Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem

by

Kayo Yoshida

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

Waterloo, Ontario, Canada, 2009

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

The Boneh-Boyen signature scheme is a short signature scheme which is provably secure in the standard model under the q-Strong Diffie-Hellman (SDH) assumption. The primary objective of this thesis is to examine the relationship between the Boneh-Boyen signature scheme and SDH. The secondary objective is to survey surrounding topics such as the generic group model, related signature schemes, intractability assumptions, and the relationship to identity-based encryption (IBE) schemes. Along these lines, we analyze the plausibility of the SDH assumption using the generic bilinear group model. We present the security proofs for the Boneh-Boyen signature scheme from [14], with the addition of a small improvement in one of the probability bounds. Our main contribution is to give the reduction in the reverse direction; that is, to show that if the SDH problem can be solved then the Boneh-Boyen signature scheme can be forged. This contribution represents the first known proof of equivalence between the SDH problem and Boneh-Boyen signatures. We also discuss the algorithm of Cheon [25] for solving the SDH problem. We analyze the implications of Cheon's algorithm for the security of the Boneh-Boyen signature scheme, accompanied by a brief discussion on how to counter the attack.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. David Jao, who has supported me throughout the degree program with numerous pieces of advice.

I would like to thank my friends whom I met in Waterloo. You studied with me, you ate with me, you drank with me, you danced with me, you laughed with me. Every moment that I shared with each of you has been precious. Without you I could not have possibly come all the way to complete the degree.

I wish to thank my parents and my sister, who have supported me from overseas for more than six years since I left the country.

Dedication

This is dedicated to my parents.

Contents

1 Introduction **1**

1.1 Signature Schemes and Security Definitions 1

1.2 Bilinear Pairings 3

1.3 Short Signatures 4

1.4 Our Results 5

1.5 Outline 6

2 Intractability Assumptions **7**

2.1 Diffie-Hellman and Related Assumptions 7

2.1.1 Diffie-Hellman 7

2.1.2 co-Diffie-Hellman 8

2.1.3 Bilinear Diffie-Hellman 9

2.2 SDH and Related Assumptions 11

2.2.1 Strong Diffie-Hellman and Strong Diffie-Hellman′ 11

2.2.2 Diffie-Hellman Inversion 13

2.2.3 Modified and Hidden Strong Diffie-Hellman 14

2.2.4 2-Variable Strong Diffie-Hellman 15

2.3 Generic Security of the SDH′ Assumption 16

2.3.1 Random Oracles and Generic Groups 21

3 Boneh-Boyen Signatures and Related Schemes **25**

3.1 Boneh-Boyen Signatures 25

3.1.1 Boneh-Boyen Signatures: Version 1 26

3.1.2 Boneh-Boyen Signatures: Version 2 27

3.2 BLS Signatures 28

3.3 Waters Signature Scheme 29
3.4 Okamoto Signature Scheme 30
3.5 Identity-Based Encryption and Relationship to Signature Schemes . 31

4 Reductions Between Boneh-Boyen Signatures and the SDH′ Problem 35

4.1 Reduction from SDH′ to the Basic Signature Scheme 35
4.2 Reduction from SDH′ to the Full Signature Scheme 37
4.3 Method of Partial Fractions 40
4.4 Reduction from the Basic Signature Scheme to SDH′ 42
4.5 Reduction from the Full Signature Scheme to SDH′ 44
4.6 Boneh-Boyen Signatures (Original Version) 46

5 Security Analysis of Boneh-Boyen Signatures 49

5.1 Shanks' Baby-Step Giant-Step Algorithm 50
5.2 Cheon's Algorithm for SDH 50
5.2.1 Cheon's $p-1$ Algorithm 50
5.2.2 Cheon's $p+1$ Algorithm 52
5.3 Recovering the Private Key in the Boneh-Boyen Signature Scheme . 55
5.4 Countermeasures and Mitigation 57

6 Conclusions and Future Work 59

References 61

Chapter 1

Introduction

The notion of a *digital signature scheme* as a digital analog of hand-written signatures was introduced by Diffie and Hellman in 1976 [28]. The first concrete signature scheme to appear was the RSA signature scheme [49]. Many signature schemes followed after that: Rabin signatures [48], ElGamal signatures [30], Full Domain Hash (FDH) [9], Schnorr signatures [50], DSA [1], and Elliptic Curve DSA (ECDSA) [2, 3], to name a few.

In this thesis, we focus on the signature scheme recently introduced by Boneh and Boyen [13, 14]. We begin by reviewing some basic definitions and notations that are needed for the remainder of this thesis.

1.1 Signature Schemes and Security Definitions

In the analog world, we associate a signed document to a unique person, say Alice, according to her unique hand-writing. In the digital world, Alice's secret key, which she keeps secret from anyone else as the name suggests, plays the role of her unique hand-writing. The public key associated with Alice's secret key is used to verify whether or not a given signature was generated by Alice. We formally define a signature scheme as follows.

1.1.1 Definition. A *digital signature scheme* consists of a triplet of algorithms (*KeyGen*, *Sign*, *Verify*) satisfying the following properties.

- *KeyGen*, given some security parameter ℓ, outputs a random key pair $(\mathsf{PK}, \mathsf{SK})$ of size specified by ℓ. PK denotes a public key, and SK denotes a private key.
- *Sign* takes a message $m \in \mathcal{M}$ and a private key SK as input, and outputs a signature σ. We call the set $\mathcal{M}$ the *message space*.
- *Verify* receives a message m, a signature σ, and a public key PK as input, and outputs `true` or `false`.

We say that a signature scheme is *consistent* if, whenever the key pair $(\mathsf{SK}, \mathsf{PK})$ is properly generated by the algorithm *KeyGen*, it satisfies

$$\forall m \in \mathcal{M}, \quad \sigma \leftarrow \mathit{Sign}(m, \mathsf{SK}) \implies \mathit{Verify}(m, \sigma, \mathsf{PK}) = \texttt{true}.$$

The definition of consistency guarantees that Alice never gets rejected for a signature that she generates herself, which is clearly a property that any signature scheme should satisfy.

In addition to consistency, we require a signature scheme to be *secure*. In the world of hand-written signatures, Oscar should not be able to create a document which Alice did not sign, and have others accept it as a document signed by Alice. For digital signatures to have security characteristics analogous to hand-written signatures, we require that only a holder of a secret key SK can generate a signature σ corresponding to SK. The definition of security commonly used today is called *existential unforgeability under an adaptive chosen message attack*, and was established by Goldwasser, Micali, and Rivest [33]. We describe two versions of existential unforgeability — strong and weak — found in [13, 14], both defined in terms of games played between a challenger and a forger.

Strong Existential Unforgeability. Strong existential unforgeability is defined via the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger $\mathcal{C}$ generates a random key pair $(\mathsf{PK}, \mathsf{SK})$, and gives the public key PK to the adversary $\mathcal{A}$.

2. The adversary $\mathcal{A}$ can adaptively make up to q_S queries for signatures of messages $m_1, \ldots, m_{q_S} \in \mathcal{M}$ of its choice. Each time $\mathcal{C}$ receives a message m_i for $1 \leq i \leq q_S$, it must respond with a valid signature $\sigma_i = \mathit{Sign}(m_i, \mathsf{SK})$.

3. Eventually, $\mathcal{A}$ outputs a (message, signature) pair (m_*, σ_*), and wins the game if $(m_*, \sigma_*) \neq (m_i, \sigma_i)$ for all $i = 1, \ldots, q_S$ and $\mathit{Verify}(m_*, \sigma_*, \mathsf{PK}) = \texttt{true}$.

The adversary $\mathcal{A}$'s *advantage* in the above game is defined as

$$\operatorname{Adv} \operatorname{Sig}(\mathcal{A}) := \Pr\left[\mathit{Verify}(m_*, \sigma_*, \mathsf{PK}) = \texttt{true}\right],$$

where the probability is taken over the coin tosses made by $\mathcal{A}$ and $\mathcal{C}$.

1.1.2 Definition. An algorithm $\mathcal{A}$ is said to (t, q_S, ϵ)-*break a signature scheme* if $\mathcal{A}$ runs in time t, makes at most q_S signature queries, and $\operatorname{Adv} \operatorname{Sig}(\mathcal{A}) \geq \epsilon$. We say that a signature scheme is (t, q_S, ϵ)-*strongly existentially unforgeable under an adaptive chosen message attack* if there is no algorithm that (t, q_s, ϵ)-breaks it.

Weak Existential Unforgeability. Weak existential unforgeability is defined via the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The adversary $\mathcal{A}$ chooses up to q_S messages $m_1, \ldots, m_{q_S} \in \mathcal{M}$ and sends them to the challenger $\mathcal{C}$.
2. The challenger $\mathcal{C}$ generates a random key pair $(\mathsf{PK}, \mathsf{SK})$, and gives $\mathcal{A}$ the public key PK and valid signatures $\sigma_i = \mathit{Sign}(m_i, \mathsf{SK})$ for all $i = 1, \ldots, q_S$.
3. Eventually, $\mathcal{A}$ outputs a (message, signature) pair (m_*, σ_*), and wins the game if $m_* \neq m_i$ for all $i = 1, \ldots, q_S$ and $\mathit{Verify}(m_*, \sigma_*, \mathsf{PK}) = \mathtt{true}$.

The adversary $\mathcal{A}$'s advantage is defined as

$$\mathrm{Adv}\,\mathrm{Sig}\,\mathrm{W}(\mathcal{A}) = \Pr\left[\mathit{Verify}(m_*, \sigma_*, \mathsf{PK}) = \mathtt{true}\right]$$

where the probability is taken over the coin tosses made by $\mathcal{A}$ and $\mathcal{C}$.

1.1.3 Definition. An algorithm $\mathcal{A}$ is said to (t, q_S, ϵ)-*weakly break a signature scheme* if $\mathcal{A}$ runs in time t, makes at most q_S signature queries, and $\mathrm{Adv}\,\mathrm{Sig}\,\mathrm{W}(\mathcal{A}) \geq \epsilon$. We say that a signature scheme is (t, q_S, ϵ)-*existentially unforgeable under a weak chosen message attack* if there is no algorithm that (t, q_s, ϵ)-weakly breaks it.

The difference between strong and weak existential unforgeability is that, in the weak scenario, an adversary $\mathcal{A}$ is required to submit signature queries before receiving a public key. Thus, roughly speaking, signature schemes with strong existentially unforgeablility are 'more secure' than those with weak existential unforgeability. Boneh and Boyen [13, 14] use the notion of weak existential unforgeability as a building block to establish strong existential unforgeability for the full version of their signature scheme (see Section 4.1 and Section 4.2 for details).

1.2 Bilinear Pairings

Many recent cryptographic schemes, including Boneh-Boyen signatures, make use of *bilinear pairings*. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of order $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$, where p is prime. The operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are written multiplicatively.

1.2.1 Definition. A function $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is called a *bilinear pairing* if it satisfies the following conditions:

Bilinearity: For any $u_1, u_2, u \in \mathbb{G}_1$ and $v_1, v_2, v \in \mathbb{G}_2$,

$$e(u_1 u_2, v) = e(u_1, v) \cdot e(u_2, v) \quad \text{and} \quad e(u, v_1 v_2) = e(u, v_1) \cdot e(u, v_2).$$

Non-degeneracy: There exists $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$ such that $e(u, v) \neq 1_{\mathbb{G}_T}$.

A pair of groups $(\mathbb{G}_1, \mathbb{G}_2)$ of prime order p is called a *bilinear group pair* if there exists an efficiently computable bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ into a target group $\mathbb{G}_T$ of order p, and group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ can be computed efficiently.

It is often useful to classify pairings into three different types following the convention in [32]. We say that a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ or its corresponding pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is of:

Type 1 if $\mathbb{G}_1 = \mathbb{G}_2$ (i.e. the pairing is *symmetric*), or if there exists an isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ such that ψ and its inverse $\psi^{-1}\colon \mathbb{G}_1 \to \mathbb{G}_2$ are both efficiently computable.

Type 2 if $\mathbb{G}_1 \neq \mathbb{G}_2$ but there exists an isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ which is efficiently computable. (Without loss of generality, we can assume the direction of the isomorphism is from $\mathbb{G}_2$ to $\mathbb{G}_1$ by swapping $\mathbb{G}_1$ and $\mathbb{G}_2$ if necessary.)

Type 3 if $\mathbb{G}_1 \neq \mathbb{G}_2$ and there are no efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$.

1.3 Short Signatures

A good signature scheme is both secure and efficient. Naturally, there is a trade-off between these two goals. Making the algorithms more complex or making the key and signature sizes longer generally increases the level of security, but at the cost of decreased efficiency. Although balancing the two conflicting goals is of interest in any application, certain applications require especially short signatures. For example, a person may need to type in a digital signature when registering a purchased product, in which case short signatures are necessary to minimize typing. Likewise, when we imprint a digital signature on a barcode such as on a postage stamp, the information capacity is highly limited [42, 46].

The U.S. national standards DSA and ECDSA are relatively short compared to widely-used RSA-based signatures; both DSA and ECDSA achieve the 80-bit security level using signatures having length equal to 320 bits [26, 1, 2]. Several modifications for DSA signatures have been suggested [44, 46, 42, 5], with the shortest ones being half the length of the original DSA under certain conditions. We note, however, that none of these are always as short as 160 bits.

Boneh, Lynn, and Shacham [17, 18] proposed a short signature scheme, which is derived from the Boneh-Franklin *identity-based encryption (IBE)* scheme [15] by applying a generic transformation from an IBE scheme to a signature scheme. Widely known as the BLS signature scheme, their scheme achieves 80-bit security with signatures of length approximately 160 bits for messages of any length. Its security is based on a variant of the *computational Diffie-Hellman (CDH)* assumption. The drawback of BLS signatures is that the security proof makes use of the

random oracle model. Since no concrete hash function is capable of perfectly modeling a random oracle [9, 24], the use of the random oracle model is a controversial issue. For this reason, a short signature scheme that is provably secure under the standard model is of interest to cryptographers.

In 2004, Boneh and Boyen [13, 14] introduced a new signature scheme with signatures as short as BLS, and which is provably secure under the standard model. In order to prove the security of the new signature scheme, Boneh and Boyen introduced a new assumption called the *q-Strong Diffie-Hellman (q-SDH)* assumption. Roughly speaking, the q-SDH assumption in a cyclic group $\mathbb{G}$ of order p states that, given a generator $g \in \mathbb{G}$ and $g^x, g^{x^2}, \ldots, g^{x^q}$ for some secret exponent $x \in \mathbb{Z}_p^*$, it is hard to find a pair $(c, g^{\frac{1}{x+c}})$ where c is any elemnt of $\mathbb{Z}_p$. The problem of finding a pair $(c, g^{\frac{1}{x+c}})$ is called the q-SDH problem, and it can be reduced to the CDH problem.

1.4 Our Results

Previously, no equivalence was known between the security of the SDH assumption and the security of the Boneh-Boyen signature scheme. The original Boneh-Boyen paper [13, 14] provides a security reduction, but it only goes in one direction: namely, if the q-SDH assumption holds, then Boneh-Boyen signatures are unforgeable. In this thesis, we study the security of Boneh-Boyen signatures, and focus on proving the converse of the security proofs provided in [13, 14]. In other words, we show that solving SDH is at least as hard as forging Boneh-Boyen signatures.

The motivation for reducing the signature scheme to the underlying SDH problem is two-fold. One motivation is provided by the work of Koblitz and Menezes [36, 37]. They raised a concern about using a non-standard problem in a security proof. Inventing a new 'hard' problem seems to have become a common practice when one wants a provable security guarantee on a new cryptographic scheme and existing assumptions do not ensure the desired security property. Statements of such new problems tend to be more complicated than standard hard problems; many of them are even defined in terms of interactive games played between a challenger and a solver. They point out that there is a risk in using a non-standard problem because we often do not know the exact level of difficulty of such a problem. Encouraging study of such a problem is important in order to gauge its level of difficulty, and to assess the plausibility of the resulting assumption. Koblitz and Menezes thus suggest one or both of the following when introducing a new problem: use a problem which is interesting to study in its own right, or to prove that solving the problem is as hard as attacking the protocol. The latter ensures that solving the problem enables an attack on the protocol, so that cryptoanalysts are encouraged to study the problem. We thus implement the second suggestion in order to encourage more study regarding the SDH problem.

Another motivation for our work is provided by Cheon's recent analysis of the

q-SDH problem [25]. Prior to Cheon's result, the only way to solve the q-SDH problem was to use a generic *discrete logarithm (DLOG)* algorithm such as Pollard rho [47], which takes time $O(p^{\frac{1}{2}+\epsilon})$. Cheon [25] presents algorithms that can be used to solve the q-SDH problem for large q in $O(p^{\frac{1}{3}+\epsilon})$ time in most cases. Thus, using Cheon's algorithms in conjunction with the reduction in the reverse direction allows the forgery of Boneh-Boyen signatures in faster than $O(p^{\frac{1}{2}+\epsilon})$ time. In fact, our result indicates that under some circumstances Boneh-Boyen signatures can be forged in $O(p^{\frac{2}{5}} + \epsilon)$ time.

1.5 Outline

In Chapter 2, we define the strong Diffie-Hellman assumption and several related intractability assumptions. The two versions of the Boneh-Boyen paper ([13] and [14]) use slightly different definitions of SDH. Following the convention established in [19], we use the notation SDH and SDH$'$ to denote the versions appearing in [13] and [14] respectively. Note that although both versions of SDH are presented in Chapter 2, we mostly deal with SDH$'$ in this thesis. Boneh and Boyen analyze the security of the SDH$'$ assumption in the *generic bilinear group model*. Section 2.3 defines the generic bilinear group model, presents the generic group security bound for SDH$'$ from [14], and briefly discusses philosophical aspects of the generic (bilinear) group model.

Section 3.1 presents the basic and full versions of the Boneh-Boyen signature schemes as they appear in [13] and [14]. In Sections 3.2 to 3.4, we present several signature schemes based on assumptions presented in Chapter 2. We also briefly describe how the Boneh-Boyen and BLS signature schemes were derived; in fact, they are both obtained via a generic transformation from an IBE scheme. We introduce the notion of IBE in Section 3.5, and present the Boneh-Boyen IBE scheme from which the Boneh-Boyen signature scheme was derived.

Chapter 4 and Chapter 5 contain our main contributions. In Chapter 4, we first present the (forward) security proof for both the basic and full versions of the Boneh-Boyen signature scheme as given in [14]. In Section 4.4 and Section 4.5, we present the converse of the security theorems; that is, we show that forging both basic and full Boneh-Boyen signatures is no harder than solving SDH$'$.

Chapter 5 discusses how the theorems of Chapter 4 affect the security of the Boneh-Boyen signature schemes. We review Cheon's algorithms for solving the SDH problem. We then show how these algorithms can be used to forge Boneh-Boyen signatures. In Section 5.4 we discuss some possible ways to counter our attack.

Chapter 2

Intractability Assumptions

A number of pairing-based cryptographic protocols rely on the q-SDH problem and its variants as the basis for their security. In this chapter, we describe the q-SDH problem and other related problems [13, 14, 41, 12, 29, 19].

Many of the problems that we discuss come in two forms — *computational* and *decisional*. Generally speaking, a decisional problem asks to distinguish a random element and a valid output for its computational counterpart. Although the corresponding variants are usually obvious, we present both computational and decisional versions of the problems in this chapter, whenever helpful.

The problems we present in this chapter naturally give rise to associated intractability assumptions. Throughout this thesis, whenever we say "`xyz` assumption" where the `xyz` problem is defined, we mean that the `xyz` problem is *intractable*. In the case of a computational problem, intractable means no *probabilistic polynomial time (PPT) algorithm* can output a correct answer with non-negligible probability. In the case of a decisional problem, it means no PPT algorithm can output a correct answer with probability non-negligibly greater than 50%.

2.1 Diffie-Hellman and Related Assumptions

As its name suggest, the SDH problem is closely related to the classic *Diffie-Hellman (DH) problem* [28]. We start by reviewing this classic problem and its variations.

2.1.1 Diffie-Hellman

The idea behind the Diffie-Hellman problem first appeared in the classic key agreement protocol of Diffie and Hellman [28]. Some articles that study the DH problems or that use the DH assumptions include [8, 11, 38, 16, 54, 23, 39, 40, 7, 43, 10]. We recall both the computational and decisional versions of DH.

Computational Diffie-Hellman (CDH). [40]

Set up: a cyclic group $\mathbb{G}$ of prime order p

Input: $g, g^x, g^y \in \mathbb{G}$ where $g \in \mathbb{G}$ is a generator, and $x, y \in \mathbb{Z}_p$

Output: $g^{xy} \in \mathbb{G}$

The decisional version of the DH problem, first explicitly formulated by Brands [21], is given as follows.

Decisional Diffie-Hellman (DDH). [40]

Set up: a cyclic group $\mathbb{G}$ of prime order p

Input: $g, g^x, g^y, h \in \mathbb{G}$ where $g \in \mathbb{G}$ is a generator, $x, y \in \mathbb{Z}_p$, and $h \in \mathbb{G}$

Output: `true` if $h = g^{xy}$, `false` otherwise

It is easy to see from the definition that, if we have an algorithm to solve the DLOG problem (given $g, g^x \in \mathbb{G}$, output $x \in \mathbb{Z}_p$) then CDH can also be solved. Whether the converse is also true stands as one of the major open problems in cryptography.

