defined over $\mathbb{F}_q$.*

So the image under $\Phi$ of a point defined over $\mathbb{F}_q$ will be a point defined over $\mathbb{F}_q$. As a consequence, if $P \in E(\mathbb{F}_q)$ is a point of order $m$ for some positive integer $m$ and $\Phi$ is an automorphism of $E$, then either $\Phi(P)$ is in the group generated by $P$, or $E[m] \subset E(\mathbb{F}_q)$, in which case the security multiplier of the group of points generated by $P$ is 1. Additional evidence that there does not exist an efficiently computable group homomorphism to implement the strategy of Section 6.2.5 can be found in [51].

However, it is usually possible to modify the schemes that use a bilinear map so

that they can use the unmodified Weil or Tate pairings. We present some of those modified schemes in appendix A.

### 6.2.8 Fast Implementation of the Tate Pairing

In this section, we present results from [2] and [23] which greatly speed up the computation of the Tate pairing.

Let $E$ be an elliptic curve defined over $\mathbb{F}_q$. Our first, and perhaps most important observation is that if $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$, then the comptation of $\langle P, Q \rangle$ is much faster than the computation of $\langle Q, P \rangle$. The reason is that when computing $\langle P, Q \rangle$, the calculations to get the coefficients of the lines $g_{U,V}$, as well as all the elliptic curve the point additions, occur in $\mathbb{F}_q$, while they would occur in $\mathbb{F}_{q^k}$ when computing $\langle Q, P \rangle$. (This is why, in Section 6.2.5, we defined $\tilde{t}_m(P, Q) = \langle P, \Phi(Q) \rangle^{(q^k-1)/m}$ and not $\langle \Phi(P), Q \rangle^{(q^k-1)/m}$.)

#### Simplifying the Computation of the Tate Pairing

It is a well-known fact that, in finite fields, divisions are much more expensive than multiplications. Therefore, it is desirable to reduce the number of divisions in Miller's algorithm to a minimum. We first observe that all the divisions in Miller's algorithm can be gathered into a single divisions at the end of the algorithm by representing the value of $f$ by a quotient $f_1/f_2$, gather all the numerators in $f_1$, the denominators in $f_2$ and compute only one division at the end of the algorithm.

The following results can be used to reduce even further the number of divisions (and the number of operations in general) in the computation of the Tate pairing.

Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ whose group of points contains a cyclic subgroup of order $n$. Let $m$ be a divisor of $n$. Let $P \in E(\mathbb{F}_q)[m]$ and $Q \in E[m]$ be linearly independent points (so in particular, $Q \neq \mathcal{O}_E$). Let $f_P \in \overline{\mathbb{F}}_q(E)$ be a function with $\mathrm{div}(f_P) = m\langle P\rangle - m\langle \mathcal{O}_E\rangle$. Let $k$ be the embedding degree of the cyclic subgroup of $E$ of order $m$.

**Theorem 6.2.4** *If $q-1$ is a factor of $(q^k-1)/m$, then $t_m(P,Q) = f_P(Q)^{(q^k-1)/m}$.*

**Proof** Suppose $R \notin \{\mathcal{O}_E, -P\}$ is some point in $E(\mathbb{F}_q)$. Let $f'_P$ be a rational function with $\mathrm{div}(f'_P) = m\langle P+R\rangle - m\langle R\rangle$ Therefore, by definition, $t_m(P,Q) = f'_P(\langle Q\rangle - \langle \mathcal{O}_E\rangle)^{(q^k-1)/m}$ because $\langle P+R\rangle - \langle R\rangle \sim \langle P\rangle - \langle \mathcal{O}_E\rangle$. Since $P$ and $R$ are $\mathbb{F}_q$-rational points, we may assume, without loss of generality, that $f'_P \in \mathbb{F}_q(E)$ [10], and because $f'_P$ does not have a zero or a pole at $\mathcal{O}_E$, we know that $f'_P(\mathcal{O}_E) \in \mathbb{F}_q^*$. By Fermat's Little Theorem for finite fields (lemma 2.3 in [33]), $f'_P(\mathcal{O}_E)^{q-1} = 1$. Since $q-1$ is a factor of $(q^k-1)/m$, we have that $f'_P(\mathcal{O}_E)^{(q^k-1)/m} = 1$ as well. Therefore,

$$\begin{aligned} t_m(P,Q) = f'_P(\langle Q\rangle - \langle \mathcal{O}_E\rangle)^{(q^k-1)/m} &= f'_P(Q)^{(q^k-1)/m} / f'_P(\mathcal{O}_E)^{(q^k-1)/m} \\ &= f'_P(Q)^{(q^k-1)/m}. \end{aligned}$$

Now, consider $P, Q$ to be fixed and $R$ to be variable. Since the statement above holds for all $R \notin \{\mathcal{O}_E, -P\}$, we have that $f'_P(Q)$ is constant when viewed as a function of $R$, and coincides with the value of $f_P(Q)$. Therefore, $t_m(P,Q) =$

[10] We can build such an $f'_P \in \mathbb{F}_q(E)$ using the idea in Theorem 6.2.1, and since the value of $t_m$ is independent of the choice of $f'_P$, as long as $f'_P = mD$ for $D \sim \langle P\rangle - \langle \mathcal{O}_E\rangle$, we may as well choose it in $\mathbb{F}_q(E)$.

