Executive Summary

This dissertation is concentrated on the research of efficiency analysis of block-wise P-signatures, and the objective of this project is to investigate possible techniques that can be used to reduce the running time cost of block-wise P-signatures. Overall, this project contains both theoretical investigation and practical implementation, as well as efficiency analysis.

Block-wise P-signatures were recently proposed as a new primitive to achieve non-interactive anonymous credentials with efficient attributes, which could be used in a variety of contexts such as e-Commerces and e-Voting systems. Nonetheless, the introduced instantiation of block-wise P-signatures are based on symmetric pairings, which is regarded to be inefficient and impractical in implementation. Motivated by this observation we engage in investigating possible techniques to enhance performance of block-wise P-signatures.

We have received significant progress to improve the efficiency of block-wise P-signatures both in theoretical algorithm design and in practical implementation. Our adopting techniques mainly include reconstructing block-wise P-signatures based on asymmetric pairings, as well as adopting pre-computation of pairing elements to speed up pairing computation. The major achievements are described as follows.

1. We firstly propose block-wise P-signatures based on asymmetric pairings, which can be proven secure against forgery attacks, see pages 31-40.

2. Our proposed block-wise P-signatures are theoretically efficiency while comparing with Izabachène et al.'s original construction, see pages 40-43.

3. We both implement block-wise P-signatures based on symmetric pairings and ones based on asymmetric pairings using MIRACL Crypto. SDK, and our proposed schemes achieve enormous efficiency improvement, see pages 43-46.

4. Based on our proposed schemes, we apply technique of pairing element alignment and propose an optimized block-wise P-signature scheme, which is also proven secure against forgery attacks. see pages 47-57.

5. We successfully speed up optimized block-wise P-signatures by adopting techniques of pairing pre-computation and elements multiplication pre-computation, see pages 57-60.

6. We evaluate the performance of our proposed schemes on five pairing curves and give a comprehensive comparison among them, see pages 61-66.

Contents

List of Figures

Chapter 1

Introduction

Credential systems play important roles in modern society for people to demonstrate the validity of requested activities. Nowadays an variety of paper-based credentials have been used in our daily life, such as drivers' licences, passports, and even tickets for entering galleries. In electronic world, digital credentials such as national identification cards and Oyster cards have also been the essential tools that people use everyday. To apply for credentials, in general, people are required to register themselves to trusted organizations, then credentials will be issued if applicants are qualified. Credentials which issued by organizations explicitly shows the applicants' personal information such as name, birthdate or home address, as well as the statement of allowed activities. With the issued credentials, people are allowed to do some activities identified in the credentials.

Traditionally a simple digital credential system can be constructed on digital signatures. Without loss of generality, suppose an applicant has registered to a credential issuer and both two parties use public-key system. Firstly one applicant sends a digital signature containing a message to be authorized to one credential issuer. While receiving the application, the credential issuer checks validity of the applicant's message by verifying the received digital signature and then produces a new digital signature which contains the applicant's public key information and the applied message. The new digital signature signed by the credential issuer is then sent back to the applicant. Afterwards, the applicant can present the issued digital signature to a third party and claim the ownership and the validity of underlying message by verifying the digital signature.

In digital world the above context of digital credential system shows two evident security flaws. Firstly it is clear that both in the credential applying stage and in the credential proving stage, the message is required to presented in a plaintext form, which is not desirable if the underlying message contains private information. Secondly, digital credentials are replicatable hence a malicious attacker can easily get a replication of one valid digital credential and play as an imposter on the Internet. Moreover, some further applications of digital credentials, such as e-cash and digital transactions, can also concern the linkage issue and the reuse issue of credential proofs. Motivated by the issues of privacy and anonymity, digital credential systems need to be further reconsidered.

Anonymous credentials

Anonymous credentials system is one of many solutions to address privacy and anonymity issues of digital credentials. Anonymous credentials, which are firstly introduced by Chaum [6], have been widely investigated by cryptographers and received a significant progress [7, 8, 9, 10, 11, 12, 13, 14, 15, 4]. The introduced anonymous credentials systems are essentially constructed on a variety of digital signatures with two protocols. The first protocol is designed to proceed users' credential requests in an anonymous manner while users try to submit both their secret keys and commitments on requested messages to credential issuers. After operating the first protocol, users can get credentials and reveal neither key information nor requested messages to other protocol participants nor malicious eavesdroppers. In terms of the second protocol, it is designed for users to prove the knowledge of ownership on credentials to third parties. For example, assume an user possesses a credential containing a signature (m, σ), he can firstly reveal a commitment c on the message m and then demonstrate his possession of (c, σ) using *zero-knowledge proofs* [16], so that third parties cannot learn any information about the message m. Furthermore, anonymous credential systems also provide the property of *credential-unlinkability*, which means that eavesdroppers learn nothing even multiple proofs are connected to the same credential.

Anonymous credentials are theoretically feasible, however, the majority of instantiations are impractical due to the inefficiency of interactive zero-knowledge proofs [16]. Traditional interactive zero-knowledge proofs are significantly constructed on NP-complete problems such as graph isomorphism problems or circuit satisfiability problems. To achieve sufficient security degree, NP-reduction mechanism may take an enormous amount of interactions between provers and verifiers, which is not practical for deployment in real applications. Although interactive zero-knowledge proofs can be replaced with non-interactive ones using *Fiat-Shamir paradigm* [17] and *Random Oracle Model* [18]; however, this approach may probably show security flaws while replacing random oracle with real hash functions [19, 20].

P-signatures

In 2008, Groth and Sahai [3] suggested a new non-interactive zero-knowledge proof system (NIZK), which firstly provides an effective proof mechanism to address the efficiency issue of zero-knowledge proofs. Not like NP-reduction methodologies, Groth-Sahai NIZK proofs adopt idea of *group-dependent language*, which is to to conceal the elements of one cyclic group by transferring them into another group, and then provide a mathematical relation to prove the connectivity of these two cyclic groups. Although the new NIZK proofs can be only used in pairing-based cryptographic primitives, Groth-Sahai NIZK proofs have provided an opportunity for constructing practical anonymous credential systems.

With the help of Groth-Sahai proofs, P-signatures are introduced to provide solutions to construct anonymous credentials. P-signatures, which stands for *signatures with efficient Protocols*, are firstly introduced by Belenkiy et al. [13] in 2008 and further extended

to support multi-block messages [14]. P-signatures can be easily and practically extended to implement a variety of anonymous credential systems. In general, one P-signature scheme comprises a F-unforgeable signature scheme, a perfect hiding commitment scheme, and three protocols:

1. An interactive protocol for users to request signatures on commitments of messages from signature signers.
2. A non-interactive proof for users to prove the validity of signatures without revealing the underlying messages.
3. A non-interactive protocol for users to prove the equality of two proofs on the same commitments with different opening values.

1.1 Implementation of block-wise P-signatures

General P-signatures schemes have made a significant achievement in providing efficient non-interactive protocol to anonymous credentials; but, their designs can be only deployed in general applications based on public-key-centric system. Traditional public-key-centric systems assume users maintain unique secret keys to represent their digital identities. However, in real life some emerging web technologies including cloud applications, have turned to describe web resources or network elements in an attribute-based methodology, such as eXtensible Markup Language (XML). Not like general messages, it is evident that the representation of XML is well-structured and can be further processed to fit in with different applications. Cryptographers [12, 21] have tried to construct attribute-based anonymous credential systems, but their constructions could not provide efficient non-interactive proof. To address this issue, there are three critical points P-signatures should be considered. It is obvious that the first point is to construct attribute-based P-signatures. The second consideration is to make attributes in P-signatures be capable of doing some simple logic calculation; for example, deciding if some set of attributes are contained in the signatures. And the last point is to keep the availability of non-interactive proofs of P-signatures.

In 2011, Izabachène et al. [4] proposed a powerful cryptographic primitive called block-wise P-signatures, to be the response to the above three considerations. Block-wise P-signatures not only further improve the efficiency of NIZK proofs of general P-signatures; more importantly, block-wise P-signatures can support anonymous credentials with efficient attributes. Therefore, block-wise P-signatures may hold high potential to be the building blocks of cloud credential applications in the future. Due to the importance of block-wise P-signatures, in our project, we concentrate on the implementation issue of block-wise P-signatures. An observation on the implementations of block-wise P-signatures is that they heavily rely on bilinear-pairings computation. Bilinear-pairings computation may show inefficient and impractical consequence if the underlying pairings computation is not treated well. Motivated by this issue, in this dissertation, we concentrate on improving the efficiency of block-wise P-signatures.

1.2 Scope of this dissertation

This dissertation aims to comprehensively analyze the efficiency of block-wise P-signatures. The objective of our works is not only on reviewing and discussing the efficiency of block-wise P-signatures; more importantly, we concentrate on improving the performance of block-wise P-signatures with possible optimization techniques. In Chapter 2, we introduce the essential mathematical knowledge and security assumptions of bilinear pairings that will be used in this dissertation. In addition, we also review several optimization techniques that could be used to enhance the efficiency of pairing-based cryptographic schemes. Then in Chapter 3, we review three introduced P-signature schemes, which are single-block P-signatures, multi-block P-signatures, and most importantly, block-wise P-signatures. In Chapter 4, we propose a new block-wise P-signature scheme based on asymmetric pairings and analyze both the theoretical performance and the practical performance between our proposed scheme and Izabachène et al.'s construction. In Chapter 5, we apply optimization techniques into our scheme, and an optimized block-wise P-signature scheme is proposed. Moreover, the further efficiency analysis of applying optimization techniques in block-wise P-signatures is also given in this chapter. Lastly in chapter 6, we evaluate our works and suggest some future works that could further increase the performance of block-wise P-signatures.

Chapter 2

Pairing-based Cryptography

Bilinear pairings have been one of most important research topics in modern cryptography since they can be used for constructing a variety of cryptographic primitives that cannot be effectively solved in general public-key system. Bilinear pairings are essentially constructed on elliptic curves and define functions that map a pair of elliptic curve points to a finite field. In the past decade cryptographers utilized bilinear pairings to construct a variety of cryptographic primitives and protocols. In this chapter we introduce the essential knowledge of bilinear pairings, as well as some important security assumptions adopted by P-signatures schemes.

2.1 Bilinear pairings

Without loss of generality, let p be a prime integer. Suppose $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ be cyclic groups of order p and can be expressed in multiplicative notation with identity element 1. A bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ can be defined as a function

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$$

with three properties.

1. **Bilinearity:** For all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, and $x, y \in \mathbb{Z}$, $e(u^x, v^y) = e(u, v)^{xy}$
2. **Non-degeneracy:** For all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ be the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, $e(g_1, g_2)$ generates $\mathbb{G}_T$.
3. **Computability:** There exits an probabilistic polynomial time algorithm to generate bilinear map $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, and for any $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, it is efficient to compute $e(u, v)$.

Types of pairing implementations

From the bilinear map function $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow G_T$, it describes the pair association of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ with elements in G_T; hence this technique is called *bilinear pairings*

or *pairings* for short. In general, according to the constructions of bilinear maps, there are three types of pairing implementations on elliptic curves, which are:

Type-1 pairings: It describes a special form of bilinear maps where $\mathbb{G}_1 = \mathbb{G}_2$, so the pairing expression can be written as $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. Such the symmetric pairing constructions are widely used in constructing pairing-based cryptographic schemes because of their concise expressions.

Type-2 pairings: This type of pairing constructions $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow G_T$ illustrate that there exits an efficiently computable homomorphism ϕ from $\mathbb{G}_2$ to $\mathbb{G}_1$.