Also, in general, if one has an algorithm to solve the computational version of a problem, then using the algorithm as a subroutine, one can immediately solve the decisional version of the problem. This principle applies to CDH and DDH. The converse of such a statement is usually not known to hold. Some cryptographic protocols utilize such a "gap" between the difficulty of the computational and decisional versions of a problem (see [18], for example).

2.1.2 co-Diffie-Hellman

The *co-Diffie-Hellman* problem was introduced by Boneh, Lynn, and Shacham [18, 17] to assist in the security proof of the BLS signature scheme.

Computational co-Diffie-Hellman (co-CDH). [18]

Set up: cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p, generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, and an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(g_2) = g_1$

Input: $g_2, g_2^x \in \mathbb{G}_2$ and $g_1^y \in \mathbb{G}_1$ where $x, y \in \mathbb{Z}_p$

Output: $g_1^{xy} \in \mathbb{G}_1$

Decisional co-Diffie-Hellman (co-DDH). [18]

Set up: cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p, generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, and an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(g_2) = g_1$

Input: $g_2, g_2^x \in \mathbb{G}_2$ and $g_1^y, h \in \mathbb{G}_1$ where $x, y \in \mathbb{Z}_p$

Output: `true` if $h = g_1^{xy}$, and `false` otherwise

2.1.1 Definition. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups of prime order p, and let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We say that $(g_2, g_2^x, g_1^y, h) \in \mathbb{G}_2^2 \times \mathbb{G}_1^2$ is a *co-Diffie-Hellman tuple* if $h = g_1^{xy}$.

We observe that the above problems are generalizations of CDH and DDH. When the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are identical, co-CDH and co-DDH are equivalent to CDH and DDH, respectively.

2.1.3 Bilinear Diffie-Hellman

The most natural extension of the Diffie-Hellman problem to a bilinear group setting is the series of *bilinear Diffie-Hellman (BDH)* problems. We present three variations of the BDH problems in order to account for the different types of pairings. The most basic BDH version appeared in Joux's construction of a three-party Diffie-Hellman key agreement protocol [35], and also in the security proof of the IBE scheme of Boneh and Franklin [15]. The three variations of the BDH problems, each with both computational and decisional problems, are as follows.

Computational BDH. [35, 15, 19]

Set up: a cyclic group $\mathbb{G}$ of prime order p with an efficiently computable bilinear map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Input: $g, g^x, g^y, g^z \in \mathbb{G}$, where $g \in \mathbb{G}$ is a generator and $x, y, z \in \mathbb{Z}_p$

Output: $e(g, g)^{xyz} \in \mathbb{G}_T$

Decisional BDH. [35, 15, 19]

Set up: a cyclic group $\mathbb{G}$ of prime order p with an efficiently computable bilinear map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Input: $g, g^x, g^y, g^z \in \mathbb{G}$ and $v \in \mathbb{G}_T$, where $g \in \mathbb{G}$ is a generator and $x, y, z \in \mathbb{Z}_p$

Output: `true` if $v = e(g, g)^{xyz}$, and `false` otherwise

Computational BDH′. [12]

Set up: cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p with an efficiently computable bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

Input: $g_1, g_1^x, g_1^z \in \mathbb{G}_1$ and $g_2, g_2^x, g_2^y \in \mathbb{G}_2$ where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators and $x, y, z \in \mathbb{Z}_p$

Output: $e(g_1, g_2)^{xyz} \in \mathbb{G}_T$

Decisional BDH′. [12]

Set up: cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p with an efficiently computable bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

Input: $g_1, g_1^x, g_1^z \in \mathbb{G}_1,\ g_2, g_2^x, g_2^y \in \mathbb{G}_2$, and $v \in \mathbb{G}_T$ where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators and $x, y, z \in \mathbb{Z}_p$

Output: `true` if $v = e(g_1, g_2)^{xyz}$, and `false` otherwise

2.1.2 Remark. BDH is computationally equivalent to BDH′ in symmetric pairings.

Proof. It is clear that BDH $\leq_P$ BDH′. In order to prove BDH $\geq_P$ BDH′, we use an algorithm $\mathcal{A}$ that solves the computational BDH problem to solve the computational BDH′ problem. Suppose we receive $g_1, g_1^x, g_1^z, g_2, g_2^x, g_2^y \in \mathbb{G}$ as input to the BDH′ problem. Let $a \in \mathbb{Z}_p$ be such that $g_2 = g_1^a$. If we give $g_1, g_1^x, g_2^y = g_1^{ay}, g_1^z$ as input to $\mathcal{A}$, the algorithm returns $e(g_1, g_1)^{axyz} = e(g_1, g_2)^{xyz}$, as desired. □

Computational BDH″. [19]

Set up: cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p with an efficiently computable bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

Input: $g_1^z \in \mathbb{G}_1$ and $g_2, g_2^x, g_2^y \in \mathbb{G}_2$ where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators and $x, y, z \in \mathbb{Z}_p$

Output: $e(g_1, g_2)^{xyz} \in \mathbb{G}_T$

Decisional BDH″. [19]

Set up: cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p with an efficiently computable bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

Input: $g_1^z \in \mathbb{G}_1,\ g_2, g_2^x, g_2^y \in \mathbb{G}_2$, and $v \in \mathbb{G}_T$ where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators and $x, y, z \in \mathbb{Z}_p$

Output: `true` if $v = e(g_1, g_2)^{xyz}$, and `false` otherwise

2.1.3 Remark. It is clear that BDH′ $\leq_P$ BDH″ for all types of pairings. Further, BDH″ is actually equivalent to BDH′ for Type 1 and Type 2 pairings. Below we prove that BDH″ $\leq_P$ BDH′ for Type 1 and Type 2 pairings.

Proof. We use an oracle for BDH′ to solve the BDH″ problem. Suppose we are given $g_1^z \in \mathbb{G}_1$ and $g_2, g_2^x, g_2^y \in \mathbb{G}_2$. Let $u \leftarrow \psi(g_2)$, and suppose $u = g_1^a$ for some $a \in \mathbb{Z}_p$. Then we can compute $v \leftarrow \psi(g_2^x) = u^x$. If we give the BDH′ oracle $u, v = u^x, g_1^z = u^{z/a} \in \mathbb{G}_1$ and $g_2, g_2^x, g_2^y \in \mathbb{G}_2$ as an instance of the BDH′ problem, then the oracle outputs $e(u, g_2)^{xyz/a} = e(g_1, g_2)^{xyz}$, as desired. □

To sum up, we have the following relationships between the hardness of the three variations of the BDH problems.

$$\text{BDH} \underset{\text{symmetric}}{=_P} \text{BDH}' \begin{array}{l} \underset{\text{Type 1, 2}}{=_P} \\ \underset{\text{Type 3}}{\leq_P} \end{array} \begin{array}{l} \text{BDH}'' \\ \text{BDH}'' \end{array}$$

2.2 SDH and Related Assumptions

In this section we review the class of *Strong Diffie-Hellman (SDH)*-type problems. These problems are related to the standard family of Diffie-Hellman problems. In most cases, the problems are parametrized by a parameter q, which is usually reflected in the notation, as in q-SDH.

2.2.1 Strong Diffie-Hellman and Strong Diffie-Hellman′

We start with the two variations of the strong Diffie-Hellman problems that are used in the Boneh-Boyen signature scheme [13, 14]. Below, we give the formal definition of the q-Strong Diffie-Hellman problem in a cyclic group pair $(\mathbb{G}_1, \mathbb{G}_2)$ [13], where q is some parameter. Note that we occasionally drop the parameter q, and simply call the problem SDH.

q-Strong Diffie-Hellman (q-SDH). [13]

Set up: cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p with an isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$

Input: a $(q+2)$-tuple $(g_1, g_2, g_2^x, \ldots, g_2^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, where $g_1 = \psi(g_2)$

Output: $(c, g_1^{\frac{1}{x+c}})$ for some $c \in \mathbb{Z}_p^*$ such that $x + c \neq 0$

The SDH problem forms the basis for the security of the version of the Boneh-Boyen signature scheme given in [13]. The full version of the paper [14] uses a slightly different version of SDH, which we call SDH′ in accordance with Boyen [19]. The *q-Strong Diffie-Hellman′ (q-SDH′)* problem in a cyclic group pair $(\mathbb{G}_1, \mathbb{G}_2)$, where q is some parameter, is given in [14] as follows:

q-Strong Diffie-Hellman′ (q-SDH′). [14]

Set up: cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p

Input: a $(q+3)$-tuple $(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$, where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators and $x \in \mathbb{Z}_p^*$

Output: $(c, g_1^{\frac{1}{x+c}})$ for some $c \in \mathbb{Z}_p$ such that $x + c \neq 0$

Since the SDH′ problem is central to this thesis, we give here a rigorous definition of the SDH′ assumption. A similar definition regarding SDH is omitted.

2.2.1 Definition. We define the *advantage* $\mathrm{Adv}\, q\text{-SDH}'(\mathcal{A})$ of a polynomial time algorithm $\mathcal{A}$ in solving the q-SDH′ problem in $(\mathbb{G}_1, \mathbb{G}_2)$ by

$$\mathrm{Adv}\, q\text{-SDH}'(\mathcal{A}) := \Pr\left[\mathcal{A}(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x) = (c, g_1^{\frac{1}{x+c}})\right],$$

where the probability is taken over the random choice of generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, the random choice of $x \in \mathbb{Z}_p^*$, and the coin tosses made by $\mathcal{A}$. An algorithm $\mathcal{A}$ is said to (t, ϵ)-*break the q-SDH′ problem in* $(\mathbb{G}_1, \mathbb{G}_2)$ if $\mathcal{A}$ runs in time t and $\mathrm{Adv}\, q\text{-SDH}'(\mathcal{A}) \geq \epsilon$. We say that the (q, t, ϵ)-*SDH′ assumption* holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if there is no algorithm that (t, ϵ)-breaks the q-SDH′ problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

We occasionally refer to the (q, t, ϵ)-SDH′ assumption as the q-SDH′ assumption, or sometimes simply as the SDH′ assumption. Wei and Yuen [57] observed the following reduction between SDH and SDH′.

2.2.2 Lemma. *If an isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ can be computed in time T_ψ, then the (q, t, ϵ)-SDH assumption implies the $\left(q, t - \Theta(q^2T + qT_\psi), \frac{p-1}{p-2}\epsilon\right)$-SDH′ assumption, where T is the upper bound for the time needed for one exponentiation in $\mathbb{G}_2$.*

Proof. Suppose $\mathcal{A}$ is an algorithm that (t', ϵ')-breaks the q-SDH′ problem. We construct an algorithm that $(t' + \Theta(q^2T + qT_\psi), \frac{p-2}{p-1}\epsilon')$-breaks the q-SDH problem. We are given an input

$$g_1, g_2, g_2^x, \ldots, g_2^{x^q}$$

to the q-SDH problem, where $x \in \mathbb{Z}_p^*$ is random, $g_2 \in \mathbb{G}_2$ is a random generator, and $g_1 = \psi(g_2)$. We pick $r \in \mathbb{Z}_p^*$ and $s \in \mathbb{Z}_p$ randomly such that $x + s \neq 0$. Let $h_2 \leftarrow g_2^r$ and $y \leftarrow x + s$. For each $k = 1, 2, \ldots, q$, we calculate

$$\prod_{i=1}^{k} \left(g_2^{x^i}\right)^{\binom{k}{i}s^{k-i}} = g_2^{\sum_{i=1}^{k}\binom{k}{i}s^{k-i}x^i} = g_2^{(x+s)^k} = g_2^{y^k}$$

and apply the mapping ψ to each $g_2^{y^k}$ to obtain $g_1^y, g_1^{y^2}, \ldots, g_1^{y^q}$. We also compute $g_2^r = h_2$ and $(g_2^y)^r = h_2^y$. We give

$$g_1, g_1^y, \ldots, g_1^{y^q}, h_2, h_2^y$$

to $\mathcal{A}$. Because of the randomizing effect of r and s, the generators $g_1 \in \mathbb{G}_1$ and $h_2 \in \mathbb{G}_2$ are random generators that are independent of each other.

Thus, with probability ϵ, the algorithm $\mathcal{A}$ outputs $(c, g_1^{\frac{1}{y+c}}) = (c, g_1^{\frac{1}{x+s+c}})$ for some $c \in \mathbb{Z}_p$ such that $x+s+c \neq 0$. Since it is information-theoretically impossible for $\mathcal{A}$ to know the value of s, it follows that $s+c$ is uniform in $\mathbb{Z}_p \backslash \{-x\}$. Therefore, $s + c \in \mathbb{Z}_p^*$ occurs with probability $\frac{p-2}{p-1}$. The process of calculating $g_2^y, \ldots, g_2^{y^q}$ takes $\Theta(q^2T)$ time, and mapping these to $g_1^y, \ldots, g_1^{y^q}$ takes $\Theta(qT_\psi)$ time; hence the whole process takes time $t + \Theta(q^2T + qT_\psi)$. □

2.2.2 Diffie-Hellman Inversion

Note that a solver of the SDH problem and the SDH′ problem can freely choose the value $c \in \mathbb{Z}_p$, allowing many possible correct solutions for those problems. We now present the two variants of the q-SDH problem, known as the *q-Diffie-Hellman Inversion (q-DHI)* problem and the *q-Diffie-Hellman Inversion′ (q-DHI′)* problem, in which the value of c is specified in the input. Afterwards, we present the *q-weak Diffie-Hellman (q-wDH)* problem, which is computationally equivalent to the q-DHI problem. Unlike SDH, these problems are defined over a single cyclic group $\mathbb{G}$ of prime order p.

q-Diffie-Hellman Inversion (q-DHI). [19, 12, 41, 29]

Set up: a cyclic group $\mathbb{G}$ of prime order p

Input: a $(q+1)$-tuple $(g, g^x, \ldots, g^{x^q}) \in \mathbb{G}^{q+1}$, where $g \in \mathbb{G}$ is a generator and $x \in \mathbb{Z}_p^*$

Output: $g^{1/x}$

q-Diffie-Hellman Inversion′ (q-DHI′).

Set up: a cyclic group $\mathbb{G}$ of prime order p

Input: a $(q+1)$-tuple $(g, g^x, \ldots, g^{x^q}, c) \in \mathbb{G}^{q+1} \times \mathbb{Z}_p$, where $g \in \mathbb{G}$ is a generator, $x \in \mathbb{Z}_p^*$, and $x + c \neq 0$

Output: $g^{\frac{1}{x+c}}$

We observe that, given an algorithm to solve the DHI problem in $\mathbb{G}_1$, the SDH′ problem can immediately be solved. Similarly, if there is an algorithm that solves the DHI problem in $\mathbb{G}_2$ and if there exists an isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$ that maps g_2 to g_1, then the SDH problem can be solved. The converse of these reductions are not known to hold. We also obtain the following reduction between DHI and DHI′.

2.2.3 Lemma. *The q-DHI problem in $\mathbb{G}$ is computationally equivalent to the q-DHI′ problem in $\mathbb{G}$ up to polynomials in* $\log p$ *and* q.

Proof. Clearly, if an oracle for the q-DHI′ problem is available, then we can solve the q-DHI problem by letting $c = 0$. Conversely, suppose $\mathcal{A}$ is an oracle that solves the q-DHI problem. Given an instance $(g, g^x, \ldots, g^{x^q}, c)$ of the q-DHI′ problem, we compute $g^{\frac{1}{x+c}}$ using $\mathcal{A}$. For each $k = 1, 2, \ldots, q$, we calculate

$$g_k \leftarrow \prod_{i=1}^{k} \left(g^{x^i}\right)^{\binom{k}{i} c^{k-i}} = g^{\sum_{i=1}^{k} \binom{k}{i} c^{k-i} x^i} = g^{(x+c)^k}.$$

If we give $(g, g_1, \ldots, g_q)$ to $\mathcal{A}$ as an input to the q-DHI problem, then $\mathcal{A}$ will output $g^{\frac{1}{x+c}}$, as desired. The reduction requires $\Theta(q^2)$ exponentiations in $\mathbb{G}$, which is polynomial in q and $\log p$. □

Next, we define the q-weak Diffie-Hellman (q-wDH) problem in $\mathbb{G}$.

q-Weak Diffie-Hellman (q-wDH).

Set up: a cyclic group $\mathbb{G}$ of prime order p

Input: a $(q+1)$-tuple $(g, g^x, \ldots, g^{x^q}) \in \mathbb{G}^{q+1}$, where $g \in \mathbb{G}$ is a generator and $x \in \mathbb{Z}_p^*$

Output: $g^{x^{q+1}}$

Some authors [41, 25] use the term "weak Diffie-Hellman" to refer to DHI. Also, Cheon [25] refers to our wDH as the "strong Diffie-Hellman" problem. Note, however, that no computational equivalence is known between SDH in Section 2.2.1 and wDH. Instead, the following equivalence is known.

2.2.4 Lemma. *The q-wDH problem in $\mathbb{G}$ is computationally equivalent to the q-DHI problem in $\mathbb{G}$ up to polynomials in $\log p$.*

Proof. Let $\mathcal{A}$ be an oracle that solves the q-wDH problem, and suppose an instance $(g, g^x, \ldots, g^{x^q})$ of the q-DHI problem is given. We compute $g^{\frac{1}{x}}$ using $\mathcal{A}$. Let $h \leftarrow g^{x^q}$ and let $y \leftarrow 1/x$ in $\mathbb{Z}_p$. We know h is a generator because $x \neq 0$ implies $p \nmid x^q$. Observe that $h^y = g^{x^{q-1}}$, $h^{y^2} = g^{x^{q-2}}$, $\ldots$, $h^{y^{q-1}} = g^x$, $h^{y^q} = g$. If we give $\mathcal{A}$ the $(q+1)$-tuple $(g^{x^q}, \ldots, g^x, g) = (h, h^y, \ldots, h^{y^q})$ as an input to the q-wDH problem, then $\mathcal{A}$ will return $h^{y^{q+1}} = g^{1/x}$, as desired.

Conversely, let $\mathcal{B}$ be an oracle that solves the q-DHI problem, and suppose an instance $(g, g^x, \ldots, g^{x^q})$ of the q-wDH problem is given. We compute $g^{x^{q+1}}$ using $\mathcal{B}$. Let $h \leftarrow g^{x^q}$ and let $y \leftarrow 1/x$ in $\mathbb{Z}_p$ as above. Then, if we give $\mathcal{B}$ the $(q+1)$-tuple $(g^{x^q}, \ldots, g^x, g) = (h, h^y, \ldots, h^{y^q})$ as an input to the q-DHI problem, then $\mathcal{B}$ will return $h^{1/y} = g^{x^{q+1}}$, as desired. □

Bao et al. [7] observe the special case of the above lemma where $q = 1$. They also note the equivalence between 1-wDH and CDH. These observations can also be found in [10].

2.2.3 Modified and Hidden Strong Diffie-Hellman

Further variations of the q-SDH problem can be found in the work of Boyen and Waters [20, 19]. The *modified q-SDH (q-MSDH)* problem in a cyclic group $\mathbb{G}$ is as follows:

q-Modified Strong Diffie-Hellman (q-MSDH). [20, 19]

Set up: a cyclic group $\mathbb{G}$ of prime order p

Input: a generator $g \in \mathbb{G}$, a group element g^x where $x \in \mathbb{Z}_p^*$, and $q-1$ pairs $(c_1, g^{\frac{1}{x+c_1}}), \ldots, (c_{q-1}, g^{\frac{1}{x+c_{q-1}}})$ where each $c_i \in \mathbb{Z}_p$

Output: $(c, g^{\frac{1}{x+c}})$ for some $c \in \mathbb{Z}_p \setminus \{c_1, \ldots, c_{q-1}\}$

The above formulation corresponds to the version given in [19]. The version given in [20] differs slightly in that the values of c_i and c are required to be nonzero. The difference is not significant, because the two versions reduce to each other in a manner similar to that given in the proof of Lemma 2.2.2.

Solving the q-MSDH problem is essentially equivalent to existentially forging Boneh-Boyen basic signatures (Section 3.1) given $q-1$ known (message, signature) pairs. Boyen [19] remarks that the MSDH assumption is weaker than the SDH assumption. Our result in Section 4.4, however, implies that solving $(q+1)$-MSDH is in fact no harder than solving q-SDH.

A computationally harder version of MSDH, where g, g^x are not given and the constant $c \in \mathbb{Z}_p$ is prescribed in input, appears in Mitsunari et al.'s paper [41]. They call this problem the collusion attack (CA) problem, and they show that the CA problem is equivalent to the DHI problem (wDH in their terminology). Our result, although discovered independently, represents an extension of their result to the (M)SDH setting, and relies on many of the same techniques.