$f_P(Q)^{(q^k-1)/m}$. ∎

Note that unless $\#E(\mathbb{F}_q) = q-1$, $q-1$ is usually factor of $(q^k-1)/m$ in cryptographic applications for the following reasons. First, $m$ is generally a prime, so either $m$ divides $q-1$ or $\gcd(m, q-1) = 1$. Second, we know $m$ divides $\#E(\mathbb{F}_q)$, so if $m$ also divides $q-1$, then $m$ must divide $|\#E(\mathbb{F}_q) - (q-1)|$. But, by Theorem 6.1.1, $|\#E(\mathbb{F}_q) - (q-1)| \leq 2\sqrt{q}+2$. So if we choose $m \geq 2\sqrt{q}+2$ (or if $\#E(\mathbb{F}_q)$ is very close to $q-1$), which is generally the case (especially when using a curve with embedding degree 4 or 6), we are assured that $m$ does not divide $q-1$, so $q-1$ will be a factor of $(q^k-1)/m$ (assuming that $m$ is prime).

### Irrelevant Denominators

We note that, in Miller's algorithm, the denominators are always of the form $g_U(Q)$, where $g_U$ is the equation of the vertical line through the point $U$ and $U$ is an $\mathbb{F}_q$-rational point. The following theorem shows that all these denominators can be discarded without affecting the value of $t_m(P,Q)$ when using one of the curves listed in Table 6.5 (the curve equations are listed in Table 6.2).

| Curve | Field | Automorphism | Conditions |
|---|---|---|---|
| $E_2$ | $\mathbb{F}_p$, $p > 3$<br>$p \equiv 3 \pmod 4$ | $\Phi(x,y) = (-x, iy)$ | $i \in \mathbb{F}_{p^2}$, $i^2 = -1$ |
| $E_{3,b}$ | $\mathbb{F}_{2^l}$, $l$ prime | $\Phi(x,y) = (x+s^2, y+sx+t)$ | $s,t \in F_{2^{4l}}$<br>$s^4 + s = 0$<br>$t^2 + t + s^6 + s^2 = 0$ |
| $E_{4,b}$ | $\mathbb{F}_{3^l}$, $l$ prime | $\Phi(x,y) = (-x + r_b, iy)$ | $r_b, i \in \mathbb{F}_{3^{6l}}$<br>$r_b^3 - r_b - b = 0$, $i^2 = -1$ |

Table 6.5: Parameters for Theorem 6.2.5.

**Theorem 6.2.5** *With the settings listed in Table 6.5, the denominators in Miller's algorithm can be discarded altogether without changing the value of* $t_m(P,Q)$.

**Proof** We show that the denominators become unity after the final powering in $t_m$.

- $E_2$: The denominators in Miller's algorithm have the form $g_U(\Phi(Q)) = -x - c$, where $x \in \mathbb{F}_q$ is the abscissa of $Q$ and $c \in \mathbb{F}_q$ is the abscissa of $U$. Hence, $g_U(\Phi(Q))^p = -x^p - c^p = -x - c = g_U(\Phi(Q))$, by the linearity of raising to the power $p$. Thus, $g_U(\Phi(Q))^{p-1} = 1$. Now, the exponent in the final powering is $z = (p^2-1)/m$ where $m$ divides $\#E(\mathbb{F}_p) = p+1$. So, $p-1$ is a factor of $z$. Therefore, $g_U(\Phi(Q))^z = 1$.

- $E_{3,b}$: Let $q = 2^l$. From the defining condition $s^4 = s$, it follows by induction that $s^{4^t} = s$ for all $t \geq 0$, in particular, $s^{q^2} = s^{4^l} = s$. The denominators in Miller's algorithm have the form $g_U(\Phi(Q)) = x + s^2 + c$, where $x \in \mathbb{F}_q$ is the abscissa of $Q$ and $c \in \mathbb{F}_q$ is the abscissa of $U$. Thus, $x^{q^2} = x$ and $c^{q^2} = c$. Therefore, $g_U(\Phi(Q))^{q^2} = x^{q^2} + (s^{q^2})^2 + c^{q^2} = x + s^2 + c = g_U(\Phi(Q))$, by the linearity of raising to the power $q$. Hence, $g_U(\Phi(Q))^{q^2-1} = 1$. Now, the exponent in the final powering is $z = (q^4-1)/m = (q+1+\sqrt{2q})(q+1-\sqrt{2q})(q^2-1)/m$ where $m$ divides $\#E(\mathbb{F}_q) = q+1\pm\sqrt{2q}$. So, $q^2-1$ is a factor of $z$. Therefore, $g_U(\Phi(Q))^z = 1$.