Type-3 pairings: This kind of pairing constructions define $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ where $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exits no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$.

In fact, even in Type-3 pairings, it is evident that there exists homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ since $\mathbb{G}_1$ and $\mathbb{G}_2$ are inherently cyclic groups of the same order. However, computing these homomorphisms may be computationally infeasible. This is the major difference between Type-2 pairings and Type-3 pairings, and up to now there is few implementation of Type-2 pairings. Among theses three types of pairings, the research of Galbraith et al. [1] shows Type-1 pairings and Type-3 pairings may be the most common and practical pairing implementations. Moreover, The research of Scott [2] further suggests that Type-3 pairings can show more flexibility and efficiency than Type-1 pairings. In this dissertation, we use the terms *symmetric pairings* and *asymmetric pairings* to respectively describe Type-1 pairings and Type-3 pairings, which can clearly distinguish these two types of pairings from the view of construction difference.

2.2 Security assumption

In modern cryptography, in order to theoretically measure the security degree of cryptographic schemes, it is a common methodology to reduce one cryptographic scheme to one known hard problem, such as *factoring problem*, *square root problem*, or *discrete-logarithm problem*. Based on these hard problems, cryptographers have built a variety of security models for measuring the security of proposed schemes. In terms of pairing-based cryptography, the majority of proposed schemes are based on well-know *Diffie-Hellman problem*. Here we introduce the essential security assumptions of pairings that will be used in this dissertation.

Group-related security assumptions.

Definition 2.1. ***Computational Diffie-Hellman Problem (CDH)***
Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Given $a = g^x$ and $b = g^y$ where $x, y \in \mathbb{Z}_p^*$, computer $c \in \mathbb{G}$ such that $c = g^{xy}$.

Definition 2.2. ***Decisional Diffie-Hellman Problem (DDH)***
Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Given $a = g^x$, $b = g^y$ and $c = g^z$ where $x, y, z \in \mathbb{Z}_p^*$, determine if $z = xy$.

Definition 2.3. ***Decisional Linear Problem (DLIN) [22]***
Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Given $(g, g^x, g^y, g^{rx}, g^{sy}, g^{r+s})$ and $(g, g^x, g^y, g^{rx}, g^{sy}, g^z)$ where $x, y, r, s, z \in \mathbb{Z}_p^*$, determine if $g^z = g^{r+s}$.

Definition 2.4. ***q-Hidden Strong Diffie-Hellman Problem (q-HSDH) [24]***
Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Given $g, u, g^x \in \mathbb{G}$, and $\{g^{1/(x+c_i)}, g^{c_i}, u^{c_i}\}_{i=1...q}$ where $x, c_1, ..., c_q \in \mathbb{Z}_p^*$, output a new tuple $(g^{1/(x+c)}, g^c, u^c)$.

Definition 2.5. ***Flexible Diffie-Hellman Problem (FlexDH) [4]***
Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Given (g, g^x, g^y) where $x, y \in \mathbb{Z}_p^*$, find a triple $(g^\mu, g^{\mu x}, g^{\mu xy})$ such that $\mu \neq 0$.

Definition 2.6. ***n-Diffie-Hellman Exponent Problem (n-DHE) [25]***
Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Given $(g, g_1, ..., g_n, g_{n+2}, ..., g_{2n})$ such that $g_i = g^{(\alpha^i)}$ where $\alpha \in \mathbb{Z}_p^*$ and $i \in [1, 2n] \setminus \{n+1\}$, compute element $g_{n+1} = g^{(\alpha^{n+1})}$.

Definition 2.7. ***Flexible n-Diffie-Hellman Exponent Problem (n-FlexDHE) [4]***
Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Given $(g, g_1, ..., g_n, g_{n+2}, ..., g_{2n})$ such that $g_i = g^{(\alpha^i)}$ where $\alpha \in \mathbb{Z}_p^*$ and $i \in [1, 2n] \setminus \{n+1\}$, compute a triple $(g^\mu, g_{n+1}^\mu, g_{2n}^\mu)$ where $\mu \in \mathbb{Z}_p^*$ and $g_{n+1} = g^{\alpha^{n+1}}$.

Symmetric pairings-related security assumptions.

Definition 2.8. ***Bilinear Diffie-Hellman Problem (BDH)***
BDH problem is defined in Type-1 pairings. Let bilinear map $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ and $g \in \mathbb{G}$ be the generator of $\mathbb{G}$. Given $a = g^x$, $b = g^y$ and $c = g^z$ where where $x, y, z \in \mathbb{Z}_p^*$, compute $e(g, g)^{xyz}$.

Asymmetric pairings-related security assumptions.

Definition 2.9. ***Symmetric External Diffie-Hellman Problem (SXDH) [23]***
SXDH problem is defined in Type-3 pairings. Let bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, SXDH problem assumes DDH problem is hard both in $\mathbb{G}_1$ and $\mathbb{G}_2$.

Definition 2.10. ***Triple Diffie-Hellman Problem (TDH) [13]***
TDH problem is defined in Type-3 pairings. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of order p and define bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. Given $g_1, g_1^x, g_1^y \in \mathbb{G}_1$, $g_2, g_2^x \in \mathbb{G}_2$, and $\{c_i, g_1^{1/(x+c_i)}\}_{i=1...q}$ where $x, y, c_1, ..., c_q \in Z_p^*$, output a tuple $(g_2^{\mu x}, g_1^{\mu y}, g_1^{\mu xy})$ for some $\mu \in \mathbb{Z}_p$.

Figure 2.1: Suggested key size of different style of cryptographic primitives [1]

Author	κ	ECC-style	RSA-style
NIST[26]	80	160	1024
	128	256	3072
	256	512	15360
Lenstra[27]	80	160	1329
	128	256	4440
	256	512	26268
ECRYPT[28]	80	160	1248
	128	256	4440
	256	512	15424

2.3 Practical issues of pairing implementations

Bilinear pairings have been utilized to construct a variety of applications that traditional public-key system cannot provide; however, the majority of proposed pairing-based cryptographic schemes concentrate on theoretical investigation rather than considering the practical issue of pairing implementations. In reality, implementations of bilinear pairings are significantly restricted since it is unfortunately evident that a few of elliptic curves can be used for constructing bilinear pairings. Moreover, even one pairing-based scheme can be successfully constructed, the computational cost may be extremely high and hence cannot be practically deployed. Seeing this problem, in this section, it is necessary to discuss the feasibility of implementing different types of pairing constructions, as well as the techniques that we can used to improve the efficiency of pairing-based schemes.

The implementation of bilinear pairings contains complex point computation on elliptic curves. Let e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ be a bilinear map where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order p, and let $e(P,Q)$ where $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$ be the expression of pairing computations. In general, $\mathbb{G}_1$ is constructed as a subgroup of $E(\mathbb{F}_q)$ where E is a elliptic curve and q is the field size of one field $\mathbb{F}$. Similarly, $\mathbb{G}_2$ is constructed as a subgroup of $E(\mathbb{F}_{q^k})$, which is k-th extension of finite field $\mathbb{F}_q$ and k is also called *embedding degree*. Lastly $\mathbb{G}_T$ is a subgroup of $E(\mathbb{F}^*_{q^k})$.

Galbraith et al. [1] comprehensively discuss the implementation issue of bilinear pairings. Here we review several points that are critical for designing pairing-based schemes. Firstly it is pivotal to realize that the efficiency of bilinear pairings are more like traditional public-key system, rather than elliptic curves. Please refer to Table 2.1, which is the suggested key size of ECC-style and RSA-style for achieving security degree κ by NIST [26], Lenstra [27], and ECRYPT [28]. Suppose we want to construct a bilinear pairing satisfying security degree $\kappa = 256$ of NIST, we can choose $E(\mathbb{F}_q)$ of ECC-style 512 as the cyclic group $\mathbb{G}_1$, and set the size of $\mathbb{G}_2 = E(\mathbb{F}_{q^k})$ at lease greater than 15360 bits. It is clear the bottleneck of this bilinear pairing function is on the computation of $E(\mathbb{F}_{q^k})$,

which is like RSA-style system, rather than efficient elliptic curves. This fact illustrates why the majority of proposed pairing-based schemes may show inefficiency in practical implementations.

Secondly, it is not always true for pairing-based schemes to find suitable elliptic curves. Please continue the above example, if we want to choose $E(\mathbb{F}_{q^k})$ of group size p, given q be 512 bits, the embedding degree k will be $15360/512 = 30$. However, currently we still have no way to construct such elliptic curves with embedding degree 30. This fact significantly shorten the feasibility of implementing some pairing-based schemes.

According to the collection and analysis of known bilinear pairings in [1], up to now only Type-1 pairings and Type-3 pairings can be practically implemented. Basically Type-3 pairings can provide flexible choice for implementing pairing-based schemes without any limitation. Type-1 pairings, $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$, however, are usually constructed on elliptic curves over $\mathbb{F}_p$, $\mathbb{F}_{2^d}$ and $\mathbb{F}_{3^d}$ with embedding degree 2 ,4 and 6. The significant feature of Type-1 pairings is that group elements are commutable, e.g. $e(P,Q)$=$e(Q,P)$, which simplify the design of pairing-based schemes. However, Type-1 pairings show two shortcomings. Firstly to defend Pollard-rho attack on elliptic curve fields, the group size of elliptic curves should be set to larger than 256 bits, which is equivalent to $\kappa = 128$. Fortunately Type-1 pairings can find suitable elliptic curves for satisfying this security degree; however, in terms of higher security degree of $\kappa = 192$ or $\kappa = 256$, Type-1 pairings show no efficient implementations. Secondly, while doing one pairing calculation, Type-1 pairings take more group elements and more group operations than Type-3 pairings take. Therefore, on the consideration of flexibility and efficiency of bilinear pairings, Type-3 pairings show more advantage than Type-1 pairings.

2.4 Optimization of pairing-based cryptography

Despite flexibility of Type-3 pairings, computational cost of bilinear pairing is still high. To address this issue, cryptographers make great efforts on optimizing pairing computation. Scott [2] concludes two techniques that could be used for pairing optimizations.

The first technique is called *fixed argument optimization*, firstly introduced in [29] and further discussed in [30], which discusses the advantage of fixed argument in pairing calculation. If both P and G are fixed arguments, the pairing calculation $e(P,Q)$ can be precomputed and stored. And if $Q \in \mathbb{G}_2 = E(\mathbb{F}_{q^k})$ is fixed, then the possible value of Q^x can be precomputed for further use. However, fixed argument $P \in \mathbb{G}_1 = E(\mathbb{F}_q)$ cannot bring any advantage for pairing optimization. Fixed argument optimization can be deployed both in Type-1 pairings and in Type-3 pairings.

The second technique aims to accelerate the calculation of products of pairings.

$$t = \prod_{i=1}^{n} e(P_i, Q_i)$$

Products of pairings is common in pairing-based cryptographic schemes and the optimization technique is firstly introduced by Scott [29] and further enhanced by Granger and

Smart [31]. This optimization focus on sharing the same variable m and final exponentiation in Miller's algorithm (Figure 2.2). However, pairing product computation may not always appear in pairing-based schemes; hence the deployment of this technique is limited.