Another problem, known as the *q-Hidden Strong Diffie-Hellman* problem or *HSDH* problem, is also introduced in [20].

q-Hidden Strong Diffie-Hellman (q-HSDH). [20, 19]

Set up: a cyclic group $\mathbb{G}$ of prime order p

Input: $g, g^x, h \in \mathbb{G}$ and $q-1$ triples $(g^{c_1}, g^{\frac{1}{x+c_1}}, h^{c_1}), \ldots, (g^{c_{q-1}}, g^{\frac{1}{x+c_{q-1}}}, h^{c_{q-1}})$ where each $c_i \in \mathbb{Z}_p$

Output: $(g^c, g^{\frac{1}{x+c}}, h^c)$ for some $c \in \mathbb{Z}_p \setminus \{c_1, \ldots, c_{q-1}\}$

HSDH can be viewed as a variation of MSDH where the constants c_i's are not given in the clear. We need to raise not only g but also h to the power of each c_i in order to prevent a trivial attack. To see this, suppose the required output of HSDH were of the form $(g^c, g^{\frac{1}{x+c}})$. If $(g^{c_1}, g^{\frac{1}{x+c_1}})$ were given as part of the input, we observe that

$$\frac{g^{\frac{1}{x+c_1}}}{g^x} = g^{\frac{1}{x+c_1}-x} = g^c \quad \text{and} \quad g^x g^{c_1} = g^{\frac{1}{x+\left(\frac{1}{x+c_1}-x\right)}} = g^{\frac{1}{x+c}}$$

where $c = \frac{1}{x+c_1} - x$. Thus, $\left(g^{\frac{1}{x+c_1}}/g^x, g^x g^{c_1}\right)$ would be a trivial solution for HSDH unless $c = c_i$ for one of the input pairs $(g^{c_i}, g^{\frac{1}{x+c_i}})$.

2.2.4 2-Variable Strong Diffie-Hellman

The *q-2-Variable Strong Diffie-Hellman (q-2SDH)* assumption and its variant (*q-2SDH$_S$*) arise in the work of Okamoto [45], where it is used to prove the security of

a signature scheme and a partially-blind signature scheme based on it. The q-2SDH problem and the q-2SDH$_S$ problem are stated as follows:

q-2-Variable Strong Diffie-Hellman (q-2SDH). [45]

Set up: a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$

Input: a $(2q+6)$-tuple $(g_1, g_2, g_2^x, \ldots, g_2^{x^q}, g_2^y, g_2^{yx}, \ldots, g_2^{yx^q}, g_2^{\frac{y+b}{x+a}}, a, b)$ where $g_2 \in \mathbb{G}_2$ is a generator, $g_1 \leftarrow \psi(g_2)$, and $x, y, a, b \in \mathbb{Z}_p^*$

Output: $(c, g_1^{\frac{1}{x+c}})$ where $c \in \mathbb{Z}_p^*$

Variant of the q-2-Variable Strong Diffie-Hellman (q-2SDH$_S$). [45]

Set up: a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$

Input: a $(3q+4)$-tuple $(g_1, g_2, g_2^x, g_2^y, g_2^{\frac{y+b_1}{x+a_1}}, \ldots, g_2^{\frac{y+b_q}{x+a_q}}, g_2^{a_1}, \ldots, g_2^{a_q}, b_1, \ldots, b_q)$ where $g_2 \in \mathbb{G}_2$ is a generator, $g_1 \leftarrow \psi(g_2)$, and $x, y, a_1, \ldots, a_q, b_1, \ldots, b_q \in \mathbb{Z}_p^*$

Output: $(g_2^c, g_1^{\frac{y+d}{x+c}}, d)$ for some $d \in \mathbb{Z}_p^* \setminus \{b_1, \ldots, b_q\}$

It is clear that the q-2SDH assumption is stronger than both the q-SDH and the q-SDH′ assumptions. Whether either of 2 SDH or 2 SDH$_S$ reduces to the other is not known.

2.3 Generic Security of the SDH′ Assumption

In order to give some assurance as to the security of the Boneh-Boyen signature scheme, Boneh and Boyen [14] prove a lower bound on the complexity of the q-SDH′ problem in the *generic bilinear group model*. In this section we review their derivation of the lower bound.

The definition of the generic bilinear group model in [14] is based on the *generic group model* introduced earlier by Shoup [53] to prove the corresponding lower bounds for the discrete logarithm and related problems. Intuitively, the generic group model is where groups and pairings are only available at the most abstract level without any specific implementations of them. The formal definition is as follows.

2.3.1 Definition. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order p such that a bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ exists. Let S_1, S_2, and S_T be sets of bit strings of cardinality at least p. We say that $(\mathbb{G}_1, \mathbb{G}_2)$ is a *generic bilinear group pair with target group* $\mathbb{G}_T$, if the following conditions hold.

1. The group order p is known.

2. Each element of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ is represented by an arbitrary unique string in S_1, S_2, and S_T, respectively.

3. Computations of group operations (multiplications and/or divisions) in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, a bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, an isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$, and the inverse $\psi^{-1}\colon \mathbb{G}_1 \to \mathbb{G}_2$ are only available through oracles.

The model of computation in which an algorithm has to work in a generic bilinear group pair is called the *generic bilinear group model*. We say that an algorithm is a *generic algorithm* if it works in the generic group model.

There are several things to note. First, the above definition implies that a generic algorithm can test the equality in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ without the help of oracles. Second, a generic algorithm can compute the identity element of any of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ by asking the respective group operation oracle to compute h/h for any element h. Third, a generic algorithm can also implement a $O(\log p)$ time square and multiply exponentiation algorithm in each group using the corresponding group operation oracle.

Also note that we provide oracles for computing isomorphisms ψ and ψ^{-1}, even though efficiently computable isomorphisms are not always available in an actual bilinear group pair [32]. This does not cause a problem in our context, because allowing more power to the adversary can only strengthen the result we obtain.

In order to derive a lower bound for q-SDH$'$, we first prove in Theorem 2.3.2 an upper bound for the advantage $\mathrm{Adv}\, q\text{-SDH}'$ in the generic group model, and then we turn the probability bound into a lower bound for the time complexity in Theorem 2.3.3.

2.3.2 Theorem. *[14, Theorem 12] Let $\mathcal{A}$ be an algorithm that solves the q-SDH$'$ problem in the generic bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with target group $\mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are of prime order p. Suppose $\mathcal{A}$ makes a total of q_G queries to oracles for the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, a bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, an isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$, and the inverse $\psi^{-1}\colon \mathbb{G}_1 \to \mathbb{G}_2$. Then,*

$$\epsilon := \mathrm{Adv}\, q\text{-SDH}'(\mathcal{A}) \le \frac{(q_G + q + 3)^2 (q+1)}{p-1}.$$

Asymptotically, we have

$$\epsilon = O\left(\frac{q_G^2 q + q^3}{p}\right).$$

Proof. We prove the theorem via an adversarial argument. We simulate the generic bilinear group model and the challenger of the q-SDH$'$ problem; we provide $\mathcal{A}$ with a random q-SDH$'$ instance, and simulate the oracles that interact with $\mathcal{A}$. We say that $\mathcal{A}$ wins the game if we fail to simulate the generic group model or if $\mathcal{A}$

successfully gives a correct q-SDH$'$ output. We argue that $\mathrm{Adv}\, q\text{-SDH}'(\mathcal{A})$ in the generic bilinear group model is bounded above by the probability that $\mathcal{A}$ wins the game, which, in turn, is bounded above by $\frac{(q_G+q+3)^2(q+1)}{p-1}$.

We first pick a random $x \in \mathbb{Z}_p^*$ and random generators g_1, g_2, and g_T of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively. The $(q+3)$-tuple $(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ represented as an element in $S_1^{q+1} \times S_2^2$ is the q-SDH$'$ instance that $\mathcal{A}$ must solve. Our aim is to simulate the oracles in such a way that no information about the value of x is leaked during the interactions with $\mathcal{A}$.

During the simulation, we maintain three lists $L_1 = \{(F_{1,i}, \xi_{1,i})\colon i = 1, \ldots, k_1\}$, $L_2 = \{(F_{2,i}, \xi_{2,i})\colon i = 1, \ldots, k_2\}$, and $L_T = \{(F_{T,i}, \xi_{T,i})\colon i = 1, \ldots, k_T\}$, where k_1, k_2, and k_T are counters that keep track of the number of elements in the lists. The strings $\xi_{1,i} \in S_1$, $\xi_{2,i} \in S_2$, and $\xi_{T,i} \in S_T$ are bit strings given to $\mathcal{A}$ as encodings of elements of $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, respectively. The polynomials $F_{1,i}$ and $F_{2,i}$ are of degree at most q in $\mathbb{Z}_p[X]$, and the $F_{T,i}$ are polynomials of degree at most $2q$ in $\mathbb{Z}_p[X]$.

The three lists L_1, L_2, and L_T are initialized by an "encoding" of the q-SDH instance $(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x)$ as follows; we set $F_{1,i} \leftarrow X^{i-1}$ for $i = 1, \ldots, q+1$, $F_{2,1} \leftarrow 1$, and $F_{2,2} \leftarrow X$; we let $\xi_{1,1}, \ldots, \xi_{1,q+1}$ be distinct random bit strings from S_1, and let $\xi_{2,1}$ and $\xi_{2,2}$ be distinct random bit strings from S_2; we let $k_1 \leftarrow q+1$, $k_2 \leftarrow 2$, and $k_T \leftarrow 0$.

We give the instance $(\xi_{1,1}, \ldots, \xi_{1,q+1}, \xi_{2,1}, \xi_{2,2}) \in S_1^{q+1} \times S_2^2$ to $\mathcal{A}$.

The algorithm $\mathcal{A}$ may make at most q_G oracle queries. Since the encodings of group elements are random bit strings, we may assume that $\mathcal{A}$ only makes oracle queries on strings that are already obtained through the input of the q-SDH problem or through previous oracle queries. We respond to each type of query as follows.

Group operation. Suppose $\mathcal{A}$ asks to perform a multiplication or a division in $\mathbb{G}_1$. Let $\xi_{1,i}$ and $\xi_{1,j}$ (where $1 \le i, j \le k_1$) be the two operands that $\mathcal{A}$ gives to us. We start by incrementing the counter k_1 by one. Next, we let $F_{1,k_1} \leftarrow F_{1,i} + F_{1,j}$ if $\mathcal{A}$ asked for a multiplication, and let $F_{1,k_1} \leftarrow F_{1,i} - F_{1,j}$ if $\mathcal{A}$ asked for a division. Note that F_{1,k_1} has degree at most q.

If the list L_1 already has an entry $(F_{1,\ell}, \xi_{1,\ell})$ for some $\ell < k_1$ with an identical polynomial $F_{1,\ell} = F_{1,k_1}$, then we copy the encoding of this entry. That is, we let $\xi_{1,k_1} \leftarrow \xi_{1,\ell}$. Otherwise, we assign ξ_{1,k_1} a random string from $S_1 \setminus \{\xi_{1,1}, \ldots, \xi_{1,k_1-1}\}$. We append the pair (F_{1,k_1}, ξ_{1,k_1}) to L_1, and return the string ξ_{1,k_1} to $\mathcal{A}$.

Queries on group operations in $\mathbb{G}_2$ and $\mathbb{G}_T$ are responded to similarly using the respective lists L_2 and L_T.

Bilinear pairing. Suppose $\mathcal{A}$ asks to perform a bilinear pairing from $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $\xi_{1,i}$ and $\xi_{2,j}$ with $1 \le i \le k_1$ and $1 \le j \le k_2$ be the two operands that $\mathcal{A}$ gives to us. We first increment the counter k_T by one. Next, we let $F_{T,k_T} \leftarrow F_{1,i} \cdot F_{2,j}$, so that F_{T,k_T} is a polynomial of degree at most $2q$.

If the list L_T already has an entry $(F_{T,\ell}, \xi_{T,\ell})$ for some $\ell < k_T$ with an identical polynomial $F_{T,\ell} = F_{T,k_T}$, then we duplicate the encoded string of this entry. That is, we let $\xi_{T,k_T} \leftarrow \xi_{T,\ell}$. Otherwise, we assign ξ_{T,k_T} a random string from $S_T \setminus \{\xi_{T,1}, \ldots, \xi_{T,k_T-1}\}$. We append the entry (F_{T,k_T}, ξ_{T,k_T}) to the list L_T, and return the string ξ_{T,k_T} to $\mathcal{A}$.

Isomorphism. $\mathcal{A}$ may ask to perform an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. Suppose $\mathcal{A}$ gives us the string $\xi_{2,i}$ for some $1 \leq i \leq k_2$. Upon receiving the query, we increment the counter k_1 by one. Next, we copy the polynomial corresponding to the input string to our new polynomial. That is, we let $F_{1,k_1} \leftarrow F_{2,i}$.

If the list L_1 happens to have a previous entry $(F_{1,\ell}, \xi_{1,\ell})$, $\ell < k_1$ with the same polynomial $F_{1,\ell} = F_{1,k_1}$, then we simply copy the corresponding string: $\xi_{1,k_1} \leftarrow \xi_{1,\ell}$. Otherwise, we assign ξ_{1,k_1} a random string from $S_1 \setminus \{\xi_{1,1}, \ldots, \xi_{1,k_1-1}\}$. We append the entry (F_{1,k_1}, ξ_{1,k_1}) to the list L_1, and return the string ξ_{1,k_1} to $\mathcal{A}$.

A query on the inverse isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ is responded to in a similar manner. We receive a string in S_1, calculate and append a new entry to the list L_2, and return a string in S_2.

After q_G queries, we have that $q_G = k_1 + k_2 + k_T - (q+3)$. Suppose $\mathcal{A}$ outputs a pair $(c, \xi_{1,\ell})$ for some $c \in \mathbb{Z}_p$ and $1 \leq \ell \leq k_1$ as an answer to the q-SDH$'$ problem. We analyze the probability that $\mathcal{A}$ wins the game.

Our simulation of the generic bilinear group model provided to $\mathcal{A}$ is flawed if and only if there are two distinct polynomials within L_1, L_2, or L_T that assume the same value when evaluated at $X \leftarrow x$. Otherwise, our simulation was perfect, and if the output provided by $\mathcal{A}$ is correct then

$$F_{1,\ell}(x) = \frac{1}{x+c} \iff (x+c)F_{1,\ell}(x) = 1.$$

Thus, the probability that $\mathcal{A}$ wins the game is at most $\Pr[E_1 \cup E_2 \cup E_T \cup E_{\mathcal{A}}]$ where the events E_1, E_2, E_T, and $E_{\mathcal{A}}$ are defined as follows:

$\boldsymbol{E_{\mathcal{A}}}$ — $(x+c)F_{1,\ell}(x) = 1$.

$\boldsymbol{E_1}$ — There exist distinct polynomials $F_{1,i}$ and $F_{1,j}$ with $1 \leq i, j \leq k_1$ such that $F_{1,i}(x) = F_{1,j}(x)$.

$\boldsymbol{E_2}$ — There exist distinct polynomials $F_{2,i}$ and $F_{2,j}$ with $1 \leq i, j \leq k_2$ such that $F_{2,i}(x) = F_{2,j}(x)$.

$\boldsymbol{E_T}$ — There exist distinct polynomials $F_{T,i}$ and $F_{T,j}$ with $1 \leq i, j \leq k_T$ such that $F_{T,i}(x) = F_{T,j}(x)$.

We further define

$\boldsymbol{E_{\mathcal{B}}}$ — $E_1 \cup E_2 \cup E_T$.

Consider the interaction between the actual generic bilinear group oracles and $\mathcal{A}$. We may assume that the oracles keep track of the lists of polynomials L'_1, L'_2, and L'_T with respective lengths k'_1, k'_2, and k'_T, similar to the lists L_1, L_2, and L_T that we maintain. The only difference is that L'_1, L'_2 and L'_T may contain distinct polynomials that have the same encoding. We define the following events for this generic bilinear group interaction.

$\boldsymbol{E'_{\mathcal{A}}}$ — $\mathcal{A}$ solves the q-SDH problem with $q'_G = k'_1 + k'_2 + k'_T - (q+3)$ oracle queries.

$\boldsymbol{E'_1}$ — There exist distinct polynomials $F'_{1,i}$ and $F'_{1,j}$ with $1 \le i, j \le k'_1$ such that $F'_{1,i}(x) = F'_{1,j}(x)$.

$\boldsymbol{E'_2}$ — There exist distinct polynomials $F'_{2,i}$ and $F'_{2,j}$ with $1 \le i, j \le k'_2$ such that $F'_{2,i}(x) = F'_{2,j}(x)$.

$\boldsymbol{E'_T}$ — There exist distinct polynomials $F'_{T,i}$ and $F'_{T,j}$ with $1 \le i, j \le k'_T$ such that $F'_{T,i}(x) = F'_{T,j}(x)$.

$\boldsymbol{E'_{\mathcal{B}}}$ — $E'_1 \cup E'_2 \cup E'_T$.

We are interested in the probability $\Pr[E'_{\mathcal{A}}]$ where $q'_G = q_G$. Note that, if $k'_1 = k_1$, $k'_2 = k_2$, and $k'_T = k_T$, then

$$\Pr[E'_{\mathcal{B}}] = \Pr[E_{\mathcal{B}}] \quad \text{and} \quad \Pr[E_{\mathcal{A}} \mid \overline{E_{\mathcal{B}}}] = \Pr[E'_{\mathcal{A}} \mid \overline{E'_{\mathcal{B}}}].$$

Hence,

$$\begin{aligned}
\Pr[E'_{\mathcal{A}}] &= \Pr[E'_{\mathcal{A}} \cap \overline{E'_{\mathcal{B}}}] + \Pr[E'_{\mathcal{A}} \cap E'_{\mathcal{B}}] \\
&\le \Pr[E_{\mathcal{A}} \cap \overline{E_{\mathcal{B}}}] + \Pr[E_{\mathcal{B}}] \\
&\le \Pr[E_{\mathcal{A}}] + \Pr[E_{\mathcal{B}}].
\end{aligned}$$

Thus, we can bound $\Pr[E'_{\mathcal{A}}]$ by bounding $\Pr[E_{\mathcal{A}}]$ and $\Pr[E_{\mathcal{B}}]$.

Note that the polynomials $F_{1,i}$, $F_{2,i}$, $F_{T,i}$, and $(X+c)F_{1,\ell}$ respectively have degree at most q, q, $2q$, and $q+1$. Also, notice that we have given $\mathcal{A}$ an SDH′ instance and query response without giving any information about our choice of x. Therefore,

$$\begin{aligned}
\Pr[E_{\mathcal{A}}] + \Pr[E_{\mathcal{B}}] &= \Pr[E_{\mathcal{A}}] + \Pr[E_1] + \Pr[E_2] + \Pr[E_T] \\
&\le \frac{q+1}{p-1} + \binom{k_1}{2}\frac{q}{p-1} + \binom{k_2}{2}\frac{q}{p-1} + \binom{k_T}{2}\frac{2q}{p-1} \\
&\le \left[\frac{q+1}{q} + \frac{k_1^2 + k_2^2 + 2k_T^2}{2}\right]\frac{q}{p-1} \\
&\le \frac{(k_1 + k_2 + k_T)^2 q}{p-1} \\
&= \frac{(q_G + q + 3)^2 q}{p-1}.
\end{aligned}$$

Asymptotically,

$$\epsilon = O\left(\frac{(q_G+q)^2 q}{p}\right) = O\left(\frac{q_G^2 q + q_G q^2 + q^3}{p}\right) = O\left(\frac{q_G^2 q + q^3}{p}\right),$$

where the last equality follows from the inequality of arithmetic and geometric means. □

2.3.3 Theorem. *[14, Corollary 13] Any algorithm that solves the q-SDH′ problem with constant probability $\epsilon > 0$ in a generic bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ of order p where $q = o(\sqrt[3]{p})$ requires $\Omega(\sqrt{\epsilon p/q})$ generic operations.*

Proof. Suppose there is a generic algorithm that solves the q-SDH′ problem with constant probability $\epsilon > 0$ as long as $q = o(\sqrt[3]{p})$. Then, by Theorem 2.3.2, there exist an integer N and a constant $C > 0$ such that

$$\epsilon \leq \frac{C(q_G^2 q + q^3)}{p} \quad \text{for all} \quad q_G, q, p \geq N \quad \text{with} \quad q = o(\sqrt[3]{p}).$$

This implies that

$$q_G^2 \geq \frac{\epsilon p}{Cq} - q^2 \quad \text{for all} \quad q_G, q, p \geq N \quad \text{with} \quad q = o(\sqrt[3]{p}).$$

Now, for $q = o(\sqrt[3]{p})$, we have that $q^3 = o(p)$ and so $q^2 = o(p/q)$. Therefore,

$$q_G^2 = \Omega\left(\frac{p}{q} - q^2\right) = \Omega\left(\frac{p}{q}\right),$$

and so $q_G = \Omega(\sqrt{p/q})$, as desired. □

2.3.1 Random Oracles and Generic Groups

Both the random oracle model [9] and the generic (bilinear) group model represent formalizations of an ideal cryptographic environment, which is in fact not realizable in practice. Naturally, this renders controversial the use of such non-standard models [19, 36, 31, 27, 24].

As we saw in Theorem 2.3.3, Boneh and Boyen use the generic bilinear group assumption to prove a lower bound on the SDH (resp. SDH′) problem on which the security of Boneh-Boyen signatures is based [13] (resp. [14]). Boyen [19], in acknowledging that SDH was viewed as a non-standard assumption when first introduced in [13], states that "without the support of a bilinear generic-group analysis, the authors of (the reference [13, 14]) would have never dared make such an unusual assumption."