- $E_{4,b}$: Let $q = 3^l$. From the defining condition $r_b^3 - r_b - b = 0$, it follows by induction that $r_b^{3^t} = r_b + b(t \bmod 3)$ for all $t \geq 0$, in particular, $r_b^{q^3} = r_b^{3^{3m}} = r_b$. The denominators in Miller's algorithm have the form $g_U(\Phi(Q)) = r_b - x - c$, where

$x \in \mathbb{F}_q$ is the abscissa of $Q$ and $c \in \mathbb{F}_q$ is the abscissa of $U$. Thus, $x^{q^3} = x$ and $c^{q^3} = c$. Therefore, $g_U(\Phi(Q))^{q^3} = r_b^{q^3} - x^{q^3} - c^{q^3} = r_b - x - c = g_U(\Phi(Q))$, by the linearity of raising to the power $q$. Hence, $g_U(\Phi(Q))^{q^3-1} = 1$. Now, the exponent in the final powering is $z = (q^6 - 1)/m = (q + 1 + \sqrt{3q})(q + 1 - \sqrt{3q})(q^3 - 1)(q + 1)/m$ where $m$ divides $\#E(\mathbb{F}_q) = q + 1 \pm \sqrt{3q}$. So, $q^3 - 1$ is a factor of $z$. Therefore, $g_U(\Phi(Q))^z = 1$. ∎

This strategy does not seem to apply to $E_1$ because, with the automorphism given in Table 6.3, the abscissa of $\Phi(Q)$ is not in $\mathbb{F}_p$.

### Advantages in Fields of Characteristic Three

In elliptic curves defined over fields of characteristic three, the operation of point tripling can be done extremely efficiently. We show how thic fact can be used to improve the efficiency of the point exponentiation operation and of Miller's algorithm.

Let $E$ be an elliptic curve defined over a field of characteristic three with equation of the form $y^2 = x^3 + Ax + B$ [11]. Then, for a point $P\ (x_1, y_1)$ on $E$, the point $3P = (x_3, y_3)$ has coordinates $x_3 = x_1 + y_1^2 + y_1^6$ and $y_3 = -y_1^9$. So the computation of $3P$ does not require any division. This is especially fast since the cubing operation can be done in linear time in fields of characteristic three. In particular, if we use $E_{4,b}$, then $x_3 = x_1^9 - b$ and $y_3 = -y_1^9$.

This leads to a triple-and-add scalar multipliction algorithm that is much faster than the standard double-and-add algorithm. Let the signed ternary representation

[11] Every elliptic curve defined over a field of characteristic greater or equal to 3 is isomorphic to a curve with an equation of this form, see [16] for more details.

of $k$ be $k = (k_t \ldots k_1 k_0)_3$ where $k_i \in \{1, 0, -1\}$ and $k_t \neq 0$. Then we compute $kP$ as follows.

If $k_t = 1$, set $V = P$, if $k_t = -1$, set $V = -P$.

For $i = t - 1$ down to 0

Set $V = 3V$.

If $k_i = 1$ then set $V = V + P$.

If $k_i = -1$ then set $V = V - P$.

Return V.

We can also adapt Miller's algorithm to work in base three. From Theorem 6.2.1, we find that $\mathrm{div}(f_{P,3a}) = 3\mathrm{div}(f_{P,a}) + \mathrm{div}(g_{aP,aP}) + \mathrm{div}(g_{2aP,aP}) - \mathrm{div}(g_{2aP}) - \mathrm{div}(g_{3aP})$. Thus,

$$f_{P,3a}(Q) = \frac{f_{P,a}(Q)^3 \cdot g_{aP,aP}(Q) \cdot g_{2aP,aP}(Q)}{g_{2aP}(Q) \cdot g_{3aP}(Q)}.$$

Note that it is not necessary to compute $2aP$ to obtain the coefficients of $g_{2aP,aP}$ since they can be computed from $aP$ and $3aP$ (because the line $g_{2aP,aP}$ goes through $aP$, $2aP$ and $-3aP$). We also need the fact that if $P = (x_0, y_0)$, then $\mathrm{div}(1/(x - x_0)) = -\langle P \rangle - \langle -P \rangle + 2\langle \mathcal{O}_E \rangle = f_{P,-1}$. Thus, after applying the other optimizations described in this section, we can compute $t_m(P, Q)$ as follows. Let the signed ternary representation of $m$ be $m = (m_t \ldots m_1 m_0)_3$ where $m_i \in \{1, 0, -1\}$ and $m_t = 1$ (because $m > 0$) and let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$.