Figure 2.2: Miller's algorithm [2]

Algorithm 1 Computation of basic ate pairing $e(P, Q)$ using Miller's algorithm

INPUT: $P \in E(\mathbb{F}_{p^d}), Q \in E(\mathbb{F}_p)$, trace t, even k, where P has order r
OUTPUT: $e(P, Q)$
1: $T \leftarrow P$, $m \leftarrow 1$
2: $n \leftarrow t - 1$
3: **for** $i \leftarrow \lfloor \lg(n) \rfloor - 1$ **downto** 0 **do**
4: $\quad m \leftarrow m^2.l_{T,T}(Q)$
5: $\quad T \leftarrow 2T$
6: $\quad$ **if** $n_i = 1$ **then**
7: $\quad\quad m \leftarrow m.l_{T,P}(Q)$
8: $\quad\quad T \leftarrow T + P$
9: $\quad$ **end if**
10: **end for**
11: $m \leftarrow m^{(p^k-1)/r}$
12: **return** m

Chapter 3

Review of P-signature Schemes

P-signatures, which stand for signatures with efficient protocols, are firstly introduced by Belenkiy et al. [13] in 2008 and are regarded as important building blocks for many applications in modern cryptography. Generally A P-signature scheme comprises:

1. A commitment scheme for users to hide their messages to be signed.

2. A signature scheme for signature signers to sign received commitments.

3. An interactive protocol for users to acquire signatures from signature signers.

4. A non-interactive proof for users to prove the validity of signatures to third parties without revealing the underlying messages.

5. A non-interactive proof for users to prove two different commitments are commitments to the same message.

In this chapter, we will review three P-signature schemes, which are single-block P-signature schemes [13], multi-block P-signature schemes [14], and block-wise P-signature schemes [4] respectively. Moreover, we also review Groth-Sahai proof systems [3], which are the essential tool for all P-signatures to construct non-interactive proofs.

3.1 Groth-Sahai non-interactive proof systems

Traditionally interactive zero-knowledge proofs are constructed on well-known NP-problems such as *circuit satisfiability*, which take enormous time to complete a proof of knowledge. Although interactive zero-knowledge proof can be transferred to non-interactive proof by applying Fiat-Shamir paradigm [17] and random oracle model [18], this methodology may lead proofs to be insecure [19, 20] while replacing random oracle with a real hash function. Recently Groth and Sahai [3] found an extraordinary idea to construct non-interactive system based on *group-dependent language* and common reference string (CRS), which provides a secure and efficient way to construct non-interactive proof systems. Basically, Groth-Sahai non-interactive proof systems possess *witness indistinguishability* and can be

further transferred to zero-knowledge proofs. In fact, the construction of P-signatures heavily rely on Groth-Sahai commitment schemes and proof systems; therefore, to better understand P-signatures it is necessary to review these two parts of Groth-Sahai proof systems. The original Groth-Sahai non-interactive proof systems can be either implemented on Type-1 Pairings or Type-3 Pairings; in this section, we will focus on the version of Type-3 pairings.

Figure 3.1: Groth-Sahai proof system [3]

$$\begin{array}{ccccc} \mathbb{A}_1 & \times & \mathbb{A}_2 & \xrightarrow{f} & \mathbb{A}_T \\ \iota_1 \downarrow\uparrow \rho_1 & & \iota_2 \downarrow\uparrow \rho_2 & & \iota_T \downarrow\uparrow \rho_T \\ \mathbb{B}_1 & \times & \mathbb{B}_2 & \xrightarrow{F} & \mathbb{B}_T \end{array}$$

Essentially Groth-Sahai proofs utilize group-dependent language to construct proofs. Please refer to Figure 3.1, Groth-Sahai proofs define two bilinear pairings functions $f : \mathbb{A}_1 \times \mathbb{A}_2 \longrightarrow \mathbb{A}_T$ and $F : \mathbb{B}_1 \times \mathbb{B}_2 \longrightarrow \mathbb{B}_T$, a function $\iota : \mathbb{B} \longrightarrow \mathbb{B}$ and a trapdoor one-way function $\rho : \mathbb{B} \longrightarrow \mathbb{A}$, so that computation is efficient in ι but not trivial in ρ. The defined map should satisfy the following two properties:

$$\begin{aligned} \forall x \in \mathbb{A}_1, \forall y \in \mathbb{A}_2 : \quad & F(\iota_1(x), \iota_2(y)) = \iota_T(f(x, y)), \\ \forall \mathcal{X} \in \mathbb{B}_1, \forall \mathcal{Y} \in \mathbb{B}_2 : \quad & f(\rho_1(\mathcal{X}), \rho_2(\mathcal{Y})) = \rho_T(F(\mathcal{X}, \mathcal{Y})), \end{aligned} \tag{3.1}$$

To make a Groth-Sahai proof, common reference string (CRS) is required to be set up. In general, CRS is a set of elements defined in $\mathbb{B}_i$ and the variable setting is different in different type of pairings. In general we denote CRS by $\mathcal{U} = (\mathcal{U}_1, \mathcal{U}_2)$, which $\mathcal{U}_1 = \mathcal{U}_1^1$, ..., $\mathcal{U}_1^{m_1} \in \mathbb{B}_1$ and $\mathcal{U}_2 = \mathcal{U}_2^1$, ..., $\mathcal{U}_2^{m_2} \in \mathbb{B}_2$. To commit to an element $x \in \mathbb{A}_i$, one selects random vectors $\overrightarrow{r} = (r_1, ..., r_{m_i}) \in \mathbb{F}_q^{m_i}$ and computes

$$\begin{aligned} comm_i(x) \quad &= \iota_i(x) + \sum_{j=1}^{m_i} [r_j]\mathcal{U}_i^j \\ &= \iota_i(x) + \overrightarrow{r} \cdot \mathcal{U}_i \end{aligned} \tag{3.2}$$

Then suppose we want to construct a NIWI proof for the equation

$$a \cdot y + x \cdot b + x \cdot \Gamma y = t \tag{3.3}$$

where $f(x, y) = x \cdot y$, we firstly make commitments for x and y with Groth-Sahai commitment scheme.

$$\begin{aligned} c &= \iota_i(x) + \mathcal{R}\mathcal{U}_1 \\ d &= \iota_i(y) + \mathcal{S}\mathcal{U}_2 \end{aligned} \tag{3.4}$$

where $\mathcal{R}, \mathcal{S}$ are random vectors. Next one can pick $a \in \mathbb{A}_1^m$, $b \in \mathbb{A}_2^n$, and $H \in Mat_{m_2 \times m_1}$ an computes

$$\begin{aligned} \pi &= R^T \iota_2(b) + R^T \Gamma \iota_2(y) + R^T \Gamma S \mathcal{U}_2 - H^T \mathcal{U}_2 \\ \theta &= S^T \iota_1(a) + S^T \Gamma^T \iota_1(x) + H^T \mathcal{U}_1 \end{aligned} \tag{3.5}$$

The verification of this NIWI proof can be verified by checking whether the following equation holds

$$\iota_1(a) \bullet d + c \bullet \iota_2(b) + c \bullet \Gamma d = \iota_T(t) + \mathcal{U}_1 \bullet \pi + \theta \bullet \mathcal{U}_2 \tag{3.6}$$

Here $\mathcal{X} \bullet \mathcal{Y}$ represents the computation of $F(\mathcal{X}, \mathcal{Y})$.

Depending on different setting of group $\mathbb{A}_1$ and $\mathbb{A}_2$, there are four possible instantiations for constructing Groth-Sahai proofs.[1]

1. $\mathbb{A}_1 = \mathbb{G}_1$, $\mathbb{A}_2 = \mathbb{G}_2$, $f(P, Q) = e(P, Q)$: This setting is named *pairing-product equation.*
2. $\mathbb{A}_1 = \mathbb{G}_1$, $\mathbb{A}_2 = \mathbb{F}_q$, $f(P, y) = [y]P$: This setting is named *multi-scalar multiplication* in $\mathbb{G}_1$.
3. $\mathbb{A}_1 = \mathbb{F}_q$, $\mathbb{A}_2 = \mathbb{G}_2$, $f(x, Q) = [x]Q$: This setting is named *multi-scalar multiplication* in $\mathbb{G}_2$.
4. $\mathbb{A}_1 = \mathbb{F}_q$, $\mathbb{A}_2 = \mathbb{F}_q$, $f(x, y) = x \cdot y$: This setting is named *quadratic equation* in $\mathbb{F}_q$.

3.1.1 Groth-Sahai commitment schemes

Groth-Sahai commitment schemes consist of three functions, which are GSComSetup(.), GSCom(.) and GSOpen(.) respectively.

GSComSetup($p, \mathbb{G}_1, \mathbb{G}_2, g$). Taken groups $\mathbb{G}_1$ and $\mathbb{G}_2$ generated by generator g of prime order p, GSComSetup(.) outputs common reference string $params_{com}$.

GSCom($params_{com}, m, open$). Taken common reference $params_{com}$, a message $m \in \mathbb{G}_i$ to be committed, chosen a random value *open*, GSCom(.) produces commitment *comm*. To commit to an exponent $e \in \mathbb{Z}_p$ of a base $b \in \mathbb{G}_i$, GSCom(.) can be extended to GSExpCom($params_{com}$, b, e, *open*) where $comm =$ GSCom($params_{com}$, b^e, *open*).

GSOpen($params_{com}, comm, m, open$). Taken common reference $params_{com}$, the commitment *comm* to be verified, the message $m \in \mathbb{G}_i$, and the random value *open*, GSOpen(.) accepts if *comm* is the commitment of m. Opening *comm* generated by GSExpCom($params_{com}, b, exp, open$) can also be similarly done by checking GSOpen($params_{com}, comm, b^{exp}, open$).

[1] In DLIN assumption of Type-1 pairings, there are three possible instantiations for Groth-Sahai proofs since case 2 and case 3 are the same.

Groth-Sahai commitment schemes can be constructed based on SXDH problem where $\mathbb{G}_1 \neq \mathbb{G}_2$ or DLIN problem where $\mathbb{G}_1 = \mathbb{G}_2$ and the achievement is turn out to be perfect binding, computational hiding and extractable, one can get the detailed introduction in [3].

3.1.2 Groth-Sahai pairing inner product equation and its proof

Groth-Sahai proofs [3] construct *f-extractable non-interactive proof of knowledge* (NIPK) by providing inner product equation of bilinear maps. In Type-3 pairings, assume we have bilinear maps $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is the bilinear map function. Also we define $params_1$ and $params_2$ be the system parameters for $\mathbb{G}_1$ and $\mathbb{G}_2$ to construct Groth-Sahai commitments. [2] To construct NIPK proof Groth and Sahai introduce inner product equation for commitments of $\{c_i\}_{i=1...m}$ in $\mathbb{G}_1$ and $\{d_j\}_{i=1...n}$ in $\mathbb{G}_2$.

$NIPK\{((c_1 : x_1), ..., (c_m : x_m), (d_1 : y_1), ..., (d_n : y_n))$:

$$\prod_{j=1}^{n} e(a_j, y_j) \prod_{i=1}^{m} e(x_i, b_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(x_i, y_j)^{r_{ij}} = t\} \tag{3.7}$$

where $\{a_1, ..., a_n\} \in \mathbb{G}_1$, $\{b_1, ..., b_m\} \in \mathbb{G}_2$, $t \in \mathbb{G}_T$, and $\{r_{ij}\}_{i=1...m, j=1...n} \in \mathbb{Z}_p$.

Groth-Sahai witness-indistinguishable proofs

With the help of Groth-Sahai pairing inner product, a Groth-Sahai witness-indistinguishable proof can be constructed as follows.

GSSetup($params_{BM}$). Taken security parameter of bilinear maps $params_{BM}$, GSSetup(.) generates all necessary parameters $params_{GS}$ used in Groth-Sahai proofs, including $params_1$ for constructing $\mathbb{G}_1$, $params_2$ for constructing $\mathbb{G}_1$, and $params_\pi$ for constructing proofs.