Nonetheless, Boyen recognizes the pitfalls accompanying the use of the generic group model. He warns that "the very act of placing oneself in the generic group

model amounts to making an extremely strong assumption" and even says that a security proof in the generic group model "brings little insight as to its real-world security." Comparing the random oracle model and the generic group model, he notes the following:

> After all, random oracles are used to model the destruction of any exploitable structure by "bit-mashing" hash contraptions that are designed with this particular goal as the primary objective: make the number of rounds large enough and you are almost guaranteed to fulfill that requirement, whatever the design of the round function. Generic groups, on the other hand, formalize the belief that the particular presentation of a group will expose some aspects of its structure while hiding others: one's choice of mathematical implement is thus much more constrained, as it has to fulfill the two conflicting goals of structure removal and structure preservation. Once we have chosen a particular type of presentation for our bilinear groups (e.g., supersingular elliptic curves over prime finite fields), the structure-hiding properties are mostly bound by this choice, with little room for subsequent adjustment.

In short, the above says that the primary objective of the design of good hash functions is to make them behave as close to random oracles as possible, whereas in implementing actual groups, making them resemble the generic group is certainly not the priority, and also that the extent to which we can remove the structure of groups is limited. Boyen's concludes his discussion [19] by saying:

> The generic-group model should thus be viewed as a meta-assumption, useful not for proving the security of actual schemes, but to assess the plausibility of specific, weaker assumptions on which actual schemes are shown to rest.

Koblitz and Menezes [36] make a similar point. They state that although both the random oracle model and the generic group model are idealizations of real objects, "it is reasonable to think of a well-constructed real-world hash function as a deterministic function that is essentially indistinguishable from a random function." They argue that, if we give the same sequence of inputs to a random oracle and a good hash function, then the outputs of a random oracle and a good hash function cannot be distinguished with probability significantly greater than 50%. On the other hand, they say that "the generic group model is not a literal description of any of the groups that might be used in real-world cryptography." They give several examples that show how encodings of actual groups can be easily distinguished from random strings without even consulting a group operation, some of which we describe below:

- The identity element of a group $\mathbb{G}$ is often easy to identify. If $\mathbb{G}$ is a multiplicative group of a finite field, then the identity element is 1. If $\mathbb{G}$ is an *elliptic curve* expressed in *projective coordinates*, the identity element is $(0, 1, 0)$.

- In an elliptic curve $y^2 = f(x)$, a point and its inverse have the same x-coordinate.
- The x-coordinate of any point on an elliptic curve $y^2 = f(x)$ over $\mathbb{F}_p$ evaluated at f is a quadratic residue modulo p.
- In DSA, we use a multiplicative subgroup $\mathbb{G}$ of $\mathbb{Z}_p^*$ of prime order q. If $\mathbb{Z}_p^*$ is generated by g, then $\mathbb{G}$ is generated by g^k where $p - 1 = qk$. Since such k must be even, all the elements of $\mathbb{G}$ are quadratic residues modulo p.

The primary motivation for the Boneh-Boyen signature scheme, as evidenced by the titles of [13, 14], was to achieve security without random oracles while retaining signature lengths as short as BLS signatures (Section 3.2). However, it is clear from the above discussion that the relevance of the generic group security of the SDH problem is at least as debatable as that of the random oracle model. In any case, our aim here is only to raise awareness of the issue, without taking a position for one side or the other.

Chapter 3

Boneh-Boyen Signatures and Related Schemes

So far we have encountered several DH and SDH related intractability assumptions that form the basis for proving the security of cryptographic schemes. Here we review the Boneh-Boyen signature scheme and three other schemes — the BLS, Waters, and Okamoto signature schemes — that are provably secure under these assumptions. All of the above schemes are relatively new, with the oldest being BLS which first appeared in 2001 [17]. They also all implicitly or explicitly make use of bilinear pairings. Interestingly, all three schemes except for the Okamoto scheme are obtained via a common generic conversion method from *identity-based encryption (IBE)* schemes to signature schemes. We describe IBE and the generic conversion method in Section 3.5.

3.1 Boneh-Boyen Signatures

We begin with Boneh-Boyen signatures [13, 14]. Boneh and Boyen present two variants of their signature scheme in each of [13, 14]: the basic signature scheme and the full signature scheme. The former is deterministic and weakly secure, whereas the latter is randomized and strongly secure. The basic scheme is used to prove the security of the full scheme in both papers.

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order p, and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing. The differences between the signatures defined in the two papers are as follows:

- The schemes in [13] assume the existence of an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, and the generator $g_1 \in \mathbb{G}_1$ is defined to be $\psi(g_2)$. On the other hand, in [14] both $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are taken to be random generators.

- The message space of the schemes in [13] is $\mathbb{Z}_p^*$, whereas the message space is $\mathbb{Z}_p$ in [14].
- In the full signature scheme in [13], the randomizer r is chosen from $\mathbb{Z}_p^*$. On the other hand, in the full signature scheme given in [14], the randomizer r is randomly chosen from $\mathbb{Z}_p$.
- In [13], the generator $g_1 \in \mathbb{G}_1$ is part of the public key. In [14], the generator $g_1 \in \mathbb{G}_1$ is part of the secret key, and need not be made public in practice.
- The security of the signature schemes in [13] relies on the q-SDH assumption, and that of the schemes in [14] relies on the q-SDH$'$ assumption. Recall that the SDH and SDH$'$ assumptions are similar but slightly different (Section 2.2.1).

3.1.1 Boneh-Boyen Signatures: Version 1

Following chronological order, we begin with the version of Boneh-Boyen signatures from Eurocrypt 2004 [13].

The Basic Signature Scheme.

System parameters: Cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p, an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$, and an efficiently computable bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Key generation: Choose a random generator $g_2 \in \mathbb{G}_2$, and set $g_1 \leftarrow \psi(g_2)$. Choose a random integer $x \in \mathbb{Z}_p^*$, and compute $v \leftarrow g_2^x$. Let $\zeta \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is $\mathsf{PK} = (g_1, g_2, v, \zeta)$, and the private key is $\mathsf{SK} = x$.

Signing: Given a message $m \in \mathbb{Z}_p^*$ and a private key $\mathsf{SK} = x$, output the signature $\sigma \leftarrow g_1^{\frac{1}{x+m}}$ where the exponent is calculated modulo p. In the unlikely event that $x + m \equiv 0 \pmod{p}$, $\mathit{Sign}(m, \mathsf{SK})$ outputs $\sigma \leftarrow 1$.

Verification: $\mathit{Verify}(m, \sigma, (g_1, g_2, v, \zeta)) = \texttt{true}$ if and only if

$$e(\sigma,\ v \cdot g_2^m) = \zeta.$$

The Full Signature Scheme.

System parameters: Cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p, an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$, and an efficiently computable bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Key generation: Choose a random generator $g_2 \in \mathbb{G}_2$, and set $g_1 \leftarrow \psi(g_2)$. Choose random integers $x, y \in \mathbb{Z}_p^*$, and compute $u \leftarrow g_2^x$ and $v \leftarrow g_2^y$. Let $\zeta \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is $\mathsf{PK} = (g_1, g_2, u, v, \zeta)$, and the private key is $\mathsf{SK} = (x, y)$.

Signing: Given a message $m \in \mathbb{Z}_p^*$ and a private key $\mathsf{SK} = (x, y)$, choose a random $r \in \mathbb{Z}_p^*$ such that $x + m + yr \not\equiv 0 \pmod{p}$, and compute $\sigma \leftarrow g_1^{\frac{1}{x+m+yr}}$ where the exponent is calculated modulo p. The signature is (σ, r).

Verification: *Verify*$(m, (\sigma, r), (g_1, g_2, u, v, \zeta)) =$ `true` if and only if

$$e(\sigma,\ u \cdot g_2^m \cdot v^r) = \zeta.$$

We remark that the message space as stated above is $\mathbb{Z}_p^*$, but it can be extended to $\{0,1\}^*$ using a collision resistant hash function $H\colon \{0,1\}^* \to \mathbb{Z}_p^*$. We have the following theorem regarding the security of the full signature scheme in [13].

3.1.1 Theorem. *[13, Theorem 3.1] If the (q, t', ϵ)-SDH assumption holds in the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, then the full version of the Boneh-Boyen signature scheme in [13] is (t, q_S, ϵ)-strongly existentially unforgeable under an adaptive chosen message attack provided that*

$$q_S < q, \quad \epsilon \geq 2\epsilon', \quad \text{and} \quad t \leq t' - \Theta(q^2 T)$$

where T is the maximum time required for one exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{Z}_p$.

The proof is similar to that of Theorem 4.2.1, and is omitted here. The bound for the probability ϵ is different from $\epsilon \geq 2(\epsilon' + q_S/p)$ as given in [13, Theorem 3.1]. This is because the same technique used in the proof of Lemma 4.2.2 to improve the probability bound there also applies here.

3.1.2 Boneh-Boyen Signatures: Version 2

Next, we describe the version of the Boneh-Boyen signature scheme from the Journal of Cryptology [14]. This is the version that we are going to focus on in the subsequent chapters.

The Basic Signature Scheme.

System parameters: Cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p, and a bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Key generation: Choose random generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and a random integer $x \in \mathbb{Z}_p^*$. Compute $v \leftarrow g_2^x$ and $\zeta \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is $\mathsf{PK} = (g_1, g_2, v, \zeta)$, and the private key is $\mathsf{SK} = (g_1, x)$.

Signing: Given a message $m \in \mathbb{Z}_p$ and a private key $\mathsf{SK} = (g_1, x)$, output the signature $\sigma \leftarrow g_1^{\frac{1}{x+m}}$ where the exponent is calculated modulo p. In the unlikely event that $x + m \equiv 0 \pmod{p}$, output $\sigma \leftarrow 1$.

Verification: *Verify*$(m, \sigma, (g_1, g_2, v, \zeta)) = \texttt{true}$ if and only if

$$e(\sigma,\ v \cdot g_2^m) = \zeta.$$

The Full Signature Scheme.

System parameters: Cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p, and a bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Key generation: Choose random generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and random integers $x, y \in \mathbb{Z}_p^*$. Compute $u \leftarrow g_2^x$, $v \leftarrow g_2^y$, and $\zeta \leftarrow e(g_1, g_2) \in \mathbb{G}_T$. The public key is $\mathsf{PK} = (g_1, g_2, u, v, \zeta)$, and the private key is $\mathsf{SK} = (g_1, x, y)$.

Signing: Given a message $m \in \mathbb{Z}_p$ and a private key SK, choose a random $r \in \mathbb{Z}_p$ such that $x + m + yr \not\equiv 0 \pmod{p}$, and compute $\sigma \leftarrow g_1^{\frac{1}{x+m+yr}}$ where the exponent is calculated modulo p. The signature is (σ, r).

Verification: *Verify*$(m, (\sigma, r), (g_1, g_2, u, v, \zeta)) = \texttt{true}$ if and only if

$$e(\sigma,\ u \cdot g_2^m \cdot v^r) = \zeta.$$

We remark that the message space as stated above is $\mathbb{Z}_p$, but it can be extended to $\{0,1\}^*$ using a collision resistant hash function $H\colon \{0,1\}^* \to \mathbb{Z}_p$. We also note that in the descriptions above, g_1 is present in both the public key and the private key. In practice, g_1 can be omitted from the public key. The security proofs for these signature schemes are given in Chapter 4.

3.2 BLS Signatures

The BLS signature scheme of Boneh, Lynn, and Shacham [17, 18] was the first short signature scheme to appear in the literature. The version that is first introduced [17] is shown to be secure in a bilinear group pair $(\mathbb{G}, \mathbb{G})$ if CDH is hard and DDH is easy in $\mathbb{G}$. The full version of the paper [18] generalizes the scheme to allow the use of Type 2 pairings as well as Type 1 pairings. Accordingly, the underlying assumption has also been generalized using the co-CDH and co-DDH problems.

3.2.1 Definition. A pair of groups $(\mathbb{G}_1, \mathbb{G}_2)$ of prime order p is called a *gap co-Diffie-Hellman (co-GDH) pair* if it satisfies the following properties:

1. The group operations in $\mathbb{G}_1$ and $\mathbb{G}_2$ can be computed efficiently.

2. There exists an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$.

3. The co-DDH problem on $(\mathbb{G}_1, \mathbb{G}_2)$ can be solved efficiently.

4. The co-CDH problem on $(\mathbb{G}_1, \mathbb{G}_2)$ is intractable.

Note that the definitions of co-CDH, co-DDH, and co-GDH pair do not require any pairings. In practice, however, the only group pairs that are believed to be co-GDH are bilinear group pairs. Thus we may safely call BLS a pairing based signature scheme. Nevertheless, for the sake of generality, we present the version from [18], which is based on co-GDH pairs.

System parameters: A co-Gap Diffie-Hellman group pair $(\mathbb{G}_1, \mathbb{G}_2)$, a generator $g_2 \in \mathbb{G}_2$, a full-domain hash function $H\colon \{0,1\}^* \to \mathbb{G}_1$.

Key generation: Choose random $x \in \mathbb{Z}_p$, $g_2 \in \mathbb{G}_2$, and compute $v \leftarrow g_2^x$. The public key is $\mathsf{PK} = v$, and the private key is $\mathsf{SK} = x$.

Signing: Given a message $m \in \{0,1\}^*$ and a private key $\mathsf{SK} = x$, output the signature $\sigma \leftarrow H(M)^x$.

Verification: *Verify*$(m, \sigma, v) =$ `true` if and only if $(g_2, v, H(m), \sigma)$ is a co-Diffie-Hellman tuple.

The following theorem establishes the security of the BLS signature scheme. We refer the reader to [18] for a proof.

3.2.2 Theorem. *If $(\mathbb{G}_1, \mathbb{G}_2)$ is a co-GDH group pair, then the BLS signature scheme above is existentially unforgeable under an adaptive chosen-message attack in the random oracle model.*

3.3 Waters Signature Scheme

In the paper "Efficient identity-based encryption without random oracles" [56], Waters introduces a signature scheme obtained by applying a generic conversion (see Section 3.5) to his IBE scheme. Although the scheme is not discussed in detail in the paper, it turns out that Waters signatures are short and secure under the standard model, with the only underlying assumption being CDH. The scheme is as follows.

System parameters: Cyclic group $\mathbb{G}$ of prime order p, efficiently computable bilinear pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Key generation: Choose a generator $g \in \mathbb{G}$ and $x \in \mathbb{Z}_p$, and set $u \leftarrow g^x$. Choose random elements v and w from $\mathbb{G}$. Additionally, choose a random vector $W = (w_1, \ldots, w_n) \in \mathbb{G}^n$ of length n. The public key is $\mathsf{PK} = (g, u, v, w, W)$, and the private key is $\mathsf{SK} = v^x$.

Signing: Given a message $M = \{m_1, \ldots, m_n\} \in \{0,1\}^n$ and a private key $\mathsf{SK} = v^x$, choose a random $r \in \mathbb{Z}_p$ and calculate $\sigma_1 \leftarrow v^x \left(w \prod_{i=1}^n w_i^{m_i}\right)^r$ and $\sigma_2 \leftarrow g^r$. The signature is $\sigma \leftarrow (\sigma_1, \sigma_2)$.

Verification: *Verify*$(M, (\sigma_1, \sigma_2), (g, u, v, w, W)) = \texttt{true}$ if and only if

$$e(u,v) \cdot e\Big(\sigma_2, w \prod_{i=1}^n w_i^{m_i}\Big) = e(g, \sigma_1).$$

Note that if $\sigma = (\sigma_1, \sigma_2)$ is a valid signature of message $M = \{m_1, \ldots, m_n\}$ under $\mathsf{SK} = v^x$, then

$$\begin{aligned} e(u,v) \cdot e\Big(\sigma_2, w \prod_{i=1}^n w_i^{m_i}\Big) &= e(g^x, v) \cdot e\Big(g^r, w \prod_{i=1}^n w_i^{m_i}\Big) \\ &= e(g, v^x) \cdot e\Big(g, \Big(w \prod_{i=1}^n w_i^{m_i}\Big)^r\Big) \\ &= e(g, \sigma_1), \end{aligned}$$

so that the scheme is consistent. We have the following security theorem:

3.3.1 Theorem. *If the CDH assumption holds in $\mathbb{G}$, then the Waters signature scheme above is existentially unforgeable under an adaptive chosen-message attack in the standard model.*

The proof of the above theorem is similar to the security proof of the Waters IBE scheme [56]. We remark that the message space as stated above is $\{0,1\}^n$, but it can be extended to $\{0,1\}^*$ using a collision resistant hash function $H\colon \{0,1\}^* \to \{0,1\}^n$ without loss of security.

3.4 Okamoto Signature Scheme

Okamoto [45] proposes a signature scheme as a building block for a pair of signature schemes, one blind and one partially blind, which are presented within the same paper. The security of the signatures depends on a new assumption called the 2SDH assumption which we presented in Section 2.2.4. 2SDH is stronger than SDH, but Okamoto claims that his scheme is more suitable than the Boneh-Boyen or Waters signature schemes for many applications such as blind signatures, credentials, and group signatures. The scheme is as follows.

System parameters: Cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, of prime order p, an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$, and an efficiently computable bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Key generation: Choose random generators $g_2, u_2, v_2 \in \mathbb{G}_2$, and set $g_1 \leftarrow \psi(g_2)$. Choose $x \in \mathbb{Z}_p^*$, and set $w_2 \leftarrow g_2^x$. The public key is $\mathsf{PK} = (g_1, g_2, w_2, u_2, v_2)$, and the private key is $\mathsf{SK} = x$.

Signing: Let $u_1 \leftarrow \psi(u_2)$ and $v_1 \leftarrow \psi(v_2)$. Given a message $m \in \mathbb{Z}_p^*$ and a private key $\mathsf{SK} = x$, choose random $r, s \in \mathbb{Z}_p^*$ such that $x + r \not\equiv 0 \pmod{p}$, and calculate $\sigma \leftarrow (g_1^m u_1 v_1^s)^{\frac{1}{x+r}}$. The signature is (σ, r, s).

Verification: *Verify*$(m, (\sigma, r, s), (g_1, g_2, w_2, u_2, v_2)) = \texttt{true}$ if and only if

$$e(\sigma, w_2 g_2^r) = e(g_1, g_2^m u_2 v_2^s).$$

Suppose (σ, r, s) is a valid signature of message m under $\mathsf{SK} = x$, and $\mathsf{PK} = (g_1, g_2, w_2, u_2, v_2)$. Suppose further that $u_2 = g_2^a$ and $v_2 = g_2^b$ for some $a, b \in \mathbb{Z}_p$. Then,

$$\begin{aligned} e(\sigma, w_2 g_2^r) &= e((g_1^m u_1 v_1^s)^{\frac{1}{x+r}}, g_2^{x+r}) \\ &= e(g_1^m u_1 v_1^s, g_2) \\ &= e(g_1^{m+a+bs}, g_2) \\ &= e(g_1, g_2^{m+a+bs}) \\ &= e(g_1, g_2^m u_2 v_2^s), \end{aligned}$$

so that the scheme is consistent. The security theorem is as follows.

3.4.1 Theorem. *If the 2SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the Okamoto signature scheme above is existentially unforgeable under an adaptive chosen-message attack in the standard model.*

We refer the reader to [45] for a proof. We remark that the message space as stated above is $\mathbb{Z}_p^*$, but it can be extended to $\{0,1\}^*$ using a collision resistant hash function $H\colon \{0,1\}^* \to \mathbb{Z}_p^*$.

3.5 Identity-Based Encryption and Relationship to Signature Schemes

We have mentioned on several occasions that Boneh-Boyen, BLS, and Waters signatures are obtained via a generic transformation from identity-based encryption (IBE) schemes. In this section we recall the definition of an IBE scheme, the generic transformation to signature schemes [15], and the Boneh-Boyen IBE scheme [12] from which Boneh-Boyen signatures were derived.

One of the important issues in public key cryptography is certification of public keys. When Alice sends a secret message to Bob using his public key, she wants to make sure that her copy of Bob's key really belongs to Bob and not Carol. As

obvious as it may sound, the issue of obtaining an authentic copy of a key is not trivial. One way to solve this problem is to establish a public key infrastructure (PKI) that manages keys and issues certificates of keys. However, widespread deployment of PKIs lead to many difficulties in practice [55].

Shamir introduced the notion of identity-based encryption in 1985 [51]. An IBE scheme can alleviate the problem of running a complicated PKI. In an IBE scheme, anyone can compute Bob's public key using only public information such as Bob's email address. Thus Bob does not have to ask a certification authority (CA) to issue him a certificate. Instead, we need a trusted party called a private key generator (PKG) that issues private keys corresponding to identity strings. When Bob receives an encrypted message, he authenticates himself to the PKG and obtains the private key corresponding to his public key (this procedure only needs to be done once). Note that the private key is generated using a master secret key that the PKG possesses, so only the PKG can compute the private key corresponding to an identity string. The formal definition of an IBE scheme is as follows.

3.5.1 Definition. An identity-based encryption (IBE) scheme consists of a quadruple of algorithms (*SetUp*, *Extract*, *Encrypt*, *Decrypt*) satisfying the following properties.