Set $V = P$ and $f = 1$.

Set $f' = 1/(x_Q - x_P)$.

For $i = t - 1$ down to 0

Set $f = f^3 \cdot g_{V,V}(Q) \cdot g_{2V,V}(Q)/(g_{2V}(Q) \cdot g_{3V}(Q))$ and $V = 3V$.

If $m_i = 1$, then set $f = f \cdot g_{V,P}(Q)/g_{V+P}(Q)$ and $V = V + P$.

If $m_i = -1$, then set $f = f \cdot f' \cdot g_{V,-P}(Q)/g_{V-P}(Q)$ and $V = V - P$.

Return $f^{(q^k-1)/m}$.

Where $k$ is the embedding degree of the cyclic group used. Again, all the denominators can be discarded if we use one of the curves listed in Table 6.5.

We mention that [23] also presents methods to improve the efficienty of arithmetic in fields of characteristic three and in extension fields of characteristic two and three. See [27] for a survey of point exponentiation methods in characteristic two fields.

### Choice of Groups

When working with an elliptic curve with very low embedding degree (say 2), it is necessary to work in finite fields $\mathbb{F}_q$ with $q \approx 2^{512}$. But in these cases, it is not necessary that the prime order $l$ of the subgroup be the same size as $q$, $l$ may be chosen to have only 160 bits. This choice has a huge impact on the complexity of Miller's algorithm, since the number of iterations in Miller's algorithm is a function of $\log l$.

It also makes sense to choose to work in a cyclic group of order $l$, where $l$ has low Hamming weight (or small number of nonzero 'trits' in signed ternary notation if we work in a field of characteristic three) to reduce the number of addition operations in Miller's algorithm. It ofter happens in characteristic two or three that the elliptic curve group order $N$ has low Hamming weight, but the prime order $l$ of the cyclic subgroup used for implementing schemes does not have low Hamming weight. Let

$P, Q$ be independent points of order $l$ and let $f_P$ be a rational function such that $\mathrm{div}(f_P) = l\langle P\rangle - l\langle\mathcal{O}\rangle$. If $N = hl$, $\mathrm{div}({f_P}^h) = N\langle P\rangle - N\langle\mathcal{O}\rangle$. Let $f'_P = {f_P}^h$. Then

$$t_l(P,Q) = f_P(Q)^{(q^k-1)/l} = f_P(Q)^{h(q^k-1)/(hl)} = f'_P(Q)^{(q^k-1)/N} = t_N(P,Q).$$

So one can use $t_N$ instead of $t_l$ without altering the final result.

### Speeding Up the Final Powering

If we use one of the curves in Table 6.2, then it is possible to exploit the periodical structure of the final exponent to reduce the complexity of the final powering. This can be done as follows.

- $E_1, E_2$: Let $\mathbb{F}_p$ be the defining field and assume that $p \equiv 2 \pmod 3$ and $p \equiv 3 \pmod 4$. The order of these curves is $n = p + 1$. Let $r$ be the order of the subgroup of interest, and note that $r \mid p+1$. Note also that we can represent an element $t \in \mathbb{F}_{p^2}$ as $t = u + iv$ where $u, v \in \mathbb{F}_p$ and $i$ satisfies $i^2 + 1 = 0$ ($i \notin \mathbb{F}_p$ since $p \equiv 3 \pmod 4$). The final exponent is $z = (p^2-1)/r = ((p+1)/r)(p-1)$. To compute $s = w^z$, compute $t = w^{(p+1)/r} = u + iv$ and set $s = (u+iv)^p/(u+iv) = (u-v)/(u+iv)$ using the linearity of raising to the power $p$ and the fact that $i^p = -i$ for $p \equiv 3 \pmod 4$. By simplifying further, we can obtain $s = (u^2-v^2)/(u^2+v^2) - 2uvi/(u^2+v^2)$.

- $E_3$: Let $q = 2^l$. Without loss of generality, we want to compute $t_m$ with $m = q + 1 \pm \sqrt{2q}$ (see comment above in the section 'Choice of Groups'). Thus, the final exponent has the form $z = (q + 1 \pm \sqrt{2q})(q^2 - 1)$. We can compute

$s = w^z$ by first computing $t = w^q \cdot w \cdot w^{\pm\sqrt{2q}}$ and then $s = t^{q^2}/t$. Raising to the powers $q, \sqrt{2q}, q^2$ can be done in $O(l)$ time since they are all powers of 2, so the complete running time is dominated by the 3 multiplications and at most 2 inversions, which take time $O(l^2)$.