GSProve($params_{GS}, s, (\{x_i\}_{1,...,m}, \{y_j\}_{1,...,n}, opens)$). Taken system parameters $params_{GS}$, statements s=$\{(c_1, ..., c_m, d_1, ..., d_n)$, equations $\}$, witnesses $\{ x_i \}_{1,...,m}$, $\{ y_j \}_{1,...,n}$, and opening values $opens$, output a proof π.

GSVerify($params_{GS}, \pi$). Taken a proof π, GSVerify(.) accepts if π is a valid proof.

GSExtractSetup($params_{BM}$). Taken security parameter of bilinear maps $params_{BM}$, GSExtractSetup(.) outputs a pair of auxiliary values (av_1, av_2) which is used for extract the contents of commitments.

GSExtract($params_{GS}, av_1, av_2, \pi$). GSExtract(.) extracts X= x_1, ..., $x_m \in \mathbb{G}_1$ and Y = y_1, ..., $y_n \in \mathbb{G}_2$ such that (X, Y) satisfies inner product equations and commitments.

[2] In Type-1 pairings, $\mathbb{G}_1 = \mathbb{G}_2$ and $params_1 = params_2$

proofs of equality of committed exponents

To prove equality of commitment on exponents, Groth and Sahai provide the following proving methods for exponents in different groups. .

Committed exponents in different groups. Let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, to construct a proof NIPK { $((c : \mathbb{G}_1^{\alpha}),(d : \mathbb{G}_2^{\beta}))$: $\alpha{=}\beta$ }, we can conduct Groth-Sahai equation proof NIPK { $((c : g_1^{\alpha}),(d : g_2^{\beta}))$: $\mathrm{e}(g_1^{\alpha}, g_2)\mathrm{e}(g_1^{-1}, g_2^{\beta})=1$ }.

Committed exponents in the same group. Let $g_1, u \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, to construct a proof NIPK { $((c_1 : g_1^{\alpha}),(c_2 : u^{\beta}))$: $\alpha{=}\beta$ }, we can conduct Groth-Sahai equation proof NIPK { $((c_1 : g_1^{\alpha}), (c_2 : u^{\beta}), (d : g_2^{\gamma}))$: $\alpha = \gamma \wedge \beta = \gamma$ }.

Zero-knowledge proofs. Let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, to construct a zero-knowledge proof NIZK { $((c_1 : g_1^{\alpha}),(c_2 : g_1^{\beta}))$: $\alpha{=}\beta$ }, we can conduct Groth-Sahai equation proof NIZK { $((c_1 : g_1^{\alpha}),(c_2 : g_1^{\beta}), (d : g_2^{\gamma}))$: $1 \wedge \mathrm{e}(g_1, g_2^{\gamma})\mathrm{e}(g_1^{-1}, g_2){=}1$ }.

3.1.3 SXDH-based instantiation of Groth-Sahai proofs

After grasping the general idea of Groth-Sahai proof system, here we provide a specific instantiation based on Type-3 pairings, which will be used in the reviewing P-signature schemes except for block-wise P-signatures. [3] In Type-3 pairings, Groth-Sahai proofs are constructed based on SXDH assumption. Here we follow the notion of Groth-Sahai proof provided by Ghadafi et al. [32], which can provide more clear understanding of SXDH-based instantiation.

CRS Setup. without loss of generality, let $\mathbb{B}_1 = \mathbb{G}_1^2$, $\mathbb{B}_2 = \mathbb{G}_2^2$ and $\mathbb{B}_T = \mathbb{G}_T^4$, we define

$$F = \begin{cases} \mathbb{B}_1 \times \mathbb{B}_2 & \longrightarrow \mathbb{B}_T \\ (X_1, Y_1), (X_2, Y_2) & \longrightarrow (e(X_1, Y_1), e(X_1, Y_2), e(X_2, Y_1), e(X_2, Y_2)) \end{cases}$$

In terms of CRS setting, for $i = 1, 2$, $a_i, t_i \in \mathbb{F}_q^*$, we prepare

$$Q_i = [a_i]P_i, \quad U_i = [t_i]P_i, \quad V_i = [t_i]Q_i$$

and set

$$\mathcal{U}_i^1 = (P_i, Q_i) \in \mathbb{B}_i$$
$$g(x,y)\mathcal{U}_i^2 = \begin{cases} [t_i]\mathcal{U}_i^1 & = (U_i, V_i) & \text{Binding case} \\ [t_i]\mathcal{U}_i^1 - (\mathcal{O}, P_i) & = (U_i, V_i - P_i)) & \text{Hiding case} \end{cases}$$

The CRS is $\{\mathcal{U}_1, \mathcal{U}_2\}$ where $\mathcal{U}_1 = \{\mathcal{U}_1^1, \mathcal{U}_1^2\}$ and $\mathcal{U}_2 = \{\mathcal{U}_2^1, \mathcal{U}_2^2\}$. We also define $\mathcal{W}_i = \mathcal{U}_i^2 + (\mathcal{O}, P_i) = (W_{i,1}, W_{i,2}) \in \mathbb{B}_i$.

[3] Block-wise P-signatures use DLIN-based instantiation of Groth-Sahai proofs due to the adoption of Type-1 pairings construction. Here we will not discuss DLIN-based instantiation of Groth-Sahai proof. Please refer to original paper [3] to see the detail setting DLIN-based instantiation.

ι_i, ρ_i and $comm_i$. Here we define the maps ι_i: $\mathbb{A}_i \longrightarrow \mathbb{B}_i$, ρ_i: $\mathbb{B}_i \longrightarrow \mathbb{A}_i$, and commitment scheme. There are two cases to be considered.
While $\mathbb{A}_i = \mathbb{F}_q$, the maps are defined as

$$\iota_i : \begin{cases} \mathbb{F}_q & \longrightarrow \mathbb{B}_i \\ x & \longrightarrow [x]\mathcal{W}_i \end{cases} \quad \rho_i : \begin{cases} \mathbb{B}_i & \longrightarrow \mathbb{F}_q \\ \mathcal{X} = ([c_1]P_i, [c_2]P_i) & \longrightarrow c_2 - a_i c_1 \end{cases}$$

Observing ι_i function, the underlying computation forms an ElGamal encryption. And computing ρ_i function requires one to hold the secrets c_1 and c_2; otherwise, one need to solve discrete logarithms. To commit to $x \in \mathbb{F}_q$, the commitment scheme is defined as

$$comm_i : \begin{cases} \mathbb{F}_q \times \mathbb{F}_q & \longrightarrow \mathbb{B}_i \\ (x, r) & \longrightarrow \iota_i(x) + [r]\mathcal{U}_i^1 \end{cases}$$

While $\mathbb{A}_i = \mathbb{G}_i$, the maps are defined as

$$\iota_i : \begin{cases} \mathbb{G}_i & \longrightarrow \mathbb{B}_i \\ X & \longrightarrow (\mathcal{O}, X) \end{cases} \quad \rho_i : \begin{cases} \mathbb{B}_i & \longrightarrow \mathbb{G}_i \\ \mathcal{X} = (C_1, C_2) & \longrightarrow C_2 - [a_i]C_1 \end{cases}$$

The same, ι_i function also forms an ElGamal encryption. And computing ρ_i function requires one to hold the secrets a_1; otherwise, one need to solve discrete logarithms. To commit to $x \in \mathbb{G}_i$, the commitment scheme is defined as

$$comm_i : \begin{cases} \mathbb{G}_i \times \mathbb{F}_q \times \mathbb{F}_q & \longrightarrow \mathbb{B}_i \\ (X, r_1, r_2) & \longrightarrow \iota_i(X) + [r_1]\mathcal{U}_i^1 + [r_2]\mathcal{U}_i^2 \end{cases}$$

ι_T and ρ_T. Here we define maps maps ι_T: $\mathbb{A}_T \longrightarrow \mathbb{B}_T$ and ρ_T: $\mathbb{B}_T \longrightarrow \mathbb{A}_T$. Depending on different equation we want to prove, there are four cases to define ι_t and ρ_T.
For pairing product equation

$$\iota_T : \begin{cases} \mathbb{G}_T & \longrightarrow \mathbb{B}_T \\ Z & \longrightarrow (1, 1, 1, Z) \end{cases} \quad \rho_T : \begin{cases} \mathbb{B}_T & \longrightarrow \mathbb{G}_T \\ (Z_{1,1}, Z_{1,2}, Z_{2,1}, Z_{2,2}) & \longrightarrow Z_{2,2} Z_{1,2}^{-a_1} (Z_{2,1} Z_{1,1}^{-a_1})^{-a_2} \end{cases}$$

For multi-scalar multiplication equation in $\mathbb{G}_1$ and $\mathbb{G}_2$, ι_T is respectively defined as

$$\iota_T : \begin{cases} \mathbb{G}_1 & \longrightarrow \mathbb{B}_T \\ X & \longrightarrow (1, 1, e(X, W_{2,1}), e(X, W_{2,2})) \end{cases}$$

$$\iota_T : \begin{cases} \mathbb{G}_1 & \longrightarrow \mathbb{B}_T \\ X & \longrightarrow (1, e(X, W_{2,1}), 1, e(X, W_{2,2})) \end{cases}$$

And ρ_T is defined as

$$\rho_T : \begin{cases} \mathbb{B}_T & \longrightarrow \mathbb{G}_i \\ (Z^{s_1}, Z^{s_2}, Z^{s_3}, Z^{s_4}) & \longrightarrow s_4 - a_1 s_2 - a_2 s_3 + a_1 a_2 s_1 P_i \end{cases}$$

where $Z = e(P_1, P_2)$.

For quadratic equation in $\mathbb{F}_q$, we define $Z = e(P_1, P_2)$ and then

$$\iota_T : \begin{cases} \mathbb{F}_q & \longrightarrow \mathbb{B}_T \\ z & \longrightarrow F(\mathcal{W}_1, \mathcal{W}_2)^z \end{cases}$$

$$\rho_T : \begin{cases} \mathbb{B}_T & \longrightarrow \mathbb{F}_q \\ (Z^{s_1}, Z^{s_2}, Z^{s_3}, Z^{s_4}) & \longrightarrow s_4 - a_1 s_2 - a_2 s_3 + a_1 a_2 s_1 \end{cases}$$

Figure 3.2: Group elements used in SXDH-based Groth-Sahai proof [3]

Assumption: SXDH	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{Z}_p$
Variable $x \in \mathbb{Z}_p$, $\mathcal{X} \in \mathbb{G}_1$	2	0	0
Variable $y \in \mathbb{Z}_p$, $\mathcal{Y} \in \mathbb{G}_2$	0	2	0
Pairing product equations	4	4	0
- Linear equation: $\overrightarrow{\mathcal{A}} \cdot \overrightarrow{\mathcal{Y}} = t_T$	2	0	0
- Linear equation: $\overrightarrow{\mathcal{X}} \cdot \overrightarrow{\mathcal{B}} = t_T$	0	2	0
Multi-scalar multiplication equation in $\mathbb{G}_1$	2	4	0
- Linear equation: $\overrightarrow{\mathcal{A}} \cdot \overrightarrow{y} = T_1$	1	0	0
- Linear equation: $\overrightarrow{\mathcal{X}} \cdot \overrightarrow{b} = T_1$	0	0	2
Multi-scalar multiplication equation in $\mathbb{G}_2$	4	2	0
- Linear equation: $\overrightarrow{a} \cdot \overrightarrow{\mathcal{Y}} = T_1$	0	0	2
- Linear equation: $\overrightarrow{x} \cdot \overrightarrow{\mathcal{B}} = T_2$	0	1	0
Quadratic equation in $\mathbb{Z}_p$	2	2	0
- Linear equation: $\overrightarrow{a} \cdot \overrightarrow{y} = t$	0	0	1
- Linear equation: $\overrightarrow{x} \cdot \overrightarrow{b} = t$	0	0	1

Cost of Groth-Sahai proofs.