- *SetUp* takes some security parameter ℓ as an input, and outputs a random key pair $(\mathsf{param}, \mathsf{master\text{-}key})$, where param denotes the public parameters, and $\mathsf{master\text{-}key}$ denotes a master key. The PKG publishes param and keeps $\mathsf{master\text{-}key}$ secret.
- *Extract* takes param, $\mathsf{master\text{-}key}$, $\mathsf{PK_{ID}}$ as input, where $\mathsf{PK_{ID}}$ may be any binary string, and outputs a secret key $\mathsf{SK_{ID}}$ corresponding to the public key $\mathsf{PK_{ID}}$.
- *Encrypt* receives the public parameters param, a public key $\mathsf{PK_{ID}}$, and a message M as input, and outputs a ciphertext C.
- *Decrypt* takes param, $\mathsf{PK_{ID}}$, $\mathsf{SK_{ID}}$, and a ciphertext C as input, outputs a message M.

We say that an IBE scheme is consistent if, whenever the secret key $\mathsf{SK_{ID}}$ corresponding to the ID $\mathsf{PK_{ID}}$ is properly generated by the algorithm *Extract* with parameters param and a master key $\mathsf{master\text{-}key}$ generated by *SetUp*, then,

$$\forall M \in \mathcal{M}, \quad \mathit{Decrypt}(\mathsf{param}, \mathsf{PK_{ID}}, \mathsf{SK_{ID}}, C) = M$$

where $C = \mathit{Encrypt}(\mathsf{param}, \mathsf{PK_{ID}}, M)$.

A generic method to convert any IBE scheme to a signature scheme was given by Boneh and Franklin [15], and attributed to Moni Naor. Suppose *SetUp*, *Extract*, *Encrypt*, and *Decrypt* are the four algorithms comprising an IBE scheme. We can construct a signature scheme with algorithms *KeyGen*, *Sign*, and *Verify* as follows:

- *KeyGen* takes some security parameter ℓ as an input, and outputs a random key pair $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathit{SetUp}(\ell)$. That is, the public parameters in the IBE scheme comprise the public key of the signature scheme, and the corresponding master key becomes a secret key.

- *Sign* takes a message m and a secret key SK as input, and outputs a signature $\sigma \leftarrow \mathit{Extract}(\mathsf{PK}, \mathsf{SK}, m)$. That is, σ is the secret key of the IBE scheme corresponding to the identity m.

- *Verify*, upon receiving a message m, a signature σ, and a public key PK as input, generates a random message M' for the IBE scheme. Next, using PK as the public parameters and m as the identity, it encrypts M' and lets $C' \leftarrow \mathit{Encrypt}(\mathsf{PK}, m, M')$. Finally, it computes the decryption of C' under the IBE scheme using σ as the secret key. It outputs `true` if

$$\mathit{Decrypt}(\mathsf{PK}, m, \sigma, C') = M',$$

  and `false` otherwise.

We describe below the Boneh-Boyen IBE scheme from which the Boneh-Boyen signature scheme is derived.

Boneh-Boyen IBE.

System parameters: Cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with an efficiently computable bilinear pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Here g is a generator of $\mathbb{G}$.

Set up: Choose random $x, y \in \mathbb{Z}_p^*$, and set $u \leftarrow g^x$ and $v \leftarrow g^y$. Calculate $\zeta \leftarrow e(g, g)$. The public parameters are $\mathsf{param} = (g, u, v)$, and the master secret is $\mathsf{master\text{-}key} = (x, y)$.

Secret key extraction: Given a master key $\mathsf{master\text{-}key}$ and an identity $\mathsf{PK_{ID}} \in \mathbb{Z}_p^*$, choose a random $r \in \mathbb{Z}_p$ such that $x + \mathsf{PK_{ID}} + yr \not\equiv \pmod p$. Set $K \leftarrow g^{\frac{1}{x+\mathsf{PK_{ID}}+yr}}$. The secret key corresponding to $\mathsf{PK_{ID}}$ is $\mathsf{SK_{ID}} = (r, K)$.

Encryption: Given a message $M \in \mathbb{G}_T$, a public parameter param, and the identity $\mathsf{PK_{ID}}$, choose a random $s \in \mathbb{Z}_p^*$. Calculate $a \leftarrow u^s g^{s\mathsf{PK_{ID}}}$, $b \leftarrow v^s$, and $c \leftarrow \zeta^s M$. The ciphertext is $C = (a, b, c)$.

Decryption: Given a secret key $\mathsf{SK_{ID}} = (r, K)$ and a ciphertext $C = (a, b, c)$, output the message $M = \frac{c}{e(ab^r, K)}$. Note that if C is a valid ciphertext of M, then we have

$$\frac{c}{e(ab^r, K)} = \frac{c}{e(g^{s(x+\mathsf{PK_{ID}}+yr)}, g^{\frac{1}{x+\mathsf{PK_{ID}}+yr}})} = \frac{c}{e(g, g)^s} = M.$$

It is straightforward to verify that when the generic transformation is applied to Boneh-Boyen IBE, then the resulting signature scheme is equivalent to the Boneh-Boyen full signature scheme that is defined in the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ where $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$. Also, under this transformation, the Boneh-Franklin IBE scheme [15] yields the BLS signature scheme (Section 3.2), and the Waters IBE scheme [56] yields the Waters signature scheme (Section 3.3).

Chapter 4

Reductions Between Boneh-Boyen Signatures and the SDH′ Problem

Having reviewed the definitions of signature schemes and intractability assumptions, we are now ready to examine reductions between Boneh-Boyen signatures and the SDH (SDH′) problem. As we have seen in Section 3.1, Boneh and Boyen published two versions of their paper [13, 14], which in total contain four different versions of the signature scheme — a basic scheme and a full scheme in each paper. The differences between the schemes in [13] and [14] are not significant as far as security proofs are concerned. Thus, in this chapter, we focus on the schemes from the full version of the paper [14]. We show that all four versions can be reduced to the SDH′ problem; that is, given an SDH′ oracle, it is possible to forge Boneh-Boyen signatures. This result is the converse of the security proof given by Boneh and Boyen, and shows that forging Boneh-Boyen signatures is equivalent to the SDH′ problem. We remark that our reductions from Boneh-Boyen signatures to SDH′ do not always extend to the original formulation of the SDH problem [13]. This issue is briefly discussed in Section 4.6.

4.1 Reduction from SDH′ to the Basic Signature Scheme

In this section and the next section, we give the security proofs for the two versions of the Boneh-Boyen signature scheme contained in the full version of their paper [14]. These proofs are identical to the ones in [14], except in one instance where we improve a probability bound. We start with the basic version of the signature scheme.

4.1.1 Theorem. *[14, Lemma 9] If the (q, t', ϵ)-SDH′ assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the basic Boneh-Boyen signature scheme presented in Section 3.1.2 is (t, q_S, ϵ)-existentially unforgeable under a weak chosen message attack provided that*

$$q_S \leq q, \text{ and } t \leq t' - \Theta(q_S^2 T)$$

where T is the maximum time needed for one exponentiation in $\mathbb{G}_1$ and $\mathbb{Z}_p$.

Proof. The following proof is taken from [14, Lemma 9].

Let $\mathcal{A}$ be a forger that (t, q_S, ϵ)-weakly breaks the basic Boneh-Boyen signature scheme in Section 3.1.2, and suppose $q \geq q_S$. We construct an algorithm $\mathcal{B}$ that $(t + \Theta(q^2 T), \epsilon)$-breaks the q-SDH$'$ problem with the aid of $\mathcal{A}$.

Set up. Algorithm $\mathcal{B}$ is given a random instance $(h_0, h_1, \ldots, h_q, g_2, v) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ of the q-SDH problem, where $h_i = g^{x^i}$ for $i = 0, 1, \ldots, q$, and $v = g_2^x$. The objective of the algorithm $\mathcal{B}$ is to output $(c, g_1^{\frac{1}{x+c}})$ for some $c \in \mathbb{Z}_p$.

Query and key generation. The algorithm $\mathcal{B}$ invokes $\mathcal{A}$. Since $\mathcal{A}$ is a weak forger, it submits q_S messages $m_1, \ldots, m_{q_S} \in \mathbb{Z}_p$ for signature queries to $\mathcal{B}$ before it receives a public key.

Algorithm $\mathcal{B}$ must simulate the *KeyGen* algorithm of the basic signature scheme and generate a valid public key for $\mathcal{A}$. Using the messages $m_1, \ldots, m_{q_S}$, the algorithm $\mathcal{B}$ defines a polynomial

$$f(X) \leftarrow \prod_{i=1}^{q_S} (X + m_i).$$

It then expands $f(X)$ to obtain the coefficients $\alpha_0, \ldots, \alpha_{q_S} \in \mathbb{Z}_p$ such that $f(X) = \sum_{i=0}^{q_S} \alpha_i X_i$. Now, the algorithm $\mathcal{B}$ picks $\rho \in \mathbb{Z}_p^*$ randomly, and calculatates

$$g_1' \leftarrow \left(\prod_{i=0}^{q_S} h_i^{\alpha_i} \right)^{\rho} = g_1^{\rho f(x)} \quad \text{and} \quad \zeta \leftarrow e(g_1', g_2).$$

If g_1' happens to be the identity element of $\mathbb{G}_1$, then it means $f(x) = 0$, and we have $x + m_i = 0$ for one of $i \in \{1, \ldots, q_S\}$. In this case, $\mathcal{B}$ can recover the secret x and easily solve the given q-SDH$'$ problem. Thus, we assume that g_1' is a generator of $\mathbb{G}_1$ and also that $f(x) \neq 0$. The algorithm $\mathcal{B}$ gives $\mathsf{PK} \leftarrow (g_1', g_2, v, \zeta)$ to $\mathcal{A}$ as a public key of the basic signature scheme. Note that the randomizer ρ makes g_1' a random generator of $\mathbb{G}_1$, so that $\mathcal{A}$ cannot distinguish PK from a public key generated by the *KeyGen* algorithm.

Query response. Next, $\mathcal{B}$ simulates the *Sign* algorithm of the basic signature scheme and produce signatures of the given messages $m_1, \ldots, m_{q_S} \in \mathbb{Z}_p$. To do so, $\mathcal{B}$ defines the polynomials

$$f_i(X) \leftarrow \prod_{j=1, j \neq i}^{q_S} (X + m_j) = \frac{f(X)}{X + m_i}$$

for $i = 1, \ldots, q_S$, and expands them to get the coefficients $\beta_{i,0}, \ldots, \beta_{i,q_S}$ such that $f_i(X) = \sum_{j=0}^{q_S-1} \beta_{i,j} X^j$. It then calculates

$$\sigma_i \leftarrow \left(\prod_{j=0}^{q_S-1} h_j^{\beta_{i,j}} \right)^{\rho} = {g'_1}^{\frac{1}{x+m_i}}$$

for each $i = 1, \ldots, q_S$, and gives $(\sigma_1, \ldots, \sigma_{q_S})$ to $\mathcal{A}$ as a response to the signature query $\mathcal{A}$ issued.

Signature output. Eventually, $\mathcal{A}$ outputs $(m_*, \sigma_*) \in \mathbb{Z}_p \times \mathbb{G}_1$ as a forgery of the basic signature scheme. If (m_*, σ_*) is not a valid forgery, then $\mathcal{B}$ aborts. Otherwise, we have that m_* is distinct from all of $m_1, \ldots, m_{q_S}$, and that $e(g'_1, g_2) = e(\sigma_*, v g_2^{m_*})$. That is,

$$\sigma_* = {g'_1}^{\frac{1}{x+m_*}} = g_1^{\frac{\rho f(x)}{x+m_*}}.$$

SDH output. Finally, $\mathcal{B}$ derives a valid SDH output from (m_*, σ_*). Since $m_* \neq m_i$ for $i = 1, \ldots, q_S$, the algorithm $\mathcal{B}$ can find a polynomial quotient $s(X)$ and a nonzero remainder $t \in \mathbb{Z}_p^*$ such that $f(X) = (X + m_*)s(X) + t$. Therefore, we have

$$\sigma_* = g_1^{\rho s(x)} g_1^{\frac{\rho t}{x+m_*}}.$$

The algorithm $\mathcal{B}$ expands $s(X)$ and obtains the coefficients $\gamma_0, \ldots, \gamma_{q_S-1}$ such that $s(X) = \sum_{i=0}^{q_S-1} \gamma_i X^i$. It calculates

$$M_* \leftarrow \left(\sigma_*^{1/\rho} \prod_{i=0}^{q_S-1} h_i^{-\gamma_i} \right)^{1/t} = g_1^{\frac{1}{x+m_*}},$$

and outputs (m_*, M_*) as a solution to the q-SDH problem. It is clear that $\mathcal{B}$ succeeds when $\mathcal{A}$ successfully forges the basic signature scheme, which happens with probability ϵ. The running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$ and the calculations of $\sigma_1, \ldots, \sigma_{q_S}$, which take $t + \Theta(q_S^2 T)$ time in total. □

4.2 Reduction from SDH′ to the Full Signature Scheme

In this section, we prove that the SDH′ problem can be reduced to an existential forgery of the Boneh-Boyen full signature scheme. Specifically, we prove the following.

4.2.1 Theorem. *[14, Theorem 8] If the (q, t', ϵ)-SDH′ assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then the full Boneh-Boyen signature scheme presented in Section 3.1.2 is (t, q_S, ϵ)-strongly existentially unforgeable under an adaptive chosen message attack provided that*

$$q_S \leq q, \quad \epsilon \geq 2\epsilon', \quad \text{and} \quad t \leq t' - \Theta(q^2 T)$$

where T is the maximum time needed for one exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{Z}_p$.

Proof. This follows immediately from Theorem 4.1.1 and Lemma 4.2.2 below. □

We note that the bound $\epsilon \geq 2\epsilon'$ for the probability above is slightly better than the bound $\epsilon \geq 2\epsilon' + q_S/p$ given in [14, Theorem 8]. This is due to a slight modification in the proof of Lemma 4.2.2.

4.2.2 Lemma. *[14, Lemma 10] If the basic Boneh-Boyen signature scheme presented in Section 3.1.2 is (t', q_S, ϵ')-existentially unforgeable under a weak chosen message attack, then the full signature scheme in Section 3.1.2 is (t, q_S, ϵ)-strongly existentially unforgeable under an adaptive chosen message attack provided that*

$$\epsilon \geq 2\epsilon' \quad \text{and} \quad t \leq t' - \Theta(q_S T)$$

where T is the maximum time required for one exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{Z}_p$.

Proof. This proof is identical to [14, Lemma 10] except for a small modification which yields a slightly better probability analysis.

Let $\mathcal{A}$ be a forger that (t, q_S, ϵ)-breaks the full Boneh-Boyen signature scheme. We construct an algorithm $\mathcal{B}$ that $(t + \Theta(q_S T), q_S, \epsilon/2)$-weakly breaks the basic signature scheme.

We distinguish two types of forgers that $\mathcal{A}$ can simulate depending on the signature queries it makes and the forgery it outputs at the end. Suppose $\mathcal{A}$ is given a public key $\mathsf{PK}_{\mathcal{A}} = (g_1, g_2, U \leftarrow g_2^x, V \leftarrow g_2^y, \zeta)$, makes signature queries on messages $m_1, \ldots, m_{q_S}$, and receives $\sigma_1^{(\mathcal{A})}, \ldots, \sigma_{q_S}^{(\mathcal{A})}$ as a response to the queries. Suppose also that $\mathcal{A}$ outputs a valid forgery $(m_*, (\sigma_*, r_*))$ at the end. We say that $\mathcal{A}$ is a:

Type 1 forger if either

1. $m_i = -x$ for some $i \in \{1, \ldots, q_S\}$, or
2. $\sigma_* \neq \sigma_i^{(\mathcal{A})}$ for all $i \in \{1, \ldots, q_S\}$; and

Type 2 forger if both

1. $m_i \neq x$ for all $i \in \{1, \ldots, q_S\}$, and
2. $\sigma_* = \sigma_i^{(\mathcal{A})}$ for some $i \in \{1, \ldots, q_S\}$.

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ denote Type 1 and Type 2 forgers of the full signature scheme, respectively. We show how to construct algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ that successfully forge the basic signature scheme using $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. Since $\mathcal{B}$ does not know the type of forger $\mathcal{A}$ simulates, it flips a fair coin, and decides whether it simulates $\mathcal{B}_1$ or $\mathcal{B}_2$. We describe below how the forgery is performed when the types of $\mathcal{A}$ and $\mathcal{B}$ are the same.

Case 1: $\mathcal{A} = \mathcal{A}_1$ and $\mathcal{B} = \mathcal{B}_1$

Set up. The algorithm $\mathcal{B}_1$ picks $w_1, \ldots, w_{q_S} \in \mathbb{Z}_p$ uniformly at random with replacement, and sends them to the challenger. In return, $\mathcal{B}_1$ receives the public key $\mathsf{PK} = (g_1, g_2, u, \zeta)$ and the signatures $\sigma_1, \ldots, \sigma_{q_S}$ for the messages $w_1, \ldots, w_{q_S}$. For convenience, we write $u = g_2^x$ where $x \in \mathbb{Z}_p^*$. Since $\mathsf{PK}_{\mathcal{A}}$ is generated by the *KeyGen* algorithm, g_1 and g_2 are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and x is a random element of $\mathbb{Z}_p^*$. $\mathcal{B}_1$ checks if any of the signatures satisfy $\sigma_i = 1$. If this holds for some i, then $x = -w_i$, and $\mathcal{B}_1$ can produce a forgery on any message of its choice. Otherwise, $\mathcal{B}_1$ proceeds to the next step.

Key generation. $\mathcal{B}_1$ picks $y \in \mathbb{Z}_p^*$ randomly, and provides $\mathcal{A}_1$ with a public key $\mathsf{PK}_{\mathcal{A}} = (g_1, g_2, U \leftarrow u, V \leftarrow g_2^y, \zeta)$. Note that $\mathsf{PK}_{\mathcal{A}}$ is indistinguishable from a public key picked uniformly at random from the key space.

Queries and response. $\mathcal{A}_1$ asks $\mathcal{B}_1$ for signatures on messages $m_1, \ldots, m_{q_S} \in \mathbb{Z}_p$ in an adaptive manner. Upon receiving m_i, the algorithm $\mathcal{B}_1$ simply calculates $r_i \leftarrow \frac{w_i - m_i}{y}$, and gives (σ_i, r_i) to $\mathcal{A}_1$ as a signature on m_i. Note that this gives $\sigma_i = g_1^{\frac{1}{x+w_i}} = g_1^{\frac{1}{x+m_i+yr_i}}$. Also, since w_i is uniform in $\mathbb{Z}_p \backslash \{-x\}$, the randomizer r_i is uniform in $\mathbb{Z}_p \setminus \left\{-\frac{x+m_i}{y}\right\}$, which makes (σ_i, r_i) a valid signature on m_i.

Full signature output. With probability ϵ, the algorithm $\mathcal{A}_1$ eventually outputs a valid forgery $(m_*, (\sigma_*, r_*))$ of the full signature scheme such that $\sigma_* = g_1^{\frac{1}{x+m_*+yr_*}}$.

Basic signature output. The algorithm $\mathcal{B}_1$ outputs $(m_* + yr_*, \sigma_*)$ as a forgery to the basic signature scheme. Since $\sigma_* \neq \sigma_i$ for all $i \in \{1, \ldots, q_S\}$, it follows that $(m_* + yr_*, \sigma_*)$ is a valid forgery of the basic signature scheme.

Case 2: $\mathcal{A} = \mathcal{A}_2$ and $\mathcal{B} = \mathcal{B}_2$

Set up. The algorithm $\mathcal{B}_2$ picks $w_1, \ldots, w_{q_S} \in \mathbb{Z}_p^*$ uniformly at random with replacement, and sends them to the challenger. In return, $\mathcal{B}_2$ receives the public key $\mathsf{PK} = (g_1, g_2, u, \zeta)$ and the signatures $\sigma_1, \ldots, \sigma_{q_S}$ of the messages $w_1, \ldots, w_{q_S}$. For convenience, we write $u = g_2^y$ where $y \in \mathbb{Z}_p^*$. Since $\mathsf{PK}_{\mathcal{A}}$ is generated by the *KeyGen* algorithm, g_1 and g_2 are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and y is a random element of $\mathbb{Z}_p^*$. $\mathcal{B}_2$ checks if any of the signatures satisfy $\sigma_i = 1$. If this holds for some i, then $y = -w_i$, and $\mathcal{B}_2$ can produce a forgery on any message of its choice. Otherwise, $\mathcal{B}_2$ proceeds to the next step.

Key generation. The algorithm $\mathcal{B}_2$ picks $x \in \mathbb{Z}_p^*$ randomly, and provides $\mathcal{A}_2$ with a public key $\mathsf{PK}_{\mathcal{A}} = (g_1, g_2, U \leftarrow g_2^x, V \leftarrow u, \zeta)$. Note that $\mathsf{PK}_{\mathcal{A}}$ is indistinguishable from a public key picked uniformly at random from the key space.