- $E_4$: Let $q = 3^l$. Without loss of generality, we want to compute $t_m$ with $m = q + 1 \pm \sqrt{3q}$ (see comment above in the section 'Choice of Groups'). Thus, the final exponent has the form $z = (q + 1 \pm \sqrt{3q})(q^3 - 1)(q + 1)$. We can compute $s = w^z$ by first computing $u = w^q \cdot w \cdot w^{\pm\sqrt{3q}}$, $t = u^{q^3}/q$ and then $s = t^q \cdot t$. Raising to the powers $q, \sqrt{3q}, q^3$ can be done in $O(l)$ time since they are all powers of 3, so the complete running time is dominated by the 4 multiplications and at most 2 inversions, which take time $O(l^2)$.

# Chapter 7

# Conclusion

This thesis presented several ways in which bilinear maps can be used to construct cryptographic schemes.

The most significant of these applications is certainly an efficient identity-based encryption scheme. It is also interesting to note that this scheme can easily be modified to have additional features, such as authenticated encryption, or support a hierarchy of PKGs.

Bilinear maps can also be used to construct a signature scheme with very short signatures, even it the message signed is also very short. We also presented an identity-based signature scheme, but this result is not as significant as the others since efficient identity-based signature schemes were already known ([18, 17]).

The one-round three party key agreement has special significance since it was one of the first constructive application of bilinear maps in cryptography. We showed how this original scheme can be modified into an authenticated key agreement protocol. However, the security arguments were only heuristic. It would be

interesting to see if the security definition given by Canetti and Krawczyk in [8] can be extended to three party protocols and if the schemes they present can be modified to a three party key agreement scheme using a bilinear map.

We also note that, with the improvement presented in Section 6.2.8, bilinear maps can now be efficiently implemented using the Tate pairing in elliptic curves so that the protocols above can be executed in times comparable to other encryption and signature schemes.

An interesting line for further research would be to find a way to implement multilinear maps, i.e. maps linear in more than two components. This topic has already been explored by Boneh and Silverberg in [7]. They give several interesting applications of such multilinear maps, but also give some evidence that the implementation of such maps may require genuinely new techniques. It would also be interesting to see if bilinear maps can be implemented using a technique other than pairings on abelian varieties.

# Bibliography


[1] S. Al-Riyami and K. Paterson. *Authenticated Three Party Agreement Protocols from Pairings*, Cryptology ePrint Archive, Report 2002/035, 2002. http://eprint.iacr.org/2002/035/

[2] P. Barreto, H. Kim, B. Lynn and M. Scott. *Efficient Algorithms for Pairing-Based Cryptosystems*, Proceedings of Crypto 2002, LNCS 2442, pp. 354-368, 2002.

[3] P. Barreto, B. Lynn and M. Scott. *Constructing Elliptic Curves with Prescribed Embedding Degrees*, Cryptology ePrint Archive, Report 2002/088, 2002. http://eprint.iacr.org/2002/088/

[4] A. Boldyreva. *Efficient Threshold Signature, Multisignature and Blind Signature Schemes Based on the Gap-Diffie-Hellman-Group Signature Scheme*, Cryptology ePrint Archive, Report 2002/118, 2002. http://eprint.iacr.org/2002/118/

[5] D. Boneh and M. Franklin. *Identity Based Encryption Scheme from*

*the Weil Pairing*, Cryptology ePrint Archive, Report 2001/090, 2001. http://eprint.iacr.org/2001/090/

[6] D. Boneh, B. Lynn and H. Shacham. *Short Signatures from the Weil Pairing*, Proceedings of Asiacrypt 2001, LNCS 2248, pp. 514-532, 2001.

[7] D. Boneh and A. Silverberg. *Applications of Multilinear Forms to Cryptography*, Cryptology ePrint Archive, Report 2002/080, 2002. http://eprint.iacr.org/2002/080/

[8] R. Canetti and H. Krawczyk. *Anaylsis of Key-Exchange Protocols and their Use for Building Secure Channels*, Cryptology ePrint Archive, Report 2001/040, 2001. http://eprint.iacr.org/2001/040/

[9] R. Canetti and H. Krawczyk. *Universally Composable Notions of Key Exchange and Secure Channels*, Cryptology ePrint Archive, Report 2002/059, 2002. http://eprint.iacr.org/2002/059/

[10] J. Cha and J. Cheon. *An Identity-Based Signature from Gap Diffie-Hellman Groups*, Cryptology ePrint Archive, Report 2002/018, 2002. http://eprint.iacr.org/2002/018/

[11] L. Charlap and D. Robbins. *An Elementary Introduction to Elliptic Curves*, CRD Exposition Report 31, Institute for Defense Analyses, 1988.