Depending on different type of equations of different pairing environment, Groth-Sahai proofs require different group elements involved in computation. Please refer to 3.2 and 3.3 that shows required group elements in SXDH setting and DLIN setting. Overall Groth-Sahai proofs cost less in SXDH setting than in DLIN setting. For the cost of making commitments, SXDH setting only need two group elements, but it costs three in DLIN setting. Moreover, in pairing product equation for example, SXDH setting costs 8 group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ while DLIN setting costs 9 group elements in $\mathbb{G}$. The difference of used group elements between these two pairing settings significantly shows that SXDH setting take advantage in proofs computation.

Figure 3.3: Group elements used in DLIN-based Groth-Sahai proof [3]

Assumption: DLIN	$\mathbb{G}$	$\mathbb{Z}_p$
Variable $x \in \mathbb{Z}_p$, $\mathcal{Y} \in \mathbb{G}$	3	0
Pairing product equations	9	0
- Linear equation: $\overrightarrow{\mathcal{A}} \cdot \overrightarrow{\mathcal{Y}} = t_T$	3	0
Multi-scalar multiplication equation in $\mathbb{G}$	9	0
- Linear equation: $\overrightarrow{a} \cdot \overrightarrow{\mathcal{Y}} = T$	0	3
- Linear equation: $\overrightarrow{x} \cdot \overrightarrow{\mathcal{B}} = T$	2	0
Quadratic equation in $\mathbb{Z}_p$	6	0
- Linear equation: $\overrightarrow{x} \cdot \overrightarrow{b} = t$	0	2

3.2 Single-block P-signature schemes

In 2008 Belenkiy et al. [13] introduced the first P-signature scheme with efficient non-interactive proof and the achievement can be directly deployed for anonymous credentials. Despite single block construction that might be only suitable for single message, this paper defines the essential properties and security models of P-signatures, which can be regarded as the preliminary of P-signatures.

3.2.1 F-unforgeable signatures of single-block P-signatures

Single-block P-signatures take F-unforgeable signatures as building blocks. F-unforgeable signatures are firstly formalized by Belenkiy et al. [13] and the idea comes from the fact that Groth-Sahai proofs cannot completely prevent attackers from making forgeries while seeing (m, σ). Instead, if attackers are capable of compromising one way function F, they can produce a forgery $(F(m), \sigma)$ that can pass the verification procedure. Basically F-unforgeable signatures is inspired by the Boneh-Boyen signature scheme [33] and can be illustrated as follows.

FSigSetup(1^k). Taken security parameter k, output parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ for bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ where $\mathbb{G}_1 = \langle g \rangle$ and $\mathbb{G}_2 = \langle h \rangle$ are cyclic groups of prime order p. Moreover, we define $t = e(g, h)$.

FKeyGen($params$). The signer chooses secret-key $sk = (\alpha, \beta)$ where $\alpha, \beta \in Z_p$ and computes public-key $pk = (v, w, \bar{v}, \bar{w})$ where $v = h^\alpha$, $w = h^\beta$, $\bar{v} = g^\alpha$, $\bar{w} = g^\beta$. The generated public key pk can be verified by checking $e(g, v) = e(\bar{v}, h)$ and $e(g, w) = e(\bar{w}, h)$.

FSign($params, (\alpha, \beta), m$). The signer signs message m by randomly choosing $r \in Z_p - \{(\alpha - m)/\beta\}$ and then computing signature $\sigma = (C_1, C_2, C_2)$ where $C_1 = g^{1/(\alpha+m+\beta r)}$, $C_2 = w^r$, $C_3 = u^r$.

FVerify$(params, (v, w, \bar{v}, \bar{w}), m, \sigma)$**.** The signature $\sigma = (C_1, C_2.C_2)$ is valid if the public key is valid by checking by checking $e(g, v) = e(\bar{v}, h)$ and also the signature is valid by checking $e(g, w) = e(\bar{w}, h)$ and $e(C_1, vh^m C_2) = t$, $e(u, C_2) = e(C_3, w)$.

The above F-unforgeable signatures is proven secure based on HSDH problem and TDH problem. The full proof can be found in [13].

3.2.2 Single-block P-signatures schemes

With the help of F-unforgeable signatures, the algorithms of single-block P-signatures are described as follows.

SBPSigSetup(1^k)**.** Taken security parameter k, this function generates $params_{BM} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ for bilinear maps, $params_{GS} = (params_{BM}, params_1, params_2, params_\pi)$ for Groth-Sahai commitments and proofs. Moreover, this function randomly chooses $u \in \mathbb{G}_1$ and sets $t = e(g, h)$, finally outputs $params = (params_{GS}, u)$.

SBPKeyGen$(params)$**.** This function operates FKEYGen$(params_{BM})$ and generates public key pair $(sk, pk) = ((\alpha, \beta), (v, w, \bar{v}, \bar{w}))$.

SBPSign$(params, sk, m)$**.** Given secret-key sk and message m, a signature $\sigma = (C_1, C_2, C_3)$ is generated by operating FSign$(params, sk, m)$; that is, $C_1 = g^{1/(\alpha+m+\beta r)}$, $C_2 = w^r$, and $C_3 = u^r$.

SBPVerify$(params, pk, m, \sigma)$**.** Given public-key and a signature $sigma$, the signature is valid if it passes FVerify$(params, pk = (v, w, \bar{v}, \bar{w}), m, \sigma)$.

SBPCom$(params, m, open)$**.** Given a message m and a random value $open$, a commitment $comm$ is generated by operating GSCom$(params_2, h^m, open)$.

SBPSigObtain$(params, pk, comm, open)$ $\longleftrightarrow$ **SBPSigIssue**$(params, sk, comm)$**.** Single block P-signatures define a interactive protocol for one user to obtain a signature from an issuer. The protocol is described as follows.

1. The user randomly picks $\rho_1, rho_2 \in Z_p$ and the issuer randomly picks $z \in Z_p$.
2. The user exchanges secret variables with the issuer in a secure two-party computation protocol. After exchange, the user gets z and the issuer acquires ρ_1, rho_2.
3. The user operatesg GSCom$(params, m, open)$ and generates a commitment $comm$ to a message m and send $comm$ to the issuer.
4. The issuer verify the validity of $comm$, if valid, the issuer computes $x = (\alpha + m + \beta\rho_1 r)\rho_2$ and generates a tuple (C_1', C_2', C_3') where $C_1' = g^{1/x}$, $C_2' = w^z$, and $C_3' = u^z$, and then sends back to the user.

5. The user computes signature $\sigma = (C_1, C_2, C_3)$ where $C_1 = C_1'^{\rho_2}$, $C_2 = C_2'^{\rho_1}$, and $C_3 = C_3'^{\rho_1}$, and verifies if σ is a valid signature.

SBPProve$(params, pk, m, \sigma)$**.** The user computes commitments $R_1 = \text{GSCom}(params_1, C_1, open_1)$, $R_2 = \text{GSCom}(params_1, C_2, open_2)$, $R_3 = \text{GSCom}(params_1, C_3, open_3)$, $R_4 = \text{GSCom}(params_2, h^m, open_4)$, $R_5 = \text{GSCom}(params_1, u^m, open_5)$, then the user constructs Groth-Sahai proof
$\pi = NIPK\{(R_1 : C_1), (R_2 : C_2), (R_3 : C_3), (R_4 : h^\alpha)m(R_5, u^\beta) :$

$$e(C_1, vh^\alpha C_2) = t \wedge e(u, C_2) = e(C_3, w) \wedge \alpha = \beta\} \tag{3.8}$$

and outputs $comm = R_4$ and π.

SBPProveVerify$(params, pk, comm, \pi)$**.** One verifier accepts the statement of relation of m and σ if both public-key pk and Groth-Sahai π pass the verification.

SBPEqCommProve$(params, m, open, open')$**.** Taken two opens $open$ and $open'$, two commitments $comm$ and $comm'$ can be respectively computed by $\text{GSCom}(params_2, h^m, open)$ and $\text{GSCom}(params_2, h^m, open')$. Then a non-interactive zero-knowledge proof can be constructed as

$$\pi \longleftarrow NIZK\{(comm : g_2^\alpha), (comm' : g_2^\beta) : \alpha = \beta\} \tag{3.9}$$

SBPEqCommVerify$(params, comm, comm', \pi)$**.** A verifier accepts π if the inner product equation of π is valid.

Single-block p-signatures can only support one block of message and the security is based on HSDH problem and THD problem that are inherently given by F-unforgeable signatures, as well as SXDH problem that is ensured by Groth-Sahai proof system. Please refer the corresponding proofs in the original paper [13].

3.3 Multi-block P-signature schemes

Motivated by the applications of e-cash that contains a bunch of messages, Belenkiy et al. [14] extend single-block P-signature to support multiple blocks of messages. The invention of multi-block P-signatures can not only significantly diversify the possible applications of anonymous credentials, but also help to strengthen other cryptographic primitives, such as strongly simulatable verifiable random functions (sVRFs). In this section, we review the significant improvement of multi-block P-signatures.

3.3.1 F-unforgeable signatures of multi-block P-signatures

F-unforgeable signatures of multi-block P-signatures are extended from the F-unforgeable signatures of single-block P-signatures. The major enhancement is that the exponent calculation of C_1 of signature σ supports multi-block message $m = (m_1, ..., m_n)$. The detail algorithm of F-secure signatures is described as follows.

MFSigSetup(1^k). Taken security parameter k, output parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ for bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ where $\mathbb{G}_1 = \langle g \rangle$ and $\mathbb{G}_2 = \langle h \rangle$ are cyclic groups of prime order p. Moreover, we define $t = e(g, h)$.

MFKeyGen($params$). The signer chooses secret-key $sk = (\alpha, \beta = (\beta_1, ..., \beta_n))$ where $\alpha, \beta \in \mathbb{Z}_p$ and computes public-key $pk = (v, w, \bar{v}, \bar{w})$ where $v = h^\alpha$, $\bar{v} = g^\alpha$, $w_i = h^{\beta_i}$, , $\bar{w}_i = g^{\beta_i}$. The generated public key pk can be verified by checking $e(g, v) = e(\bar{v}, h)$ and $e(g, w_i) = e(\bar{w}_i, h)$ where $i = 1, ..., n$.

MFSign($params, (\alpha, \beta), m$). The signer signs message $m = (m_1, m_2, ..., m_n)$ by randomly choosing $r \in Z_p - \{-(\alpha + \beta_1 m_1 + ... + \beta_n m_n)\}$ and then computing a signature $\sigma = (C_1, C_2, C_2)$ where $C_1 = g^{1/(\alpha + r + \beta_1 m_1 + ... + \beta_n m_n r)}$, $C_2 = h^r$, $C_3 = u^r$.

MFVerify($params, (v, w, \bar{v}, \bar{w}), m, \sigma$). The signature $\sigma = (C_1, C_2.C_2)$ is valid if the public key is valid by checking by checking $e(g, v) = e(\bar{v}, h)$ and also the signature is valid by checking $e(g, w) = e(\bar{w}, h)$ and $e(C_1, vC_2 \prod_{i=1}^{n} w_i^{m_i}) = t$, $e(u, C_2) = e(C_3, h)$.