Queries and response. The algorithm $\mathcal{A}_2$ asks $\mathcal{B}_2$ for signatures on messages $m_1, \ldots, m_{q_S}$ in an adaptive manner. Upon receiving m_i, the algorithm $\mathcal{B}_2$ flips a weighted coin that gives $c_i = 1$ with probability $\frac{p-2}{p-1}$ and $c_i = 0$ with probability $\frac{1}{p-1}$. If $c_i = 1$, then $\mathcal{B}_2$ lets $r_i \leftarrow \frac{x+m_i}{w_i}$, and calculates

$$\sigma_i^{(\mathcal{A})} \leftarrow \sigma_i^{1/r_i} = g_1^{\frac{1}{w_i r_i + y r_i}} = g_1^{\frac{1}{x+m_i+y r_i}}.$$

If $c_i = 0$, then $\mathcal{B}_2$ simply lets $r_i \leftarrow 0$, and calculates $\sigma_i^{(\mathcal{A})} \leftarrow g_1^{\frac{1}{x+m_i}}$. The algorithm $\mathcal{B}_2$ then gives $(\sigma_i^{(\mathcal{A})}, r_i)$ to $\mathcal{A}_2$ as a signature of m_i. Note that r_i is uniform in $\mathbb{Z}_p \backslash \{-\frac{x+m_i}{y}\}$, because $\frac{x+m_i}{w_i}$ is uniform in $\mathbb{Z}_p^* \backslash \{-\frac{x+m_i}{y}\}$. Therefore, $(\sigma_i^{(\mathcal{A})}, r_i)$ is a valid signature of message m_i.

Full signature output. With probability ϵ, the algorithm $\mathcal{A}_2$ eventually outputs a valid forgery $(m_*, (\sigma_*, r_*))$ of the full signature scheme.

Basic signature output. Let $i \in \{1, \ldots, q_S\}$ be such that $\sigma_i^{(\mathcal{A})} = \sigma_*$. We know that $r_* \neq r_i$, because otherwise we would also have $m_* = m_i$, which makes $(m_*, (\sigma_*, r_*))$ an invalid forgery. Therefore,

$$\begin{aligned} \sigma_* = g_1^{\frac{1}{x+m_*+y r_*}} &= g_1^{\frac{1}{x+m_i+y r_i}} = \sigma_i^{(\mathcal{A})} \\ \implies y &= \frac{m_i - m_*}{r_* - r_i}. \end{aligned}$$

Thus, $\mathcal{B}_2$ has found the secret key to the basic signature scheme, and it can forge any signature of its choice.

From the above discussion, we know that $\mathcal{B}$ succeeds in forging the basic signature scheme if the types of $\mathcal{A}$ and $\mathcal{B}$ are the same and $\mathcal{A}$ successfully forges the full signature scheme. This happens with probability $\frac{1}{2}\epsilon$. We note that in some cases it is possible for $\mathcal{B}$ to exploit $\mathcal{A}$ even when the types of the two algorithms do not match. However, such cases contribute very little to the overall success probability of $\mathcal{B}$, and we can safely ignore them. The overall running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$ which is t, and the time required to calculate r_i and $\sigma_i^{(\mathcal{A})}$ for all $i \in \{1, \ldots, q_S\}$, which is $\Theta(q_S T)$. □

4.3 Method of Partial Fractions

In the next two sections and in Chapter 5, we prove a series of theorems, all of which involve using either an SDH solver or Cheon's algorithm [25] to forge Boneh-Boyen signatures. In order to do so, we need to construct an input to one of these algorithms using queried signatures. We use a technique involving partial fractions to accomplish this task. The following result is a refinement and generalization of a formula given by Mitsunari et al. [41].

4.3.1 Lemma. *Let $\mathbb{F}$ be a field, and $x \in \mathbb{F}$. Let $d, k \in \mathbb{Z}$ be such that $d \geq 1$, $k \geq 0$. Let m_i for $i = 1, \ldots, d$ be distinct elements of $\mathbb{F}$ such that $x + m_i \neq 0$. Then,*

$$\frac{x^k}{\prod_{i=1}^{d}(x+m_i)} = \begin{cases} \displaystyle\sum_{i=1}^{d} \frac{(-m_i)^k}{(x+m_i)\prod_{j\neq i}(m_j - m_i)} & \text{for } 0 \leq k < d, \\ \displaystyle 1 + \sum_{i=1}^{d} \frac{(-m_i)^d}{(x+m_i)\prod_{j\neq i}(m_j - m_i)} & \text{for } k = d, \\ \displaystyle x + \sum_{i=1}^{d} \left[-m_i + \frac{(-m_i)^{d+1}}{(x+m_i)\prod_{j\neq i}(m_j - m_i)} \right] & \text{for } k = d+1. \end{cases}$$

Proof. By the principle of permanence of identity [4, p. 456], it suffices to prove that the equations hold when $\mathbb{F} = \mathbb{C}$, since they then form an algebraic identity. Thus, we let

$$f(x) = \frac{x^k}{(x+m_1)\cdots(x+m_d)},$$

and treat $f(x)$ as a complex function in x. We can write $f(x)$ as a partial fraction of the form

$$f(x) \;=\; a_k x + b_k + \frac{c_1}{x+m_1} + \frac{c_2}{x+m_2} + \cdots + \frac{c_d}{x+m_d}$$

where

$$a_k \;=\; \begin{cases} 1 & \text{if } k = d+1, \\ 0 & \text{otherwise,} \end{cases} \qquad b_k \;=\; \begin{cases} -\sum_{i=1}^{d} m_i & \text{if } k = d+1, \\ 1 & \text{if } k = d, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

and each c_i is a constant. By symmetry, we only need to prove

$$c_1 = \frac{(-m_1)^k}{\prod_{j\neq 1}(m_j - m_1)}.$$

Note that

$$f(x) - \frac{c_1}{x+m_1} = a_k x + b_k + \frac{c_2}{x+m_2} + \cdots + \frac{c_d}{x+m_d}$$

has an analytic Taylor series expansion about $x = -m_1$. Thus c_1 is the residue of f at the simple pole $x = -m_1$. If we write $f(x) = \frac{\phi(x)}{x+m_1}$ where $\phi(x) = \frac{x^k}{(x+m_2)\cdots(x+m_d)}$, then $\phi(x)$ is analytic and nonzero at $x = -m_1$. A standard theorem in complex analysis (see [22, p. 234] or [6, p. 115]) gives

$$c_1 = \phi(-m_1) = \frac{(-m_1)^k}{\prod_{j\neq 1}(m_j - m_1)}$$

as desired. □

4.3.2 Corollary. *Let $\mathbb{G}$ be a cyclic group of order p. Let $g \in \mathbb{G}$ be a generator and $x \in \mathbb{Z}_p$. If $m_1, \ldots, m_d$ are distinct elements of $\mathbb{Z}_p$ such that $x + m_i \not\equiv 0 \pmod{p}$, then we have*

$$g^{\frac{x^k}{\prod_{i=1}^{d}(x+m_i)}} = \begin{cases} \displaystyle\prod_{i=1}^{d} g^{\frac{(-m_i)^k}{(x+m_i)\prod_{j\neq i}(m_j-m_i)}} & \text{for } 0 \le k < d, \\ \displaystyle g \cdot \prod_{i=1}^{d} g^{\frac{(-m_i)^d}{(x+m_i)\prod_{j\neq i}(m_j-m_i)}} & \text{for } k = d, \\ \displaystyle g^x \cdot g^{-\sum_{i=1}^{d} m_i} \cdot \prod_{i=1}^{d} g^{\frac{(-m_i)^{d+1}}{(x+m_i)\prod_{j\neq i}(m_j-m_i)}} & \text{for } k = d+1. \end{cases}$$

Assume that all the m_i and the $g^{\frac{1}{x+m_i}}$ are known. Further, assume that g is also known in the last two cases, and g^x is known in the last case. Then, calculating $g^{\frac{x^k}{\prod_{i=1}^{d}(x+m_i)}}$ for a single value of k takes $\Theta(dT + d^2T_p)$ time, where T is the upper bound for the time needed for one exponentiation in $\mathbb{G}$, and T_p is the time needed for one operation in $\mathbb{Z}_p$. Calculating all of $g^{\frac{1}{\prod_{i=1}^{d}(x+m_i)}}, g^{\frac{x}{\prod_{i=1}^{d}(x+m_i)}}, \ldots, g^{\frac{x^{d+1}}{\prod_{i=1}^{d}(x+m_i)}}$ takes $\Theta(d^2T)$ time.

Proof. The first part is a straightforward application of Lemma 4.3.1, and the runtime of $\Theta(dT + d^2T_p)$ is obvious from the equations. Next, given

$$g^{\frac{(-m_i)^{k-1}}{(x+m_i)\prod_{j\neq i}(m_j-m_i)}}$$

for all $i = 1, \ldots, d$, we note that calculating $g^{\frac{x^k}{\prod_{i=1}^{d}(x+m_i)}}$ takes $\Theta(dT)$ time for $k \le d$, and $\Theta(dT + dT_p)$ time for $k = d+1$. Thus, calculating $g^{\frac{x^k}{\prod_{i=1}^{d}(x+m_i)}}$ for all $k = 0, \ldots, d+1$ takes $\Theta(d^2T + d^2T_p) = \Theta(d^2T)$ time in total. □

4.4 Reduction from the Basic Signature Scheme to SDH′

In this section, we show that an existential forgery of the basic Boneh-Boyen signature scheme under chosen message attack (indeed, under known message attack) reduces to the q-SDH′ problem. This result is the converse of Theorem 4.1.1 (Lemma 9 in [14]), and it also serves to illustrate the main idea behind the corresponding result for the full signature scheme that we present in Theorem 4.5.1.

4.4.1 Theorem. *If there is an algorithm that (t', ϵ')-breaks the q-SDH′ problem, then we can (t, q_S, ϵ)-weakly break the basic Boneh-Boyen signature scheme presented in Section 3.1.2 provided that*

$$t \ge t' + \Theta(q^2T), \quad q_S \ge q, \quad \text{and} \quad \epsilon \le \frac{p-1-q}{p-1}\,\epsilon',$$

where T is the upper bound for the time needed for one exponentiation in $\mathbb{G}_1$.

Proof. Let $\mathcal{A}$ be an algorithm that (t', ϵ')-breaks the q-SDH$'$ problem. We construct an algorithm $\mathcal{B}$ which performs existential forgeries of the basic signature scheme under a known message attack.

First, the algorithm $\mathcal{B}$ selects distinct random messages $m_1, \ldots, m_{q_S}$, and send them to the challenger. The challenger returns a public key (g_1, g_2, g_2^x, ζ) together with the valid signatures

$$(\sigma_1, \ldots, \sigma_{q_S}) = (g_1^{1/(x+m_1)}, \ldots, g_1^{1/(x+m_{q_S})}).$$

of messages $m_1, \ldots, m_{q_S}$.

Let $h_k \leftarrow g_1^{\frac{x^k}{(x+m_1)\cdots(x+m_q)}}$ for each $k = 0, \ldots, q$. The algorithm $\mathcal{B}$ calculates $(h_0, h_1, \ldots, h_q)$ via Corollary 4.3.2, and feeds the $(q+3)$-tuple $(h_0, h_1, \ldots, h_q, g_2, g_2^x)$ to $\mathcal{A}$. With probability ϵ', the algorithm $\mathcal{A}$ returns

$$(m_*, g_1^{\frac{1}{(x+m_1)\cdots(x+m_q)(x+m_*)}})$$

for some $m_* \in \mathbb{Z}_p$.

We claim that m_* is not equal to any of the m_i except with negligible probability. To show this, observe that g_1 is not disclosed to $\mathcal{A}$ and that $g_1 = h_k^{\frac{(x+m_1)\cdots(x+m_q)}{x^k}}$ for all $k = 0, \ldots, q$. Thus, from the point of view of $\mathcal{A}$, any combination of $m_1, \ldots, m_q$ is equally likely to give rise to a fixed input $(h_0, h_1, \ldots, h_q)$. That is, $\mathcal{A}$ has no better than random chance of choosing an m_* which coincides with one of $m_1, \ldots, m_q$. Therefore, $m_* \neq m_i$ for all $i = 1, \ldots, q$ with probability at least $\frac{p-1-q}{p-1}$. If $m_* = m_i$ for some $1 \leq i \leq q$, then $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ can calculate $\sigma_* = g_1^{\frac{1}{x+m_*}}$ as follows. By Lemma 4.3.1,

$$\begin{aligned}&\frac{1}{(x+m_1)\cdots(x+m_q)(x+m_*)}\\ =&\frac{1}{(x+m_*)\prod_{j=1}^{q}(m_j-m_*)} + \sum_{i=1}^{q}\frac{1}{(x+m_i)\prod_{j\neq i}(m_j-m_i)},\end{aligned}$$

and therefore

$$\begin{aligned}&\frac{1}{x+m_*}\\ =&\left[\frac{1}{(x+m_1)\cdots(x+m_q)(x+m_*)} - \sum_{i=1}^{q}\frac{1}{(x+m_i)\prod_{j\neq i}(m_j-m_i)}\right]\prod_{j=1}^{q}(m_j-m_*).\end{aligned}$$

Thus $\mathcal{B}$ can calculate

$$\sigma_* \leftarrow \left[g_1^{\frac{1}{(x+m_1)\cdots(x+m_q)(x+m_*)}} / \prod_{i=1}^{q}\left(g_1^{\frac{1}{x+m_i}}\right)^{\prod_{j\neq i}\frac{1}{m_j-m_i}}\right]^{\prod_{j=1}^{q}(m_j-m_*)} = g_1^{\frac{1}{x+m_*}}.$$

In this way $\mathcal{B}$ can output (m_*, σ_*) which is a valid forgery for the basic signature scheme.

The bounds for ϵ and q_S are obvious from the construction of $\mathcal{B}$. The running time is bounded by the calculation of

$$g_1^{\frac{1}{(x+m_1)\cdots(x+m_q)}}, g_1^{\frac{x}{(x+m_1)\cdots(x+m_q)}}, \ldots, g_1^{\frac{x^q}{(x+m_1)\cdots(x+m_q)}},$$

which takes $\Theta(q^2 T)$ time by Corollary 4.3.2, and the query of $\mathcal{A}$, which takes time t'. □

Recall that in practice g_1 can be omitted from the public key of the Boneh-Boyen basic signature scheme. If g_1 is not public, then $\mathcal{B}$ cannot compute the value of h_q in the above proof. One workaround is to allow $\mathcal{B}$ to make $q+1$ signature queries, and calculate

$$h'_k \leftarrow g_1^{\frac{x^k}{(x+m_1)\cdots(x+m_{q+1})}}$$

for all $k = 0, \ldots, q$, instead of $h_0, \ldots, h_q$. In this case, the statement of Theorem 4.4.1 still holds, provided that the condition $q_S \geq q$ is replaced by $q_S \geq q+1$.

4.5 Reduction from the Full Signature Scheme to SDH′

We now show that strong existential forgery of the full Boneh-Boyen signature scheme under chosen message attack can be reduced to the q-SDH′ problem. This result is the converse of Theorem 4.2.1 (Theorem 8 in [14]).

4.5.1 Theorem. *If there is an algorithm that (t', ϵ')-breaks the q-SDH′ problem, then we can (t, q_S, ϵ)-break the full Boneh-Boyen signature scheme in Section 3.1.2 provided that*

$$t \geq t' + \Theta(q_S^2 T), \quad q_S \geq q+1, \quad \text{and} \quad \epsilon \leq \frac{(p-2-q)\,(p-1-(q^2+q)/2)}{(p-1)^2}\,\epsilon',$$

where T is the upper bound for the time needed for one exponentiation in $\mathbb{G}_1$.

Proof. Let $\mathcal{A}$ be an algorithm that (t', ϵ')-breaks the q-SDH′ problem. Using $\mathcal{A}$, we construct a forger $\mathcal{B}$ for the full signature scheme under a chosen message attack.

First, $\mathcal{B}$ receives the public key $(g_1, g_2, g_2^x, g_2^y, \zeta)$ from the challenger. Next, $\mathcal{B}$ randomly selects a message $m_* \in \mathbb{Z}_p$, and queries the challenger for q_S different signatures of m_*. Each time the challenger receives m_*, it sends back a valid signature $(\sigma_i, r_i) = (g_1^{1/(x+m_*+yr_i)}, r_i)$ to $\mathcal{B}$, where $r_i \in \mathbb{Z}_p$ is chosen at random so that $x + m_* + yr_i \not\equiv 0 \pmod{p}$. In this way, $\mathcal{B}$ obtains q_S valid (and hopefully distinct) signatures $(\sigma_1, r_1), \ldots, (\sigma_{q_S}, r_{q_S})$ of the same message m_*. If $\{r_1, \ldots, r_{q_S}\}$

does not contain $q+1$ distinct elements of $\mathbb{Z}_p$, then $\mathcal{B}$ aborts. Otherwise, let $h \leftarrow g_1^{1/y}$ and $z \leftarrow \frac{x+m_*}{y}$. Then, for each $i = 1, \ldots, q+1$,

$$\sigma_i = g_1^{\frac{1}{x+m_*+yr_i}} = \left(g_1^{\frac{1}{y}}\right)^{\frac{1}{\frac{x+m_*}{y}+r_i}} = h^{\frac{1}{z+r_i}}.$$

For each $k = 1, \ldots, q$, the algorithm $\mathcal{B}$ can calculate

$$h^{\frac{z^k}{(z+r_1)\cdots(z+r_{q+1})}} = \prod_{i=1}^{q+1} \sigma_i^{\frac{(-r_i)^k}{\prod_{j\neq i}(r_j - r_i)}}$$

by Corollary 4.3.2, because $\mathcal{B}$ knows each σ_i and each r_i. Also note that if we let $g_2' \leftarrow g_2^y$, then $g_2^x g_2^{m_*} = {g_2'}^z$. When $\mathcal{B}$ queries $\mathcal{A}$ with the input

$$(h^{\frac{1}{(z+r_1)\cdots(z+r_{q+1})}}, h^{\frac{z}{(z+r_1)\cdots(z+r_{q+1})}}, \ldots, h^{\frac{z^q}{(z+r_1)\cdots(z+r_{q+1})}}, g_2', {g_2'}^z),$$

the algorithm $\mathcal{A}$ returns $(r_*, h^{\frac{1}{(z+r_1)\cdots(z+r_{q+1})(z+r_*)}})$ for some $r_* \in \mathbb{Z}_p$ with probability ϵ'. If $r_* = r_i$ for some $1 \le i \le q+1$, then $\mathcal{B}$ aborts, but this event occurs with only negligible probability, by the same argument as in Theorem 4.4.1. Otherwise, by Lemma 4.3.1,

$$\begin{aligned}&\frac{1}{(z+r_1)\cdots(z+r_{q+1})(z+r_*)}\\ =&\frac{1}{(z+r_*)\prod_{j=1}^{q+1}(r_j - r*)} + \sum_{i=1}^{q+1} \frac{1}{(z+r_i)\prod_{j\neq i}(r_j - r_i)},\end{aligned}$$

and therefore,

$$\frac{1}{z+r_*} = \left[\frac{1}{(z+r_1)\cdots(z+r_{q+1})(z+r_*)} - \sum_{i=1}^{q+1} \frac{1}{(z+r_i)\prod_{j\neq i}(r_j - r_i)}\right] \prod_{j=1}^{q+1}(r_j - r_*).$$

Thus, $\mathcal{B}$ can calculate

$$\sigma_* \leftarrow \left[h^{\frac{1}{(z+r_1)\cdots(z+r_{q+1})(z+r_*)}} / \prod_{i=1}^{q+1} \left(h^{\frac{1}{z+r_i}}\right)^{\prod_{j\neq i}\frac{1}{r_j - r_i}}\right]^{\prod_{j=1}^{q+1}(r_j - r_*)} = h^{\frac{1}{z+r_*}} = g_1^{\frac{1}{x+m_*+yr_*}}.$$

In this way $\mathcal{B}$ outputs $(m_*, (\sigma_*, r_*))$, which is a strong existential forgery for the full signature scheme.

The bound for q_S is obvious from the construction of $\mathcal{B}$. The running time is bounded by the calculation of $h^{\frac{1}{(z+r_1)\cdots(z+r_{q+1})}}$, $h^{\frac{z}{(z+r_1)\cdots(z+r_{q+1})}}$, $\ldots$, $h^{\frac{z^q}{(z+r_1)\cdots(z+r_{q+1})}}$, which takes $\Theta(q^2 T)$ by Corollary 4.3.2, and the query of $\mathcal{A}$, which takes time t'.

The probability that $\mathcal{B}$ succeeds in the attack is $P_1 P_2 \epsilon'$ where P_1 is the probability that the random elements $\{r_1, \ldots, r_{q_S}\}$ returned by the signing oracle comprise

at least $q+1$ distinct elements, and P_2 is the probability that the r_* returned by $\mathcal{A}$ differs from the $q+1$ values r_i used by $\mathcal{B}$. We know that $P_2 \geq \frac{p-2-q}{p-1}$ using the argument from the proof of Theorem 4.4.1. Moreover, $P_1 \geq 1 - Q$ where Q is the probability that among $r_1, \ldots, r_{q+1}$ there exist $1 \leq i < j \leq q+1$ such that $r_i = r_j$. Overall, we have

$$Q \leq \sum_{j=2}^{q+1} \Pr\left(\exists i < j \text{ such that } r_i = r_j\right) \leq \sum_{j=2}^{q+1} \frac{j-1}{p-1} = \frac{q(q+1)}{2(p-1)}$$

so $P_1 \geq 1 - Q \geq \frac{p-1-q(q+1)/2}{p-1}$, which yields the bound for ϵ. □

Again, recall that in practice g_1 is omitted from the public key of the Boneh-Boyen full signature scheme. Unlike in the case of basic signatures, there is no difference in the proof in this case even if g_1 is not made public. The statement and the proof remain the same.