[12] J. Cheon. *A Universal Forgery of Hess's Second ID-based Signature against the Known-message Attack*, Cryptology ePrint Archive, Report 2002/028, 2002. http://eprint.iacr.org/2002/028/

[13] C. Cocks. *An Identity Based Encryption Scheme Based on Quadratic Residues*, Cryptography and Coding, LNCS 2260, pp. 360-363, 2001.

[14] J. Coron. *On the Exact Security of Full Domain Hash*, Proceedings of Crypto 2000, LNCS 1880, pp. 229-235, 2000.

[15] R. Dupont, A. Enge and F. Morain. *Building Curves with Arbitrary Small MOV Degree over Finite Prime Fields*, Cryptology ePrint Archive, Report 2002/094, 2002. http://eprint.iacr.org/2002/094/

[16] A. Enge. *Elliptic Curves and their Applications to Cryptography: An Introduction*, Kluwer Academic Publishers, 1999.

[17] U. Feige, A. Fiat and A. Shamir. *Zero-Knowledge Proofs of Identity*, Journal of Cryptology, Vol. 1, pp. 77-94, 1998.

[18] A. Fiat and A. Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, Proceedings of Crypto '86, LNCS 263, pp. 186-194, 1987.

[19] G. Frey and H. Rück. *A Remark Concerning m-Divisibility and the Discrete Logarithm in the Divisor Class Group of Curves*, Mathematics of Computation, Vol. 62, pp. 865-874, 1994.

[20] G. Frey, M. Müller and H. Rück. *The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems*, IEEE Transactions on Information Theory, Vol. 45, pp. 1717-1718, 1999.

[21] E. Fujisaki and T. Okamoto. *Secure Integration of Asymmetric and Symmetric Encryption Schemes*, Proceedings of Crypto '99, pp. 537-554, 1999.

[22] S. Galbraith. *Supersingular Curves in Cryptography*, Proceedings of Asiacrypt 2001, LNCS 2248, pp. 495-513, 2001. Full version available at http://www.isg.rhul.ac.uk/ sdg/ss.ps.gz

[23] S. Galbraith, K. Harrison and D. Soldera. *Implementing the Tate Pairing*, Proceedings of ANTS-V, LNCS 2369, pp. 324-337, 2002.

[24] P. Gaudry. *An Algorithm for Solving the Discrete Log Problem on Hyperelliptic Curves*, Eurocrypt 2000, LNCS 1807, pp. 19-34, 2000.

[25] P. Gemmell. *An Introduction to Threshold Cryptography*, Cryptobytes, a technical letter of RSA Laboratories, Vol. 2, No. 7, 1997.

[26] C. Gentry and A. Silverberg. *Hierarchical ID-Based Cryptography*, Cryptology ePrint Archive, Report 2002/056, 2002. http://eprint.iacr.org/2002/056/

[27] D. Hankerson, J. Hernandez and A. Menezes. *Software Implementation of Elliptic Curve Cryptography over Binary Fields*, Proceedings of CHES 2000, LNCS 1965, pp. 1-24, 2000.

[28] F. Hess. *Exponent Group Signature Schemes and Efficient Identity Based Signature Schemes Based on Pairings*, Cryptology ePrint Archive, Report 2002/012, 2002. http://eprint.iacr.org/2002/012/

[29] J. Horwitz and Ben Lynn. *Toward Hierarchical Identity-Based Encryption*, Proceedings of Eurocrypt 2002, LNCS 2332, pp. 466-481, 2002.

[30] A. Joux. *A One Round Protocol for Tripartite Diffie-Hellman*, Proceedings of ANTS-IV , LNCS 1838, pp. 385-394, Springer 2000.

[31] A. Joux and K. Nguyen. *Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups*, Cryptology ePrint Archive, Report 2001/003, 2001. http://eprint.iacr.org/2001/003/

[32] L. Law, A. Menezes, M. Qu and S. Vanstone. *An Efficient Protocol for Authenticated Key Agreement*, Technical Report CORR 98-05, Department of C&O, University of Waterloo, 1998.

[33] R. Lidl and H. Niederreiter. *Finite Fields*, Encyclopedia of Mathematics and its Applications 20, 2nd Edition, Cambridge Univeristy Press, 1997.

[34] B. Lynn. *Authenticated Identity-Based Encryption*, Cryptology ePrint Archive, Report 2002/072, 2001. http://eprint.iacr.org/2002/072/

[35] A. Lysyanskaya. *Unique Signatures and Verifiable Random Functions from the DH-DDH Separation*, Proceedings of Crypto 2002, LNCS 2442, pp. 597-612, 2002.