From the construction of above algorithm, it is evident the modification of F-unforgeable signatures of multi-block P-signatures is slight. In fact, the modified signature scheme is also proven secure based on HSDH problem and TDH problem, which is exactly the same as F-unforgeable signatures introduced in Section 3.2.1. The full proof can be found in the original paper [14].

3.3.2 Multi-block P-signatures schemes

With the help of modified F-unforgeable signatures, multi-block P-signatures can be constructed as follows.

MBPSigSetup(1^k). Taken security parameter k, this function generates $params_{BM} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ for bilinear maps, $params_{GS} = (params_{BM}, params_1, params_2, params_\pi)$ for Groth-Sahai commitments and proofs. Moreover, this function randomly chooses $u \in \mathbb{G}_1$ and sets t=e(g, h), finally outputs $params$ = ($params_{GS}$, u).

MBPKeyGen($params$). This function operates MFKEYGen($params_{BM}$) and generates public key pair (sk, pk)= ((α, β),(v,w,$\bar{v}$,$\bar{w}$)) where β=(β_1, ..., β_n), w=(w_1, ..., w_n), and $\bar{w}$=($\bar{w}_1$, ..., $\bar{w}_n$).

MBPSign($params, sk, m$). Given secret-key sk and message m = (m_1, ...,m_n), a signature $\sigma = (C_1, C_2, C_3)$ is generated by operating MFSign($params, sk, m$); that is, $C_1 = g^{1/(\alpha + r + \beta_1 m_1 + ... + \beta_n m_n r)}$, $C_2 = h^r$, and $C_3 = u^r$

MBPVerify($params, pk, m, \sigma$). Given public-key pk, message m=(m_1, ..., m_n) and a signature $sigma$, the signature is valid if it passes MFVerify($params$, pk=(v, w, $\bar{v}$, $\bar{w}$), m, σ).

MBPCom($params, m, open$)**.** Given a message m=(m_1, ..., m_n) and random values $open$ = ($open_1$, ..., $open_n$), a commitment $comm$=($comm_1$, ..., $comm_n$) where each $comm_i$ consisting of a pair (H_i, U_i) generated by operating (GSCom($params_1$, h^{m_i}, $open_{i,1}$), GSCom($params_1$, u^{m_i}, $open_{i,2}$)).

MBPSigObtain($params, pk, comm, open$) $\longleftrightarrow$ **MBPSigIssue**($params, sk, comm$)**.** In this protocol, multi-block P-signatures define a interactive protocol for one user to obtain a signature from an issuer. The protocol is the same with Single block P-signatures and described as follows.

1. The user randomly picks $\rho_1, \rho_2 \in \mathbb{Z}_p$ and the issuer randomly picks $z \in \mathbb{Z}_p$.
2. The user exchanges secret variables with the issuer in a secure two-party computation protocol. After exchange, the user gets z and the issuer acquires ρ_1, ρ_2.
3. The user operatesg GSCom($params, m, open$) and generates a commitment $comm$ to a message m and send $comm$ to the issuer.
4. The issuer verify the validity of $comm$, if valid, the issuer computes $x = (\alpha + m + \beta\rho_1 r)\rho_2$ and generates a tuple (C_1', C_2', C_3') where $C_1' = g^{1/x}$, $C_2' = w^z$, and $C_3' = u^z$, and then sends back to the user.
5. The user computes signature $\sigma = (C_1, C_2, C_3)$ where $C_1 = C_1'^{\rho_2}$, $C_2 = C_2'^{\rho_1}$, and $C_3 = C_3'^{\rho_1}$, and verifies if σ is a valid signature.

MBPProve($params, pk, m, \sigma$)**.** The user computes commitments R_1 = GSCom($params_1$, C_1, $open_1$), R_2 = GSCom($params_1$, C_2, $open_2$), R_3 = GSCom($params_1$, C_3, $open_3$), $comm$=($comm_1$, ..., $comm_n$), by operating MBPCom($params, m, open_{comm}$) then the user constructs Groth-Sahai proof

$$\begin{aligned} \pi \;=\; & NIPK[h^{m_1} \in H_1, u^{m_1} \in U_1, ..., h^{m_n} \in H_n, u^{m_n} \in U_n]\{ \\ & (h^{m_1}, u^{m_1}, w_1^{m_1}, ..., h^{m_n}, u^{m_n}, w_1^{m_n}, C_1, C_2, C_3) : \\ & e(C_1, vC_2 \prod_{i=1}^{n} w_i^{m_i}) = t \wedge e(u, C_2)e(C_3, h^{-1}) = 1 \wedge \\ & \{e(\bar{w}_i, h^{m_i})e(g^{-1}, w_i^{m_i}) = 1\}_{i=1}^{n}\} \end{aligned} \tag{3.10}$$

MBPProveVerify($params, pk, comm, \pi$)**.** One verifier accepts the statement of relation of m and σ if both public-key pk and Groth-Sahai π pass the verification.

MBPEqCommProve($params, m, open, open'$)**.** Taken two opens $open$ and $open'$, two commitments $comm$ and $comm'$ can be respectively computed by GSCom($params_2$, h^m, $open$) and GSCom($params_2$, h^m, $open'$). Then a non-interactive zero-knowledge proof can be constructed as

$$\pi \longleftarrow NIZK\{(comm : g_2^{\alpha}), (comm' : g_2^{\beta}) : \alpha = \beta\} \tag{3.11}$$

MBPEqCommVerify$(params, comm, comm', \pi)$**.** A verifier accepts π if the inner product equation of π is valid.

From the construction of multi-block signatures introduced above, even though the underlying signature is slightly modified, the corresponding proof algorithms are significant different. In order to prove the validity of P-signatures, users should generate commitments for each block of messages of P-signatures. Such construction will inevitably lead to large size of proof while the amount of blocks of messages is large. Therefore, we can conclude the efficiency of proof of multi-block P-signatures will decline as the size of message is increasing. And in terms of security, similar to single-block P-signatures, multi-block P-signatures is secure provided the hardness of HSDH problem, TDH problem and SXDH problem.

3.4 Block-wise P-signature schemes

Motivated by the emerging investigation on *attribute-based cryptography*, *predicated-based cryptography*, Izabachène et al. [4] proposed block-wise P-signatures. Essentially the message m=(m_1,...,m_n) in block-wise P-signatures represents user attributes, which is significantly different from the definition of message in traditional P-signatures. Moreover, the commitments to the blocks of messages are required to be verifiable for some predicates that represent the logical relation of different combinations of message blocks. Therefore, block-wise P-signatures can provide a variety of functionalities that traditional P-signatures cannot provide.

The proposed construction of block-wise P-signatures is based on symmetric pairings; hence, in this section we will use the notion of Type-1 pairings. Let G, $\mathbb{G}_T$ be cyclic groups of prime order p and define e: $G \times G$ be the bilinear map.

3.4.1 Vector commitments of block-wise P-signatures

Different from the traditional P-signatures, block-wise P-signatures design a new commitment scheme. Given a commitment key $(g, g_1, ..., g_n, g_{n+2}, ..., g_{2n})$, vector commitments consist of two functions.

VecCom(m, r)**.** Given message $m = (m_1, ..., m_n)$, choose $r \in Z_p$ and compute commitment V

$$V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j} \tag{3.12}$$

Moreover, to commit to a specific block of message, said m_i, we can compute a witness

$$W_i = g_i^r \cdot \prod_{j=1, j\neq i}^{n} g_{n+1-j+i}^{m_j} \tag{3.13}$$

VecOpen(m, r, V)**.** a commitment V is valid if $V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$. Moreover, to verify a witness W_i, we can check if

$$e(g_i, V) = e(g, W_i)e(g_1, g_n)^{m_i} \tag{3.14}$$

3.4.2 Block-wise P-signatures schemes

With the help of vector commitments, block-wise P-signatures, consisting F-unforgeable signatures and all proof algorithms, are described as follows.

BWPSigSetup(1^k)**.** Taken security parameter k, this function generates $params = (n, \mathbb{G}, \mathbb{G}_T, g, e, f)$ where $\mathbb{G}$ is a cyclic group of prime order p and generated by generator g, e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ is a bilinear map and $f=(f_1, f_2, f_3)$ is common reference string (CRS) for Groth-Sahai proof.

BWPKeyGen$(params)$**.** Choosing γ, ω, α, $\beta \in \mathbb{Z}_p$, $u, U_0 \in \mathbb{G}$, computing $\Omega = g^{\omega}$, $\Upsilon = g^{\gamma}$, $U_1 = g^{\beta}$, and $g_i = g^{(\alpha^i)}$ for $i \in [1, ..., n, n+2, ...2n]$. The secret-key $sk = (\gamma, \omega, \beta)$ and public-key $pk = (u, \Omega, \Upsilon, U_0, U_1, \{g_i\}_{i \in [1,2n]/\{n+1\}})$.

BWPSign$(params, sk, m)$**.** Taken message $m = (m_1, ..., m_n)$, the signer firstly chooses $r \in \mathbb{Z}_p$ and computes $V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j} = g_n^{m_1} \cdots g_1^{m_n} \cdot g^r$. Secondly the signer chooses $c \in \mathbb{Z}_p$ and computes $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$ where

$$\begin{aligned} \sigma_1 &= g^{\gamma/\omega+c}, \\ \sigma_2 &= g^c, \\ \sigma_3 &= u^c, \\ \sigma_4 &= (U_0 V^{\beta})^c, \\ \sigma_5 &= V^c, \\ \sigma_6 &= V \end{aligned} \tag{3.15}$$

BWPVerify$(params, pk, m, \sigma)$**.** Receiving message $m = (m_1, ..., m_n)$ and $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$, the verifier accepts the signature if $\sigma_6 = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$ and

$$\begin{aligned} e(A, g) &= e(\sigma_1, \Omega, \sigma_2), \\ e(u, \sigma_2) &= e(\sigma_3, g), \\ e(g, \sigma_4) &= e(U_0, \sigma_2)e(U_1, \sigma_5), \\ e(g, \sigma_5) &= e(\sigma_6, \sigma_2), \end{aligned} \tag{3.16}$$

BWPWitGen$(params, \mathcal{R}, i, m, X, \sigma)$**.** Taken relation $\mathcal{R}$, index i, $m = (m_1, ..., m_n)$, $X = (x_1, ..., x_n)$ and $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$,

1. If $\mathcal{R} = \mathcal{R}^{EQ}$ and $m_i = x_i$ where $i \in [1, n]$, compute the witness

$$W = g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j}$$

2. If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$ and $m_i \neq x_i$, where $i \in [1, n]$, compute the witness $W = (W_0, W_1, W_2, W_3, W_4)$ where

$$\begin{aligned} W_0 &= g^{1/(m_i - x_i)}, \\ W_1 &= g_1^{(m_i - x_i)}, \\ W_2 &= g^{(m_i - x_i)}, \\ W_3 &= g_{2n}^{(m_i - x_i)} \\ W_4 &= g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j} \end{aligned} \tag{3.17}$$

3. If $\mathcal{R} = \mathcal{R}^{IP}$, $i = 0$, and , $m \cdot X = 0$, compute the witness

$$\begin{aligned} W &= \prod_{i=1}^{n} W_i^{x_i} for\ each \\ W_i &= g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j} \end{aligned} \tag{3.18}$$

4. If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$, $i = 0$, and , $m \cdot X \neq 0$, compute the witness $W = (W_0, W_1, W_2, W_3, W_4)$ where

$$\begin{aligned} W_0 &= g^{1/(m_i - x_i)}, \\ W_1 &= g_1^{(m_i - x_i)}, \\ W_2 &= g^{(m_i - x_i)}, \\ W_3 &= g_{2n}^{(m_i - x_i)} \\ W_4 &= \prod_{i=1}^{n} W_{4,i}^{x_i} \quad for\ each \\ W_{4,i} &= g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j} \end{aligned} \tag{3.19}$$