4.6 Boneh-Boyen Signatures (Original Version)

In Theorems 4.4.1 and 4.5.1, we forged Boneh-Boyen signatures from the full version of the paper [14] using SDH′ solvers, thereby proving the converse of the security theorems from that paper. We now consider the corresponding result for the original version of the paper [13].

The core technique that we used to prove the two theorems is to calculate consecutive powers of the form $h_1^{\frac{1}{(z+r_1)\cdots(z+r_q)}}, \ldots, h_1^{\frac{z^q}{(z+r_1)\cdots(z+r_q)}} \in \mathbb{G}_1$ from signatures of the form $h_1^{\frac{1}{z+r_1}}, \ldots, h_1^{\frac{1}{z+r_q}} \in \mathbb{G}_1$ using Corollary 4.3.2. This is still possible under the original version of the Boneh-Boyen signature scheme. Thus, an SDH′ solver yields an algorithm for forging Boneh-Boyen signatures under the original scheme.

However, the security assumption used for the version 1 Boneh-Boyen signature scheme in [13] is the SDH assumption rather than the SDH′ assumption. Thus, a true converse to the security theorem of the version 1 scheme would be to forge version 1 signatures using an SDH solver. If we knew that the SDH′ problem was no harder than the SDH problem, then this result would be immediate. Unfortunately, this is not known, even though the reverse implication, that SDH is no harder than SDH′, is known to hold under certain conditions (Lemma 2.2.2).

We thus consider the question of proving that an SDH solver implies the existence of a forger for version 1 signatures, by applying the same technique as the one used in Theorem 4.4.1 and Theorem 4.5.1. In such a proof, we would need to use version 1 signatures to construct consecutive powers in the group $\mathbb{G}_2$ rather than $\mathbb{G}_1$, because of the difference between the input needed for SDH and SDH′. The problem is that the signatures of the version 1 scheme are elements of $\mathbb{G}_1$ rather than of $\mathbb{G}_2$. Thus, in order to construct consecutive powers of the form

$h_2^{\frac{1}{(z+r_1)\cdots(z+r_q)}}, \ldots, h_2^{\frac{z^q}{(z+r_1)\cdots(z+r_q)}}$, we would need to construct consecutive powers in $\mathbb{G}_1$, and then apply the inverse ψ^{-1} of the isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$. Now recall from the classification of pairings introduced in Section 1.2 that not every bilinear group pair admits efficiently computable isomorphisms in both directions. From the above observations, we conclude that a reduction similar to Theorem 4.4.1 and Theorem 4.5.1 applies to version 1 signatures only when the scheme is implemented using a Type 1 bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$.

Chapter 5

Security Analysis of Boneh-Boyen Signatures

It is easy to see that using a generic DLOG algorithm that runs in time t', the secret key of any version of the Boneh-Boyen signature scheme can be recovered in time t'. In fact, when Boneh and Boyen signatures were first introduced, the only known attack against the scheme was to apply a generic DLOG algorithm such as Pollard rho [47] or Shanks' baby-step giant-step algorithm [52].

We observed in the previous chapter that Boneh-Boyen signatures can be forged if there is an algorithm that efficiently solves the SDH$'$ problem. In other words, our result indicates that we do not need an algorithm as powerful as solving DLOG in order to forge the signatures; an SDH$'$ algorithm suffices.

Cheon [25] presents a pair of algorithms which under some circumstances compute the secret exponent x from the input of an instance of the q-SDH problem. Specifically, Cheon proves the following theorem:

5.0.1 Theorem. *Let $\mathbb{G}$ be a cyclic group of prime order p and let g be a generator. Let T denote the upper bound for the time needed for one exponentiation in $\mathbb{G}$.*

1. *Given a divisor d of $p-1$ and the group elements g, g^x, and g^{x^d}, the value of x can be recovered in time $O\left(\left(\sqrt{p/d}+\sqrt{d}\right)T\right)$.*

2. *Given a divisor d of $p+1$ and the group elements g, $g^x, g^{x^2}, \ldots, g^{x^{2d}}$, the value of x can be recovered in time $O\left(\left(\sqrt{p/d}+d\right)T\right)$.*

We refer to the algorithms that accomplish the first and the second task in Theorem 5.0.1 as *Cheon's $p-1$ algorithm* and *Cheon's $p+1$ algorithm*, respectively.

Note that Cheon's $p-1$ algorithm and the $p+1$ algorithm solve the q-SDH$'$ problem if $q \geq d$ in the first case or $q \geq 2d$ in the second; in fact, they reveal the secret exponent x. Thus, when applied to Boneh-Boyen signatures, Cheon's algorithms not only produce a forgery but reveal the secret key. We discuss exactly how to apply Cheon's algorithms to Boneh-Boyen signatures in Section 5.3.

5.1 Shanks' Baby-Step Giant-Step Algorithm

Cheon's algorithms can be implemented using Shanks' baby-step giant-step algorithm [52] or the Pollard lambda method [47]. For practical applications, the Pollard lambda method is preferable because it requires much less memory. However, for convenience, we will only discuss the baby-step giant-step algorithm. Let $\mathbb{G}$ be a cyclic group of order n. Recall that the discrete log (DLOG) problem is as follows:

Discrete Logarithm (DLOG)

Input: a generator $g \in \mathbb{G}$ and $h \in \mathbb{G}$

Output: the nonnegative integer $x < n$ such that $h = g^x$

Note that the brute force method of finding x would take $O(n)$ operations in $\mathbb{G}$. The baby-step giant-step algorithm accomplishes the task in $O(\sqrt{n})$ operations with $O(\sqrt{n})$ space.

Given g and $h = g^x \in \mathbb{G}$, let $m \leftarrow \lceil \sqrt{n} \rceil$ and observe that $x < m^2$. The idea is to find x by finding two integers q and r such that $x = mq + r$ and $0 \leq q, r < m$. For such q and r, we have $hg^{-r} = g^{mq}$. The algorithm is as follows:

1: $m \leftarrow \lceil \sqrt{n} \rceil$
2: **for** $i = 0$ to $m - 1$ **do**
3: compute $L_i \leftarrow hg^{-i}$, and store (L_i, i) in a hash table
4: **end for**
5: **for** $j = 0$ to $m - 1$ **do**
6: $R_j \leftarrow g^{mj}$
7: **if** the hash table has an entry (L_i, i) with $L_i = R_j$ **then**
8: **return** $i + mj$
9: **end if**
10: **end for**

We calculate L_i's and R_j's from the previous terms by repeated multiplication by g^{-1} and g^m, respectively. Each hash table look-up takes constant time. Thus the entire algorithm takes $O(\sqrt{n})$ operations in $\mathbb{G}$.

5.2 Cheon's Algorithm for SDH

5.2.1 Cheon's $p - 1$ Algorithm

We prove the first part of Theorem 5.0.1 in this section. Suppose we are given input $g, g_1, g_d \in \mathbb{G}$ where $g_i = g^{x^i}$ for some $x \in \mathbb{Z}_p^*$. We want to find x.

Instead of directly finding x using the baby-step giant-step algorithm, we write $x = \zeta_0^k$ for some fixed generator ζ_0 of $\mathbb{Z}_p^*$, and look for the value k. We do this by dividing the algorithm into two phases, and applying an idea similar to the baby-step giant-step in each phase. The two phases accomplish the following:

Phase 1. Find k_0 such that $x^d = (\zeta_0^d)^{k_0}$ so that

$$dk_0 \equiv dk \pmod{p-1} \implies k \equiv k_0 \pmod{\frac{p-1}{d}}.$$

Phase 2. Find the quotient m such that $k = k_0 + m\frac{p-1}{d}$. We can then calculate k from k_0 and m.

The full proof follows.

Proof of Theorem 5.0.1.1.

Phase 1. Suppose $g, g_1, g_d \in \mathbb{G}$ and $x \in \mathbb{Z}_p^*$ are given where each $g_i = g^{x^i}$. Let ζ_0 be a generator of $\mathbb{Z}_p^*$, and let $\zeta \leftarrow \zeta_0^d$. Then, ζ generates all the $\frac{p-1}{d}$th roots of unity. Since x^d is a $\frac{p-1}{d}$th root of unity, there exists a nonnegative integer k_0 less than $\frac{p-1}{d}$ such that $x^d = \zeta^{k_0}$. Let $d_1 \leftarrow \left\lceil \sqrt{\frac{p-1}{d}} \right\rceil$, and write $k_0 = d_1 u + v$ for some nonnegative integers u and v with $v < d_1$. Since $d_1^2 \geq \frac{p-1}{d} > k_0 \geq d_1 u$, we must have $u < d_1$ as well. For such u and v, we have

$$(x^d)\zeta^{-v} = \zeta^{d_1 u},$$

and thus

$$g_d^{\zeta^{-v}} = g^{\zeta^{d_1 u}}. \tag{5.2.1}$$

Conversely, if integers u and v satisfy (5.2.1), then $k_0 \equiv d_1 u + v \pmod{\frac{p-1}{d}}$. Since $k_0 < \frac{p-1}{d}$, it suffices to find the smallest nonnegative integer u and corresponding v that satisfy (5.2.1).

To do so, we first calculate $L_i \leftarrow g_d^{\zeta^{-i}}$ for all $i = 0, 1, \ldots, d_1 - 1$ by repeated exponentiation by ζ^{-1}, and store the entries of the form (L_i, i) in a hash table. Next, for $j = 0, 1, \ldots, d_1 - 1$, we compute $R_j \leftarrow g^{\zeta^{d_1 j}}$ (by exponentiating R_{j-1} by ζ^{d_1}) until a hash table entry (L_i, i) such that $L_i = R_j$ is found. For such i and j, let $u \leftarrow i$ and $v \leftarrow j$ so that $k_0 = d_1 u + v$. Phase 1 takes time $O(d_1 T) = O(\sqrt{\frac{p-1}{d}} T)$.

Phase 2. Let $x = \zeta_0^k$ for some integer $0 \leq k < p-1$. Then, $dk \equiv dk_0 \pmod{p-1}$ so that $k \equiv k_0 \pmod{\frac{p-1}{d}}$. Write $k = k_0 + m\frac{p-1}{d}$ for some integer m such that $0 \leq m < d$. Let $d_2 \leftarrow \left\lceil \sqrt{d} \right\rceil$. Then we can write $m = d_2 u' + v'$ for some nonnegative

integers u', v' with $v' < d_2$. Since $d_2^2 \geq d > m \geq d_2 u'$, we must have $u' < d_2$ as well. For such u' and v', we have $k - v'\frac{p-1}{d} = k_0 + d_2 u'\frac{p-1}{d}$ so that

$$x\zeta_0^{-v'\frac{p-1}{d}} = \zeta_0^{k_0}\zeta_0^{d_2 u'\frac{p-1}{d}},$$

which implies

$$g_1^{\zeta_0^{-v'\frac{p-1}{d}}} = \left(g^{\zeta_0^{k_0}}\right)^{\zeta_0^{d_2 u'\frac{p-1}{d}}}. \tag{5.2.2}$$

Conversely, if integers u' and v' satisfy (5.2.2), then $m \equiv d_2 u' + v' \pmod{d}$. Since $m < d$, it suffices to find the smallest nonnegative integer u' and corresponding v' that satisfy (5.2.2). We follow a procedure similar to what we did in Phase 1 to find u' and v'; we repeatedly exponentiate the left hand side and the right hand side by $\zeta_0^{-\frac{p-1}{d}}$ and $\zeta_0^{d_2\frac{p-1}{d}}$, respectively. Phase 2 takes $O(d_2 T) = O(\sqrt{d}T)$ time.

The total time needed for the above process is $O\left(\left(\sqrt{\frac{p-1}{d}} + \sqrt{d}\right)T\right)$. □

5.2.2 Cheon's $p+1$ Algorithm

We prove the second part of Theorem 5.0.1 in this section. Suppose we are given input $g, g_1, \ldots, g_{2d} \in \mathbb{G}$ where $g_i = g^{x^i}$ for some $x \in \mathbb{Z}_p^*$. We want to find x.

As in the previous section, the algorithm consists of two phases, in each of which we apply an idea similar to the baby-step giant-step algorithm. Let a be a quadratic non-residue modulo p and let θ be a root of $y^2 = a$ in some extension field $\mathbb{F}$ of $\mathbb{Z}_p$. Then, there exists a subgroup $\mathbb{H} \subseteq \mathbb{F}^*$ of order $p+1$. We define an embedding from $\mathbb{Z}_p$ into $\mathbb{H}$ such that $x \mapsto \beta$, and find β instead of x in a baby-step giant-step manner. We find β^d in Phase 1, and we find β in Phase 2. Finally, we recover x by inverting the embedding. The full proof follows.

Proof of Theorem 5.0.1.2.

Set up. Suppose $g, g_1, \ldots, g_{2d} \in \mathbb{G}$ and $x \in \mathbb{Z}_p^*$ are given where each $g_i = g^{x^i}$. Let a be a quadratic non-residue modulo p, and let θ be a root of $y^2 = a$ in some extension field $\mathbb{F}$ of $\mathbb{Z}_p$. Then, it is easy to verify that $\mathbb{Z}_p[\theta]$ is a field with p^2 elements, and so $\mathbb{Z}_p[\theta]^*$ is a cyclic group of order $p^2 - 1$. Let $\mathbb{H}$ be the subgroup of $\mathbb{Z}_p[\theta]^*$ of order $p+1$.

We consider mapping x to an element $\beta \leftarrow \beta_0 + \beta_1\theta$ in $\mathbb{Z}_p[\theta]^*$ where

$$\beta_0 = \frac{1+ax^2}{1-ax^2} \in \mathbb{Z}_p \quad \text{and} \quad \beta_1 = \frac{2x}{1-ax^2} \in \mathbb{Z}_p.$$

We claim that $\beta \in \mathbb{H}$. Since every $(p+1)$th root of unity in $\mathbb{Z}_p[\theta]^*$ is in $\mathbb{H}$, it suffices to show $\beta^{p+1} = 1$. Note that $\theta^p = \theta a^{\frac{p-1}{2}} = -\theta$ because a is a quadratic non-residue

modulo p. Therefore, we have

$$\begin{aligned}\beta^{p+1} &= (\beta_0 + \beta_1\theta)^p(\beta_0 + \beta_1\theta) \\ &= (\beta_0^p + \beta_1^p\theta^p)(\beta_0 + \beta_1\theta) \\ &= (\beta_0 - \beta_1\theta)(\beta_0 + \beta_1\theta) \\ &= \beta_0^2 - a\beta_1^2 = 1.\end{aligned}$$

Phase 1. Let γ be a primitive element of $\mathbb{Z}_p[\theta]$, so that $\zeta_0 \leftarrow \gamma^{p-1}$ generates $\mathbb{H}$. Let $\zeta \leftarrow \zeta_0^d$. Then ζ generates all the $\frac{p+1}{d}$th roots of unity. Since β^d is a $\frac{p+1}{d}$th root of unity, there exists an integer k_0 with $0 \le k_0 < \frac{p+1}{d}$ such that $\beta^d = \zeta^{k_0}$. We find k_0 using the baby-step giant-step method. Let $d_1 \leftarrow \left\lceil \sqrt{\frac{p+1}{d}} \right\rceil$. By the division algorithm, there exist nonnegative integers u and v such that

$$k_0 = d_1 u + v \tag{5.2.3}$$

and $v < d_1$. Since $d_1^2 \ge \frac{p+1}{d} > k_0 \ge d_1 u$, we also have $u < d_1$. (5.2.3) implies

$$\beta^d \zeta^{-v} = \zeta^{d_1 u}. \tag{5.2.4}$$

We write

$$\beta^d = \frac{1}{(1-ax^2)^d}\left(f_0(x) + f_1(x)\theta\right)$$

where $f_0(x)$ and $f_1(x)$ are polynomials of degree $2d$ and $2d-1$, respectively. Also, we let

$$\zeta^i = s_i + t_i\theta.$$

Then (5.2.4) is equivalent to

$$\frac{1}{(1-ax^2)^d}\left(f_0(x) + f_1(x)\theta\right)\left(s_{-v} + t_{-v}\theta\right) = s_{d_1u} + t_{d_1u}\theta, \quad \text{or}$$

$$\left[f_0(x)s_{-v} + af_1(x)t_{-v}\right] + \left[f_0(x)t_{-v} + f_1(x)s_{-v}\right]\theta = (1-ax^2)^d s_{d_1u} + (1-ax^2)^d t_{d_1u}\theta.$$

This implies

$$g^{f_0(x)s_{-v}} g^{f_1(x)at_{-v}} = g^{(1-ax^2)^d s_{d_1u}} \quad \text{and} \tag{5.2.5}$$

$$g^{f_0(x)t_{-v}} g^{f_1(x)s_{-v}} = g^{(1-ax^2)^d t_{d_1u}}. \tag{5.2.6}$$

Conversely, if u and v satisfy (5.2.5) and (5.2.6), then $k_0 \equiv d_1u + v \pmod{\frac{p+1}{d}}$. Since $k_0 < \frac{p+1}{d}$, it suffices to pick the smallest u and corresponding v that satisfy (5.2.5) and (5.2.6).

To do so, we first compute $g^{f_0(x)}$, $g^{f_1(x)}$, and $g^{(1-ax^2)^d}$ using the input $g, g_1, \ldots, g_{2d}$. Since $f_0(x)$, $f_1(x)$, and $(1-ax^2)^d$ respectively have degree $2d$, $2d-1$, and $2d$ in x, this can be done in time $O((6d+2)T) = O(dT)$. Next, we calculate ζ^{-i} and $\zeta^{d_1 i}$ for all $i = 0, 1, \ldots, d_1 - 1$ (and thus s_{-i}, t_{-i}, $s_{d_1 i}$, and $t_{d_1 i}$ for $i = 0, 1, \ldots, d_1 - 1$)

by repeatedly multiplying by ζ^{-1} and ζ^{d_1}, respectively. Since each operation in $\mathbb{H} \subset \mathbb{Z}_p[\theta]^*$ requires only a constant number of operations in $\mathbb{Z}_p$, this process takes $O(d_1 T_p) = O\left(\sqrt{\frac{p+1}{d}} T_p\right)$ time, where T_p is the time required for one operation in $\mathbb{Z}_p$.

Then we calculate

$$L_{i,1} \leftarrow \left(g^{f_0(x)}\right)^{s_{-i}} \left(g^{f_1(x)a}\right)^{t_{-i}} \quad \text{and} \quad L_{i,2} \leftarrow \left(g^{f_0(x)}\right)^{t_{-i}} \left(g^{f_1(x)}\right)^{s_{-i}}$$

for all $i = 0, 1, \ldots, d_1 - 1$, and store the entries of the form $(L_{i,1}, L_{i,2}, i)$ in a hash table. Next, for $j = 0, 1, \ldots, d_1 - 1$, we compute

$$R_{j,1} \leftarrow \left(g^{(1-ax^2)^d}\right)^{s_{d_1 j}} \quad \text{and} \quad R_{j,2} \leftarrow \left(g^{(1-ax^2)^d}\right)^{t_{d_1 j}}$$

until a hash table entry $(L_{i,1}, L_{i,2}, i)$ such that $L_{i,1} = R_{j,1}$ and $L_{i,2} = R_{j,2}$ is found. For such i and j, let $u \leftarrow i$ and $v \leftarrow j$ so that $k_0 = d_1 u + v$. The above process takes $O(d_1 T) = O\left(\sqrt{\frac{p+1}{d}} T\right)$ time.

The total time Phase 1 takes is $O\left(\left(d + \sqrt{\frac{p+1}{d}}\right) T\right)$.