[36] J. Malone-Lee. *Identity-Based Signcryption*, Cryptology ePrint Archive, Report 2002/098, 2001. http://eprint.iacr.org/2002/098/

[37] T. Matsumoto, Y. Takashima and H. Imai. *On Seeking Smart Public-Key-Distribution Systems*, Trans. IECE of Japan, E68:99-106, 1986.

[38] U. Maurer. *Towards Proving the Equivalence of Breaking the Diffie-Hellman*

*Protocol and Computing Discrete Logarithms*, Proceedings of Crypto '94, pp. 271-281, 1994.

[39] A. Menezes. *Elliptic Curve Public Key Cryptography*, Kluwer Academic Publishers, 1993.

[40] A. Menezes, T. Okamoto and S. Vanstone. *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field*, IEEE Transactions on Information Theory, Vol. 39, pp. 1639-1646, 1993.

[41] A. Menezes, P. van Oorschot and S. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1997.

[42] V. Miller. *Short Programs for Functions on Curves*, unpublished manuscript, 1986.

[43] T. Okamoto and D Pointcheval. *The Gap Problems: A New Class of Problems for the Security of Cryptographic Primitives*, Public Key Cryptography, PKC 2001, LNCS 1992, pp. 104-118, 2001.

[44] K. Paterson. *ID-Based Signatures from Pairings on Elliptic Curves*, Cryptology ePrint Archive, Report 2002/004, 2002. http://eprint.iacr.org/2002/004/

[45] C. Rackoff and D. Simon. *Noninteractive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack*, Proceedings of Crypto '91, pp. 433-444, 1991.

[46] K. Rubin and A. Silverberg. *The Best and Worst of Supersingular Abelian Varieties in Cryptography*, Proceedings of Crypto 2002, LNCS 2442, pp. 336-353, 2002.

[47] R. Sakai, K. Ohgishi and M. Kasahara. *Cryptosystems Based on Pairing*, In SCIS 2000, Okinawa, Japan, January 2000.

[48] A. Shamir. *Identity-Based Cryptosystems and Signature Schemes*, Proceedings of Crypto '84, pp. 47-53, 1984.

[49] J. Silverman. *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics, Springer-Verlag, 1986.

[50] N. Smart. *An Identity Based Authenticated Key Agreement Protocol Based on the Weil Pairing*, Cryptology ePrint Archive, Report 2001/111, 2001. http://eprint.iacr.org/2001/111/

[51] E. Verheul. *Evidence that XTR is More Secure than Supersingular Elliptic Curve Cryptosystems*, Proceedings of Eurocrypt 2001, pages 195-210, 2001.

[52] E. Verheul. *Self-Blindable Credential Certificates from the Weil Pairing*, Proceedings of Asiacrypt 2001, pages 533-551, 2001.

# Appendix A

# Schemes for Unmodified Weil or Tate Pairing

We show here how the Boneh-Franklin encryption scheme and the Gap signature scheme can be modified to use the unmodified Weil or Tate pairing instead of a bilinear function.

The following must be taken into account when modifying the schemes:

1. The Weil and Tate pairing are not symmetric.

2. If $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$, then computing $t_m(P, Q)$ is likely to be much faster than computing $t_m(Q, P)$.

3. The elements used in the ciphertext or signature should be in the smaller field.

The security of the two schemes is based on the following problems: Let

$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ be cyclic groups of prime order $p$. Let $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ be a bilinear non-degenerate and efficiently computable function.

The *co-Computational Diffie-Hellman Problem* in $\langle \mathbb{G}_1, \mathbb{G}_2 \rangle$ is:
*Given a generator $P$ of $\mathbb{G}_1$, an elements $aP \in \mathbb{G}_1$ for a random in $\mathbb{Z}_p$ and an element $Q \in \mathbb{G}_2$, compute $aQ$.*

The *co-Bilinear Diffie-Hellman Problem* in $\langle \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, \hat{t} \rangle$ is :
*Given a generator $P$ of $\mathbb{G}_1$, two elements $aP, bP \in \mathbb{G}_1$ for $a, b$ random in $\mathbb{Z}_p$ and an element $Q \in \mathbb{G}_2$, compute $\hat{t}(P, Q)^{ab}$.*

We note that the Computational Diffie-Hellman problem and the Bilinear Diffie-Hellman problem are special cases of those two problems with $\mathbb{G}_1 = \mathbb{G}_2$. Therefore, the co-Computational Diffie-Hellman problem and co-Bilinear Diffie-Hellman problem are at least as hard as the Computational Diffie-Hellman problem and the Bilinear Diffie-Hellman problem respectively.

The function $\hat{t}$ can be $t_p$ or $e_p$.