BWPWitVerify$(params, pk, i, X, W, \sigma)$**.** Taken public-key pk, index i, $X = (x_1, ..., x_n)$, witness W and a valid signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$,

1. If $\mathcal{R} = \mathcal{R}^{EQ}$ and $i \in [1, n]$, output 1 if

$$e(g_i, \sigma_6) = e(g_1, g_n)^{x_i} \cdot e(g, W) \tag{3.20}$$

2. If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$ and $i \in [1, n]$, parse $W = (W_0, W_1, W_2, W_3, W_4)$ and check if the following equations hold.

$$\begin{aligned} & e(g_i, \sigma \cdot g_{n+1-i}^{-x_i}) = e(W_1, g_n) \cdot e(g, W_4) \\ & e(W_0, W_1) = e(g, g_1), \\ & e(W_1, g) = e(g_1, W_2), \\ & e(W_1, g_{2n}) = e(g_1, W_3), \end{aligned} \tag{3.21}$$

3. If $\mathcal{R} = \mathcal{R}^{IP}$ and $i = 0$, output 1 if

$$e(g, W) = e(\prod_{i=1}^{n} g_i^{x_i}, \sigma_6) \tag{3.22}$$

4. If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$ and $i = 0$, parse $W = (W_0, W_1, W_2, W_3, W_4)$ and check if the following equations hold.

$$\begin{aligned} & e(\prod_{i=1}^{n} g_i^{x_i}, \sigma_6) = e(W_1, g_n) e(g, W_4) \\ & e(W_0, W_1) = e(g, g_1), \\ & e(W_1, g) = e(g_1, W_2), \\ & e(W_1, g_{2n}) = e(g_1, W_3), \end{aligned} \tag{3.23}$$

BWPSigObtain$(pk, M_{part1}, open_{part1}) \longleftrightarrow$ **MBPSigIssue**$(sk, V_{part1}, M_{part2})$**.** this interactive protocol provides users to acquire signatures from some organizations.

1. the user chooses $r_{part1} \in \mathbb{Z}_p$ and computes a commitment $V_{part1} = g^{r_{part1}} \cdot \prod_{j=1}^{n_1} g_{n+1-j}^{m_j}$ for message M_{part1}=(m_1, ..., m_{n_1}). The the user sets $open_{part1} = M_{part1}$=(m_1, ..., m_{n_1}, r') and computes a witness-indistinguishable proof for knowledge of M_{part1} such that $V_{part1} = g^{r_{part1}} \cdot \prod_{j=1}^{n_1} g_{n+1-j}^{m_j}$.
2. the signature issuer chooses $r^{part2}, c \in \mathbb{Z}_p$ and computes

$$\begin{aligned} & V = V_{part1} \cdot \prod_{j=n+1}^{n} g_{n+1-j}^{m_j} \\ & \sigma_1 = g^{\gamma/(\omega+c)}, \\ & \sigma_2 = g^c, \\ & \sigma_3 = u^c, \\ & \sigma_4 = (U_0 (V \cdot g^{r_{part2}})^{\beta})^c, \\ & \sigma_5 = (V \cdot g^{r_{part2}})^c, \sigma_6 = V \cdot g^{r_{part2}} \end{aligned} \tag{3.24}$$

and return $\bar{\sigma} = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r_{part2})$.

3. the user outputs $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$ where $r = r_{part1} + r_{part2}$.

BWPProve1$(params, pk, i, S = \{i\}, m, X, \sigma)$**.** Taken $m = (m_1, ..., m_n)$, $X = (x_1, ..., x_n)$ and $\sigma = (\sigma_1,\sigma_2,\sigma_3, \sigma_4,\sigma_5,\sigma_6,r)$. Firstly this proof needs to compute $\{ C_{X_{t,j}} = \text{GSCom}(X_{t,j}, open_{x_{t,j}}) \}_{t\in S, j\in\{1,2,3\}}$ where $\{(X_{t,1}, X_{t,2}, X_{t,2}) = (g_1^{x_t}, g^{x_t}, g_{2n}^{x_t})\}_{t\in S}$. Secondly this proof computes commitments for $\{C_{\sigma_j} = GSCom(\sigma_j, open_{\sigma_j})\}_{j=1...6}$. Thirdly this proof chooses $\sigma_7 \in \mathbb{G}$ and $\theta \in \mathbb{Z}_p$, computes $C_{\sigma_7} = \text{GSCom}(\sigma_7, open_{\sigma_7})$ and $C_{\sigma_\theta} = \text{GSCom}(\sigma_\theta, open_{\sigma_\theta})$, and then constructs a proof π_{theta} proving the following equations.

$$\begin{aligned} & e(\sigma_7, g) = e(\sigma_1, \Omega \cdot \sigma_2) \wedge \\ & e(u, \sigma_2) = e(\sigma_3, g) \wedge \\ & e(g, \sigma_4) = e(U_0, \sigma_2)e(U_1, \sigma_5) \wedge \\ & e(g, \sigma_5) = e(\sigma_6, \sigma_2) \wedge \\ & \theta = 1 \wedge \\ & e(\Upsilon/\sigma_7, g^\theta) = 1 \end{aligned} \tag{3.25}$$

There are two cases for this proof

- If $\mathcal{R} = \mathcal{R}^{EQ}$, this proof computes commitment $C_W = \text{GSCom}(W, open_W)$ where W=BWWitGen(pk,$\mathcal{R}^{EQ}$,i,m,X,σ) and then computes proofs π_{x_i}, $\{\pi_{X_{t,j}}\}_{t\in S, j=1,2}$ such that

$$\begin{aligned} & e(g_i, \sigma_6) = e(X_{i,1}, g_n)e(g, W) \wedge \\ & e(X_{i,2}, g_1) = e(X_{i,1}, g) \wedge \\ & e(X_{i,2}, g_{2n}) = e(X_{i,3}, g) \end{aligned} \tag{3.26}$$

  The proof is $\pi = (\{C_{x_{t,j}}\}_{t\in S, j=1,2,3}, \{C_{\sigma_j}\}_{j=1,...,7}, C_W, C_\theta, \pi_\theta, \pi_{x_i}, \{\pi_{x_{t,j}}\}_{t\in S, j=1,2},)$.

- If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$, this proof computes commitments $\{C_{W_j}\}_{j=0,...,4}$ to (W_0, W_1, W_2, W_3, W_4) and generates proofs π_{x_i}, π_W satisfying

$$\begin{aligned} & e(g_i, \sigma_6) \cdot e(X_{i,1}, g_n)^{-1} = e(W_1, g_n) \cdot e(g, W_4) \wedge \\ & e(W_0, W_1) = e(g, g_1) \wedge \\ & e(W_1, g) = e(g_1, W_2) \wedge \\ & e(W_1, g_{2n}) = e(g_1, W_3) \end{aligned} \tag{3.27}$$

  and also generates proof $\{\pi_{X_{t,j}}\}_{t\in S, j=1,2}$. The final proof is π=($\{C_{x_{t,j}}\}_{t\in S, j=1,2,3}$, $\{C_{\sigma_j}\}_{j=1,...,7}$, $\{C_{W_j}\}_{j=0,...,4}$, C_θ,π_θ, π_{x_i}, $\{\pi_{x_{t,j}}\}_{t\in S, j=1,2}$).

BWPProve2$(params, pk, \mathcal{R}, i, m, , X, \sigma)$**.** Similar to BWPProve1, firstly this proof computes commitments $\{C_{\sigma_j}\}_{j=1,...,7}$, C_θ, as well as proofs π_θ and π_{x_i}. Then there are four cases in this proof.

- If $\mathcal{R} = \mathcal{R}^{IP}$, compute a commitment $C_W = \text{GSCom}(W, open_W)$ where $W = \text{BWPWitGen}(pk, \mathcal{R}^{IP}, 0, m, X, \sigma)$ and compute a proof π_X such that

$$e(\prod_{j=1}^{n} g_j^{x_j}, \sigma_6) = e(g, W)$$

Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,\ldots,7}, C_W, C_\theta, \pi_\theta, \pi_X\ )$.

- If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$, compute $C_g = \text{GSCom}(g, open_g)$, $\{C_{W_j}\}_{j=0,\ldots,4}$ where $W_j = \text{BWPWitGen}(\ pk, \mathcal{R}^{\overline{IP}}, 0, m, X, \sigma)$, and compute a proof π_X such that

$$\begin{aligned} & e(W_0, W_1) = e(g, g_1) \wedge \\ & e(W_1, g) = e(g_1, W_2) \wedge \\ & e(W_1, g_{2n}) = e(g_1, W_3) \wedge \\ & e(g/g, g^\theta) = 1 \wedge \\ & e(\prod_{j=1}^{n} g_j^{x_j}, \sigma_6) = e(g, W_4) \cdot e(W_1, g_n) \end{aligned} \tag{3.28}$$

Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,\ldots,7}, \{C_{W_j}\}_{j=0,\ldots,4}, C_\theta, \pi_\theta, \pi_X\ )$.

- If $\mathcal{R} = \mathcal{R}^{EQ}$ compute a commitment $C_W = \text{GSCom}(W, open_W)$ where $W = \text{BWPWitGen}(pk, \mathcal{R}^{EQ}, i, m, X, \sigma)$, $C_{X_i} = \text{GSCom}(X_i, open_{X_i})$ where $X_i = g_1^{x_i}$, and compute proofs π_W and π_X such that

$$e(g_i, \sigma_6) = e(X_i, g_n) e(g, W) \wedge e(X_i / g_1^{x_i}, g^\theta)$$

Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,\ldots,7}, C_W, C_{X_i}, C_\theta, \pi_\theta, \pi_W, \pi_X\ )$.

- If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$, compute $\{\ C_{W_j}\ \}_{j=0,\ldots,4}$ where $W_j = \text{BWPWitGen}(pk, \mathcal{R}^{\overline{EQ}}, 0, m, X, \sigma)$, $C_{X_i} = \text{GSCom}(X_i, open_{X_i})$ where $X_i = g_1^{x_i}$, and compute proofs $\pi_{X_{i,W}}$, $\{\pi_{W_j}\}_{j=1,\ldots,3}$ π_{X_i}, π_g such that

$$\begin{aligned} & e(g_i, \sigma_6) e(X_i, g_n)^{-1} = e(W_1, g_n) e(g, W) \wedge \\ & e(W_0, W_1) = e(g, g_1) \wedge \\ & e(W_1, g) = e(g_1, W_2) \wedge \\ & e(W_1, g_{2n}) = e(g_1, W_3) \wedge \\ & e(X_i / g_1^{x_i}, g^\theta) = e(g/g, g^\theta) = 1 \end{aligned} \tag{3.29}$$

Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,\ldots,7}, \{C_{W_j}\}_{j=0,\ldots,4}, C_{X_i}, C_\theta, \pi_\theta, \pi_{X_{i,W}}, \{\pi_{W_j}\}_{j=1,\ldots,3}, \pi_\theta\ )$.