Phase 2. Write $\beta = \zeta_0^k$ for some integer $0 \le k < p+1$. Then, $\zeta_0^{dk} = \beta^d = \zeta_0^{dk_0}$ so that $dk \equiv dk_0 \pmod{p+1}$ and $k \equiv k_0 \pmod{\frac{p+1}{d}}$. Write $k = k_0 + m\frac{p+1}{d}$ for some integer m such that $0 \le m < d$. Let $d_2 \leftarrow \left\lceil \sqrt{d} \right\rceil$. Then we can write $m = d_2 u' + v'$ for some nonnegative integers u', v' with $v' < d_2$. Since $d_2^2 \ge d > m \ge d_2 u'$, we must have $u' < d_2$ as well. For such u' and v', we have $k - v'\frac{p+1}{d} = k_0 + d_2 u' \frac{p+1}{d}$ so that

$$\beta \zeta_0^{-v'\frac{p+1}{d}} = \zeta_0^{k_0} \zeta_0^{d_2 u' \frac{p+1}{d}}. \tag{5.2.7}$$

If we write

$$\zeta_0^{-i\frac{p+1}{d}} = s'_i + t'_i \theta \quad \text{and} \quad \zeta_0^{k_0} \zeta_0^{d_2 j \frac{p+1}{d}} = s''_j + t''_j \theta,$$

then (5.2.7) is equivalent to

$$\begin{aligned}
&\frac{(1+ax^2) + 2x\theta}{1-ax^2}(s'_{v'} + t'_{v'}\theta) = s''_{u'} + t''_{u'}\theta \\
\iff &\left[(1+ax^2)s'_{v'} + 2axt'_{v'}\right] + \left[(1+ax^2)t'_{v'} + 2xs'_{v'}\right]\theta = (1-ax^2)s''_{u'} + (1-ax^2)t''_{u'}\theta \\
\iff &\left[(1+ax^2)s'_{v'} + 2axt'_{v'}\right] = (1-ax^2)s''_{u'} \text{ and} \\
&\left[(1+ax^2)t'_{v'} + 2xs'_{v'}\right] = (1-ax^2)t''_{u'}.
\end{aligned}$$

This implies

$$g^{(1+ax^2)s'_{v'}} g^{2axt'_{v'}} = g^{(1-ax^2)s''_{u'}} \qquad \text{and} \tag{5.2.8}$$

$$g^{(1+ax^2)t'_{v'}} g^{2xs'_{v'}} = g^{(1-ax^2)t''_{u'}}. \tag{5.2.9}$$

Conversely, if integers u' and v' satisfy (5.2.8) and (5.2.9), then

$$\begin{aligned} & k_0 + m\frac{p+1}{d} \equiv k \equiv k_0 + (d_2u' + v')\frac{p+1}{d} \pmod{p+1} \\ \Longleftrightarrow & m\frac{p+1}{d} \equiv (d_2u' + v')\frac{p+1}{d} \pmod{p+1} \\ \Longleftrightarrow & m \equiv d_2u' + v' \pmod{d}. \end{aligned}$$

Since $m < d$, it suffices to find the smallest nonnegative integer u' and corresponding v' that satisfy (5.2.8) and (5.2.9).

To do so, we first compute g^{1-ax^2}, g^{2x}, and g^{2ax} using the input g, g_1, g_2 in time $O(T)$. Next, we calculate $\zeta_0^{-i\frac{p+1}{d}}$ and $\zeta_0^{k_0}\zeta_0^{id_2\frac{p+1}{d}}$ for all $i = 0, 1, \ldots, d_2 - 1$ (and thus s'_i, t'_i, s''_i, and t''_i for $i = 0, 1, \ldots, d_2 - 1$) by repeated multiplications by $\zeta_0^{-\frac{p+1}{d}}$ and $\zeta_0^{d_2\frac{p+1}{d}}$, respectively. Since each operation in $\mathbb{H} \subset \mathbb{Z}_p[\theta]^*$ requires only a constant number of operations in $\mathbb{Z}_p$, this process takes $O(d_2T_p) = O\left(\sqrt{d}T_p\right)$ time, where T_p is the time for one operation in $\mathbb{Z}_p$.

The remaining process to find u' and v' using a hash table is very similar to what we did in Phase 1. It requires $O(d_2) = O(\sqrt{d}T)$ time.

The total time needed for Phase 2 is $O\left(\sqrt{d}T\right)$.

Recovering x. Finally, we solve the simultaneous equations

$$\beta_0 = \frac{1+ax^2}{1-ax^2} \quad \text{and} \quad \beta_1 = \frac{2x}{1-ax^2}$$

in $\mathbb{Z}_p$ using $\beta_0^2 - \beta_1^2 a = 1$, and find the unique solution

$$x \leftarrow \frac{\beta_0 - 1}{a\beta_1}.$$

as desired. □

5.3 Recovering the Private Key in the Boneh-Boyen Signature Scheme

We now show how Cheon's algorithms can be applied to find the secret exponent in the Boneh-Boyen signature scheme. As before, we let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order p. We also let T denote the upper bound for the time needed for one exponentiation in $\mathbb{G}_1$ and T_p denote the time needed for one operation in $\mathbb{Z}_p$.

5.3.1 Theorem. *(Basic scheme)*

1. *Suppose a positive divisor d of $p-1$ is given. Given d valid message-signature pairs, the private key x in the basic Boneh-Boyen signature scheme can be computed in time at most $O\left(\left(\sqrt{p/d}+d\right)T+d^2T_p\right)$.*

2. *Suppose a positive divisor d of $p+1$ is given. Given $2d$ valid message-signature pairs, the private key x in the basic Boneh-Boyen signature scheme can be computed in time at most $O\left(\left(\sqrt{p/d}+d^2\right)T\right)$.*

5.3.2 Theorem. *(Full scheme)*

1. *Suppose a positive divisor d of $p-1$ is given. Then the private key pair (x,y) of the full Boneh-Boyen signature scheme can be computed under a chosen message attack, using $2d+2$ signature queries, in time at most $O\left(\left(\sqrt{p/d}+d\right)T+d^2T_p\right)$, with probability at least $\left(\frac{p-1-d(d+1)/2}{p-1}\right)^2$.*

2. *Suppose a positive divisor d of $p+1$ is given. Then the private key pair (x,y) of the full Boneh-Boyen signature scheme can be computed under a chosen message attack, using $4d+2$ signature queries, in time at most $O\left(\left(\sqrt{p/d}+d^2\right)T\right)$, with probability at least $\left(\frac{p-1-d(2d+1)}{p-1}\right)^2$.*

Recall that in practice g_1 is omitted from the public key of Boneh-Boyen signatures. The effect of not having g_1 in the public key on Theorem 5.3.1 and Theorem 5.3.2 is analogous to its effect on Theorem 4.4.1 and Theorem 4.5.1. That is, the full signature scheme (Theorem 5.3.2) is not affected, but one more (message, signature) pair is needed to recover the secret key of the basic scheme (Theorem 5.3.1).

The proofs of these two theorems are similar. We only give the proof of Theorem 5.3.2.

Proof. (1) Let d be a positive divisor of $p-1$. We construct an algorithm $\mathcal{A}$ which recovers the private key of the signature scheme under a chosen message attack, using Cheon's algorithm. Suppose $\mathcal{A}$ is given the public key $(g_1, g_2, g_2^x, g_2^y, \zeta)$. The algorithm $\mathcal{A}$ randomly selects a message $m_a \in \mathbb{Z}_p$, and queries for signatures of this same message $d+1$ times. As a result, $\mathcal{A}$ obtains $d+1$ valid (and hopefully distinct) signatures $(\sigma_1, r_1), \ldots, (\sigma_{d+1}, r_{d+1})$, where $\sigma_i = g_1^{\frac{1}{x+m_a+yr_i}}$ for each $i = 1, \ldots, d+1$. Let $h \leftarrow g_1^{1/y}$ and $z_a \leftarrow \frac{x+m_a}{y}$. Then, we have

$$\sigma_i = \left(g_1^{\frac{1}{y}}\right)^{\frac{1}{\frac{x+m_a}{y}+r_i}} = h^{\frac{1}{z_a+r_i}}$$

for each $i = 1, \ldots, d+1$. If the set $\{r_1, \ldots, r_{d+1}\}$ does not consist of distinct elements, then $\mathcal{A}$ aborts. Otherwise, assume $r_1, \ldots, r_{d+1}$ are distinct. Using Corollary 4.3.2, $\mathcal{A}$ calculates

$$h^{\frac{1}{(z_a+r_1)\cdots(z_a+r_{d+1})}}, h^{\frac{z_a}{(z_a+r_1)\cdots(z_a+r_{d+1})}}, \text{ and } h^{\frac{z_a^d}{(z_a+r_1)\cdots(z_a+r_{d+1})}}.$$

Then, it runs Cheon's $p-1$ algorithm in $\mathbb{G}_1$ with these inputs, and obtains $z_a = \frac{x+m_a}{y}$ as output.

Next, $\mathcal{A}$ repeats the above process with a different random message m_b, and obtains $z_b = \frac{x+m_b}{y}$. Since $\mathcal{A}$ knows z_a, z_b, m_a, and m_b, it can solve a linear system of equations to obtain the private exponents x and y.

Since calculating $h^{\frac{1}{(z+r_1)\cdots(z_a+r_{d+1})}}$, $h^{\frac{z}{(z+r_1)\cdots(z_a+r_{d+1})}}$, and $h^{\frac{z^d}{(z+r_1)\cdots(z_a+r_{d+1})}}$ for $z = z_a$ and z_b takes time $O(dT + d^2T_p)$ and Cheon's algorithm takes $O\left(\left(\sqrt{p/d} + \sqrt{d}\right)T\right)$ time, the overall runtime is $O\left(\left(\sqrt{p/d} + d\right)T + d^2T_p\right)$. The attack succeeds if the set $\{r_1, \ldots, r_{d+1}\}$ for m_a consists of distinct elements (and likewise for m_b). Using an argument analogous to the one used in Theorem 4.4.1, we see that a lower bound for this probability is $\left(\frac{p-1-d(d+1)/2}{p-1}\right)^2$.

(2) We now suppose d is a divisor of $p+1$. The proof here is similar, except that $\mathcal{A}$ needs to calculate

$$h^{\frac{1}{(z+r_1)\cdots(z+r_{2d+1})}}, h^{\frac{z}{(z+r_1)\cdots(z+r_{2d+1})}}, \ldots, h^{\frac{z^{2d}}{(z+r_1)\cdots(z+r_{2d+1})}}.$$

from the signatures $h^{\frac{1}{z+r_1}}, \ldots, h^{\frac{1}{z+r_{2d+1}}}$, in order to obtain $z = z_a$ and z_b using Cheon's $p+1$ algorithm. This takes $O\left(d^2T\right)$ time, and Cheon's algorithm takes $O\left(\left(\sqrt{p/d} + d\right)T\right)$ time, for a total runtime of $O\left(\left(\sqrt{p/d} + d^2\right)T\right)$. The attack succeeds if the set $\{r_1, \ldots, r_{2d+1}\}$ for each of z_a and z_b consists of distinct elements, and the probability of this is at least $\left(\frac{p-1-d(2d+1)}{p-1}\right)^2$. □

From Theorems 5.3.1 and 5.3.2, we find that the private key in the Boneh-Boyen signature scheme can be computed faster than with the Pollard rho algorithm as long as $p-1$ or $p+1$ has a divisor d with $d \lesssim p^{1/4}$, with the minimum running time achieved when $d \approx p^{1/5}$. If a divisor d of $p-1$ or $p+1$ size approximately $p^{1/5}$ exists, then the private key for the basic (resp., full) scheme can be computed using $O(d)$ known (resp., chosen) signatures in time $O(p^{2/5}T)$.

5.4 Countermeasures and Mitigation

Other than increasing the key length, the most direct defense against the above attack is to choose a group order p for which $p-1$ and $p+1$ admit no divisors of intermediate size. For example, suppose we choose a group order $p \approx 2^{256}$ to achieve 128-bit security against the Pollard-Rho attack, which runs in time approximately $\sqrt{\pi p/2}T_p$ [34, 47]. Analyzing the proof of Theorem 5.3.2 more closely, we find that the private key of the full Boneh-Boyen signature scheme can be obtained in time approximately $4\sqrt{p/d}T + 2d^2T_p$ if d is a divisor of $p-1$. Thus, assuming that $T \approx (2\log_2 p)T_p$ with the ordinary square and multiply algorithm for exponentiation,

we find that roughly speaking $p-1$ should not have a divisor between 23 and 64 bits long if a 128-bit security level is desired. For the $p+1$ algorithm the constants in the running time are much harder to determine, so we omit a detailed analysis of this case. Nevertheless, we expect that divisors of similar size should also be avoided for $p+1$. Primes of this form are rare, especially in the context of bilinear group pairs, but they are not impossible to find. In such cases, Cheon's algorithms no longer apply, and Theorems 5.3.1 and 5.3.2, while still valid, have no practical impact in the absence of a faster algorithm for solving q-SDH.

Another possibility is to modify the signature scheme in such a way that the proof of security under q-SDH still holds, but our reduction in the converse direction does not. We observe that our reduction algorithms rely on the fact that the denominator in the exponent $g^{\frac{1}{x+m+yr}}$ is linear in both m and r. Accordingly, it would be natural to look for signature schemes with nonlinear denominators whose security can still be proven under the q-SDH assumption. One example of such a signature scheme is given by Wei and Yuen [57], and another example is the scheme $\sigma \leftarrow (g_1^{\frac{1}{x+mr+yr^2}}, r)$. We emphasize that we have not checked the security proofs for any of these modified schemes, nor have we made any systematic effort to examine whether our techniques for proving the converse can be extended to such schemes.

Chapter 6

Conclusions and Future Work

This thesis presents the issues related to Boneh-Boyen signatures and the strong Diffie-Hellman problem on which the security of the Boneh-Boyen signature schemes rest. We present several intractability assumptions related to SDH, and several signature schemes related to Boneh-Boyen signatures. Our contribution includes showing that the existential forgery of signatures for both the basic and full versions of the Boneh-Boyen signature scheme can be reduced to the q-SDH problem via an algorithm which is quadratic in q. This result establishes the equivalence of the q-SDH assumption and the security of Boneh-Boyen signatures, thus resolving an open problem posed in [19, 37]. Together with Cheon's q-SDH analysis, the reduction algorithm allows us to recover Boneh-Boyen private keys in time $O(p^{\frac{2}{5}+\varepsilon})$ for groups of order p whenever $p \pm 1$ satisfies certain divisibility properties. There are several topics that are worthy of further study, some of which we mention below:

Modification to Boneh-Boyen Signatures. We discussed in Section 5.4 one possible way to modify Boneh-Boyen signatures to prevent our attack. Proving the security of our proposed scheme under the SDH (SDH$'$) assumption remains an open question. We are also interested in other possible modifications that would accomplish this goal.

Non-generic Algorithms. All the algorithms that we have seen in this thesis are generic algorithms. It is interesting to look for an efficient non-generic algorithm that solves the SDH (SDH$'$) problem or that forges Boneh-Boyen signatures implemented in certain groups. Such an algorithm might beat the lower bound presented in Theorem 2.3.3.

SDH Based Schemes. It may be worthwhile to look for other schemes based on SDH or related assumptions for which no reduction to the SDH problem is known, and either to prove the equivalence to the assumption, or to find a weaker assumption on which the security of such a scheme can rest.

Extending Cheon's Algorithm. We discussed Cheon's $p-1$ and $p+1$ algorithms in Section 5.2. Notice that $p-1$ and $p+1$ are the first and the second cyclotomic polynomial at p. It is worth investigating whether the algorithm can be extended to work with divisors of cyclotomic polynomials at p other than $p \pm 1$. If the algorithm can be adapted to higher degree cyclotomic polynomials in such a way that the overall algorithm outperforms Pollard rho, then one would have more options for solving SDH on a given group.

References

[1] FIPS 186. *Digital Signature Standard (DSS)*. National Institute for Standards and Technology, 1994. `http://www.itl.nist.gov/fipspubs/fip186.htm`. Accessed December 2008. 1, 4

[2] FIPS 186-2. *Digital Signature Standard (DSS)*. National Institute for Standards and Technology, 2000. `http://csrc.nist.gov/publications/PubsFIPS.html`. Accessed December 2008. 1, 4

[3] FIPS 186-3. *DRAFT, Digital Signature Standard (DSS)*. National Institute for Standards and Technology, 2008. `http://csrc.nist.gov/publications/PubsDrafts.html`. Accessed December 2008. 1

[4] Michael Artin. *Algebra*. Prentice Hall, United States edition, 1991. ISBN: 013-004763-5. 41

[5] Giuseppe Ateniese and Breno de Medeiros. A provably secure Nyberg-Rueppel signature variant with applications. Cryptology ePrint Archive, Report 2004/093, 2004. `http://eprint.iacr.org/2004/093`. Accessed December 2008. 4

[6] Joseph Bak and Donald J. Newman. *Complex Analysis*. Springer, 2nd edition, 1996. ISBN: 038-794756-6. 41

[7] Feng Bao, Robert H. Deng, and HuaFei Zhu. Variations of Diffie-Hellman problem. In *Information and Communications Security*, volume 2836 of *LNCS*, pages 301–312, 2003. 7, 14

[8] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In *Advances in Cryptology—CRYPTO 1989*, volume 435 of *LNCS*, pages 547–557. Springer, 1990. 7

[9] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993. 1, 5, 21

[10] Raghav Bhaskar, Karthekeyan Chandrasekaran, Satyanarayana Lokam, Peter L. Montgomery, Ramarathnam Venkatesan, and Yacov Yacobi. An observation about variations of the Diffie-Hellman assumption. Cryptology ePrint

Archive, Report 2008/525, 2008. http://eprint.iacr.org/2008/525. Accessed January 2009. 7, 14

[11] Dan Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory (Portland, OR, 1998)*, volume 1423 of *LNCS*, pages 48–63. Springer, 1998. 7

[12] Dan Boneh and Xavier Boyen. Efficient selective-ID identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, 2004. 7, 9, 10, 13, 31

[13] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology—EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, 2004. 1, 2, 3, 5, 6, 7, 11, 21, 23, 25, 26, 27, 35, 46

[14] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008. iii, 1, 2, 3, 5, 6, 7, 11, 16, 17, 21, 23, 25, 26, 27, 35, 36, 38, 42, 44, 46

[15] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. 4, 9, 31, 32, 34

[16] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography. In *Advances in Cryptology—CRYPTO 1996*, volume 1109 of *LNCS*, pages 283–297. Springer, 1996. 7

[17] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology—ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532, Berlin, 2001. Springer. 4, 8, 25, 28

[18] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004. 4, 8, 28, 29

[19] Xavier Boyen. The uber-assumption family – a unified complexity framework for bilinear groups. In *2nd International Conference on Pairing-based Cryptography—PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer-Verlag, 2008. 6, 7, 9, 10, 11, 13, 14, 15, 21, 22, 59

[20] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography—PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer, 2007. 14, 15

[21] Stefan A. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI (Centre for Mathematics and Computer Science), 1993. 8

[22] James Ward Brown and Ruel V. Churchill. *Complex Variables and Applications*. McGraw-Hill, Seventh edition, 2004. ISBN: 007-287252-7. 41

[23] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in Cryptology—CRYPTO 1997*, volume 1294 of *LNCS*, pages 455–469. Springer, 1997. 7

[24] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004. 5, 21

[25] Jung Hee Cheon. Security analysis of the Strong Diffie-Hellman problem. In *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–13. Springer-Verlag, 2006. iii, 6, 14, 40, 49

[26] Jean-Sébastien Coron. On the exact security of full domain hash. In *Advances in Cryptology—CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, 2000. 4

[27] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *Advances in Cryptology—ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, 2002. 21

[28] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976. 1, 7

[29] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography—PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer-Verlag, 2005. 7, 13

[30] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. 1

[31] Marc Fischlin. A note on security proofs in the generic model. In *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 458–469. Springer, 2000. 21

[32] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. 4, 17

[33] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. 2

[34] Bernard Harris. Probability distributions related to random mappings. *Annals of Mathematical Statistics*, 31:1045–1062, 1960. 57

[35] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, 2004. 9

[36] Neal Koblitz and Alfred Menezes. Another look at generic groups. *Advances in Mathematics of Communications*, 1(1):13–28, 2007. 5, 21, 22

[37] Neal Koblitz and Alfred Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. Cryptology ePrint Archive, Report 2007/442, 2007. `http://eprint.iacr.org/2007/442`. Accessed December 2008. 5, 59

[38] Ueli M. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In *Advances in Cryptology—CRYPTO 1994*, volume 839 of *LNCS*, pages 271–281. Springer, 1994. 7

[39] Ueli M. Maurer and Stefan Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 1999. 7

[40] Ueli M. Maurer and Stefan Wolf. The Diffie-Hellman protocol. *Designs, Codes and Cryptograhy*, 19(2-3):147–171, 2000. 7, 8

[41] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–484, 2002. 7, 13, 14, 15, 40

[42] David Naccache and Jacques Stern. Signing on a postcard. In *Proceedings of Financial Cryptography 2000*, pages 121–135. Springer, 2001. 4

[43] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, 2004. 7

[44] Kaisa Nyberg and Rainer A. Rueppel. Message recovery for signature schemes based on the discrete logarithm. *Designs, Codes and Cryptograhy*, 7(1-2):61–81, 1996. 4

[45] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In *Proceedings of TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer-Verlag, 2006. 15, 16, 30, 31

[46] Leon A. Pintsov and Scott A. Vanstone. Postal revenue collection in the digital age. In *Proceedings of Financial Cryptography 2000*, pages 105–120. Springer, 2001. 4

[47] John M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978. 6, 49, 50, 57

[48] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT-LCS-TR-212, Massachusetts Institute of Technology, 1979. 1

[49] Ron L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 1

[50] Claus P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. 1

[51] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology—CRYPTO 1984*, volume 196 of *LNCS*, pages 47–53. Springer, 1985. 32

[52] Daniel Shanks. Class number, a theory of factorization and genera. In *1969 Number Theory Institute*, volume 20 of *Proceedings of Symposia in Pure Mathematics*, pages 415–440. American Mathematical Society, 1971. 49, 50

[53] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology—EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997. 16

[54] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pages 31–37. ACM, 1996. 7

[55] Douglas R. Stinson. *Cryptography Theory and Practice*. Chapman & Hall/CRC, Third edition, 2005. 32

[56] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005. 29, 30, 34

[57] Victor K. Wei and Tsz Hon Yuen. More short signatures without random oracles. Cryptology ePrint Archive, Report 2005/463, 2005. `http://eprint.iacr.org/2005/463`. Accessed December 2008. 12, 58