## A.1 The Boneh-Franklin Scheme

Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ containing a cyclic subgroup of prime order $p$. Let $k$ be the security multiplier of the group of points of order $p$ in $E(\mathbb{F}_q)$. Let $\mathbb{G}_1$ be the cyclic subgroup of points of order $p$ in $E(\mathbb{F}_q)$ and $\mathbb{G}_2 \subset \mathbb{F}_{q^k}$ be the group of $p^{th}$ roots of unity.

Setup: Step 1: Pick a random point $P \in E(\mathbb{F}_q)$ of order $p$.

Step 2: Pick a random $s \in \mathbb{Z}_p^*$ and compute $P_{pub} = sP$.

Step 3: Pick cryptographic hash functions $H_1 : \{0,1\}^* \to \langle Q \rangle^*$, $H_2 : \mathbb{G}_2 \to \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_p^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$ where $Q \in E(\mathbb{F}_{q^k})$ is a point of order $p$ independent from $P$ and $n$ is a positive integer.

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $C = \langle P \rangle^* \times \{0,1\}^n$. The public system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{t}, p, n, P, P_{pub}, H_1, H_2 \rangle$. The $\mathsf{master\text{-}key}$ is $s \in \mathbb{Z}_p^*$.

Extract: Given a string $\mathsf{ID} \in \{0,1\}^*$ and the $\mathsf{master\text{-}key}$ $s$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{t}, p, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$,

Step 1: Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$.

Step 2: Compute $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ and return $d_{\mathsf{ID}}$.

Encrypt: Given a plaintext $M \in \mathcal{M}$, a public key $\mathsf{ID}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{t}, p, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$,

Step 1: Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$.

Step 2: Pick a random $\sigma \in \{0,1\}^n$.

Step 3: Compute $r = H_3(\sigma, M)$.

Step 4: Compute $g = t_p(P_{pub}, Q_{\mathsf{ID}}) \in \mathbb{G}_2^*$.

Step 5: Compute the ciphertext $\langle rP, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma) \rangle$.

Decrypt: Given a ciphertext $\langle U, V, W \rangle$ encrypted using the public key $\mathsf{ID}$, system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{t}, p, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$ and the private key $d_{\mathsf{ID}} \in G_1^*$,

Step 1: If $U \notin \langle P \rangle^*$ reject the ciphertext.

Step 2: Compute $g' = t_p(U, d_{\mathsf{ID}})$.

Step 3: Compute $\sigma = V \oplus H_2(g')$.

Step 4: Compute $M = W \oplus H_4(\sigma)$.

Step 5: Compute $r = H_3(\sigma, M)$. If $U \neq rP$ reject the ciphertext.

Step 6: Return $M$.

Consistency can be proven similarly as before.

The proof of security also remains the same, except that the security is now based on the co-Bilinear Diffie-Hellman Problem.

## A.2 The Gap Diffie-Hellman Signature Scheme

Let $E$ be an elliptic curve defined over $\mathbb{F}_q$ containing a cyclic subgroup of prime order $p$. Let $k$ be the security multiplier of the group of points of order $p$ in $E(\mathbb{F}_q)$. Let $\mathbb{G}_1$ be the cyclic subgroup of points of order $p$ in $E(\mathbb{F}_q)$ and $\mathbb{G}_2 \subset \mathbb{F}_{q^k}$ be the group of $p^{th}$ roots of unity.

Setup: Step 1: Pick an arbitrary point $P \in E(\mathbb{F}_{q^k})$ of order $p$, $P \notin E(\mathbb{F}_q)$.

Step 2: Pick a cryptographic hash function $H : \{0,1\}^* \rightarrow \langle Q \rangle^*$ where $Q \in E(\mathbb{F}_q)$ is a point of order $p$.

The system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{t}, p, P, H \rangle$.

KeyGen: Given the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{t}, p, P, H \rangle$, pick a random $x \in \mathbb{Z}_p^*$ and compute $K_{pub} = xP$. The public key is $K_{pub}$ and the private key is $x$.

Sign: Given a message $M \in \{0,1\}^*$, a private key $x$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{t}, p, P, H \rangle$,

Step 1: Compute $R = H(M)$.

Step 2: Compute $\sigma = xR$.

The signature on $M$ is $\sigma$.

Verify: Given a message-signature pair $\langle M, \sigma \rangle$, a public key $K_{pub}$ and system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{t}, p, P, H \rangle$,

Step 1: Compute $R = H(M)$.

Step 2: Compute $g_1 = \hat{t}(\sigma, P)$.

Step 3: Compute $g_2 = \hat{t}(R, K_{pub})$.

Step 4: If $g_1 = g_2$, return valid, else return invalid.

Consistency is proven similarly as in the original scheme.

Again, the proof of security remains the same, except that it is now based on the co-Computational Diffie-Hellman problem.