BWPEqCommProve$(params, m, open, open')$**.** Taken two opens *open* and *open*$'$, two commitments *comm* and *comm*$'$ can be respectively computed by C=GSCom(X,

open) and C'=GSCom(X', *open*$'$). Then a non-interactive zero-knowledge proof can be constructed with $f_1 = (f_1, 1, g)$, $f_2 = (1, f_2, g)$, $f_3 = (f_{31}, f_{32}, f_{33})$ and prove the tuple (ρ_1, ρ_2, ρ_3) such that

$$C \cdot C' = (f_1^{\rho_1} f_{31}^{\rho_3} f_2^{\rho_2} f_{32}^{\rho_3}, f_{33}^{\rho_3})$$

Efficiency is an attractive point of block-wise P-signatures. Observing the non-interactive proof algorithm of block-wise P-signatures, it is remarkable that the size of proof will not grow as the amount of message blocks increases. Therefore we can conclude that construction block-wise p-signatures is more efficient than the construction of multi-block P-signatures.

Provable security of block-wise P-signatures

Block-wise P-signatures consider three kinds of forgeries in security proof. For the later requirements of considering security of our revision on block-wise P-signatures, it is necessary to review the attack model of it.

- Type-I F-forgeries show that an adversary outputs a forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*, \sigma_6^*, r_i^*)$, such that the triple $(\sigma_1^*, \sigma_2^*, \sigma_3^*)$ is not queried before.

- Type-II F-forgeries show that an adversary outputs a forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*, \sigma_6^*, r_i^*)$, such that the tuple $(\sigma_1^*, \sigma_2^*, \sigma_3^*)$ can be taken from $j'th$ queried. However, σ_6^* is different from $\sigma_{j,6}$.

- Type-III F-forgeries show that an adversary outputs a forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*, \sigma_6^*, r_i^*)$, such that the tuple $(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_6^*)$ can be taken from $j'th$ queried and the tuple can pass the signature verification. However, the adversary is capable of generating a valid signature witness proof that passes the witness verification procedure. This attack can be further discussed into four cases.

  - Type-III-A attacks describe the adversary generates a predicate of the form $(R^{EQ}, i, F(x_t^*))$ for some $i \in 1, ..., n$, and some random vector $X^* = (x_1^*, ..., x_n^*)$ such that x_i^* is not queried before and the forged witness W^* satisfies Witness-Verify$(pk, R^{EQ}, i, X^*, W^*, \sigma) = 1$.

  - Type-III-B attacks describe the adversary generates a predicate of the form $(R^{\overline{EQ}}, i, F(x_t^*))$ for some $i \in 1, ..., n$, and some random vector $X^* = (x_1^*, ..., x_n^*)$ such that x_i^* is not queried before and the forged witness W^* satisfies Witness-Verify$(pk, R^{\overline{EQ}}, i, X^*, W^*, \sigma) = 1$.

  - Type-III-C attacks describe the adversary generates a predicate of the form $(R^{IP}, 0, X^*)$ for $X^* \in \mathbb{Z}_p^n$, and the forged witness W^* satisfies Witness-Verify$(pk, R^{IP}, i, X^*, W^*, \sigma) = 1$.

- Type-III-D attacks describe the adversary generates a predicate of the form ($R^{\overline{IP}}$, 0, X^*) for $X^* \in \mathbb{Z}_p^n$, and the forged witness W^* satisfies Witness-Verify(pk, $R^{\overline{IP}}$, i, X^*, W^*, σ) = 1.

Responding to the attack model described above, block-wise P-signatures are proved secure on the hardness of HSDH problem, FlexDH problem and n-FlexDHE. To be more specific, Type-I forgeries are proved secure on the security assumption of q-HSDH problem while Type-II forgeries are proved secure on the assumption of FlexDH problem. Moreover, all four different cases of Type-III forgeries are proved secure on the security assumption of n-FlexDHE problem. Please see the full proofs in the original paper [4].

Chapter 4

Asymmetric Pairing Based Block-wise P-signatures

The original block-wise P-signatures which is based on symmetric pairings are practically inflexible and inefficient in implementation. For the choice of symmetric pairings in different security degree, up to now there are only two known curves which can be used to implement symmetric pairings satisfying security requirements of AES-80 and AES-128; therefore, block-wise P-signatures based on symmetric pairings are highly restricted to applied in different security context. Moreover, symmetric pairings take more time in pairing computation than their asymmetric counterparts; therefore, block-wise P-signatures based on symmetric pairings are practically inefficient. Motivated by the above two critical considerations on efficiency issue, it is necessary to revise original block-wise P-signatures to asymmetric pairings. In this chapter, we firstly propose a block-wise P-signatures based on asymmetric pairings which are provably secure and computationally efficient.

4.1 Block-wise P-signatures based on asymmetric pairings

Our new block-wise P-signatures are modified based on the original block-wise P-signatures. To achieve this goal, there are two critical points we need to carefully deal with in our design. Firstly, the working environment is on asymmetric pairings, which do not provide elements commutable property between groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Therefore, it is necessary to identify each equation containing appearance of swapping group elements in the original block-wise P-signature scheme and try to design around for fitting the property of asymmetric pairings. The second critical point is that the revised block-wise P-signature scheme should be also provably secure against forgery attacks; otherwise, the revision cannot be used in practical applications. With these two design principles, we propose a new block-wise P-signatures based on Type-3 pairings and the algorithms are illustrated as follows.

Modification summary.

Pairing computation in original block-wise P-signatures significantly heavily rely on the group elements commutation in verification procedure and this situation can be found in the majority of pairing equations of block-wise P-signatures; fortunately, all of them can be designed around. To solve this problem, we mainly adopt two techniques. Firstly, instead of $\{g_i\}_{i\in[1,2n]/\{n+1\}} \in \mathbb{G}$ in the public key setting, we extend it into two set of elements: $\{g_i\}_{i\in[1,2n]/\{n+1\}} \in \mathbb{G}_1$ and $\{h_i\}_{i\in[1,2n]/\{n+1\}} \in \mathbb{G}_2$. This public key extension setting can possess the advantage of preventing the original design from dramatical modification and untwist most cases of group element commutation. Moreover, this technique will not increase extra verification equation, which means the efficiency can be maintained. Secondly, instead of $\sigma_2 = g^c \in \mathbb{G}$ where $c \in \mathbb{Z}_p^*$, we extend it into two parts, which are $\sigma_{21} = g^c \in \mathbb{G}_1$ and $\sigma_{22} = h^c \in \mathbb{G}_2$. This modification helps to solve the problem of single element being used both in $\mathbb{G}_1$ and $\mathbb{G}_2$; However, this extension will result in one extra unverified group element in signature $\sigma_2 2$. Therefore, we have to increase another verifying equation $e(\sigma_{21}, \sigma_{22}^{-1}) = 1_{\mathbb{G}_T}$ in the signature verification to ensure the security of signature. Using the above two techniques we successfully port block-wise P-signatures into asymmetric pairings.

Our proposed block-wise P-signatures.

BWPSigSetup(1^k). Taken security parameter k, this function generates $params = (n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, h, e, f)$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order p and generated by generators g_1 and g_2, e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a bilinear map and $\mathcal{U}=(\mathcal{U}_1,\mathcal{U}_2)$ is common reference string (CRS) for SXDH-based Groth-Sahai proof.

BWPKeyGen($params$). Choosing $\gamma, \omega, \alpha, \beta \in \mathbb{Z}_p$, $u \in \mathbb{G}_2$, $U_0 \in \mathbb{G}_1$, computing $\Omega = g^\omega$, $\Upsilon = g^\gamma$, $U_1 = h^\beta$, $g_i = g^{(\alpha^i)}$ for $i \in [1, ..., n, n+2, ...2n]$ and $h_i = h^{(\alpha^i)}$ for $i \in [1, ..., n, n+2, ...2n]$. The secret-key $sk = (\gamma, \omega, \beta)$ and public-key $pk = (u, \Omega, \Upsilon, U_0, U_1, \{g_i\}_{i\in[1,2n]/\{n+1\}}, \{h_i\}_{i\in[1,2n]/\{n+1\}})$.

BWPSign($params, sk, m$). Taken message $m = (m_1, ..., m_n)$, the signer firstly chooses $r \in \mathbb{Z}_p$ and computes $V = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j} = g_n^{m_1} \cdots g_1^{m_n} \cdot g^r$. Secondly the signer chooses $c \in \mathbb{Z}_p$ and computes $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$ where

$$
\begin{aligned}
\sigma_1 &= h^{\gamma/\omega+c}, \\
\sigma_{21} &= g^c, \\
\sigma_{22} &= h^c, \\
\sigma_3 &= u^c, \\
\sigma_4 &= (U_0 \times V^\beta)^c, \\
\sigma_5 &= V^c, \\
\sigma_6 &= V
\end{aligned}
\tag{4.1}
$$

BWPVerify$(params, pk, m, \sigma)$**.** Receiving message $m = (m_1, ..., m_n)$ and $\sigma = (\sigma_1, \sigma_{21}, \sigma_{21}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$, the verifier accepts the signature if $\sigma_6 = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$ and

$$\begin{aligned} &e(\Upsilon, h) = e(\Omega \times \sigma_{21}, \sigma_1), \\ &e(\sigma_{21}, u) = e(g, \sigma_3), \\ &e(\sigma_4, h) = e(U_0, \sigma_{22}) e(\sigma_5, U_1), \\ &e(\sigma_5, h) = e(\sigma_6, \sigma_{22}), \\ &e(\sigma_{21}, \sigma_{22}^{-1}) = 1_{\mathbb{G}_T}, \end{aligned} \tag{4.2}$$

BWPWitGen$(params, \mathcal{R}, i, m, X, \sigma)$**.** Taken relation $\mathcal{R}$, index i, $m = (m_1, ..., m_n)$, $X = (x_1, ..., x_n)$ and $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$,

1. If $\mathcal{R} = \mathcal{R}^{EQ}$ and $m_i = x_i$ where $i \in [1, n]$, compute the witness

$$W = g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j}. \tag{4.3}$$

2. If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$ and $m_i \neq x_i$, where $i \in [1, n]$, compute the witness $W = (W_0, W_1, W_2, W_3, W_4)$ where

$$\begin{aligned} W_0 &= h^{1/(m_i - x_i)}, \\ W_1 &= g_1^{(m_i - x_i)}, \\ W_2 &= g^{(m_i - x_i)}, \\ W_3 &= g_{2n}^{(m_i - x_i)} \\ W_4 &= g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j} \end{aligned} \tag{4.4}$$

3. If $\mathcal{R} = \mathcal{R}^{IP}$, $i = 0$, and , $m \cdot X = 0$, compute the witness

$$\begin{aligned} W &= \prod_{i=1}^{n} W_i^{x_i} \text{ for each} \\ W_i &= g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j} \end{aligned} \tag{4.5}$$

4. If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$, $i = 0$, and , $m \cdot X \neq 0$, compute the witness $W = (W_0, W_1,$

W_2,W_3,W_4) where

$$\begin{aligned} W_0 &= g^{1/(m_i - x_i)}, \\ W_1 &= g_1^{(m_i - x_i)}, \\ W_2 &= g^{(m_i - x_i)}, \\ W_3 &= g_{2n}^{(m_i - x_i)} \\ W_4 &= \prod_{i=1}^{n} W_{4,i}^{x_i} \text{ for each} \\ W_{4,i} &= g_i^r \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{m_j} \end{aligned} \tag{4.6}$$

BWPWitVerify$(params, pk, i, X, W, \sigma)$**.** Taken public-key pk, index i, $X = (x_1, ..., x_n)$, witness W and a valid signature $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$,

1. If $\mathcal{R} = \mathcal{R}^{EQ}$ and $i \in [1, n]$, output 1 if

$$e(\sigma_6, h_i) = e(g_1, h_n)^{x_i} \cdot e(W, h) \tag{4.7}$$

2. If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$ and $i \in [1, n]$, parse $W = (W_0, W_1, W_2, W_3, W_4)$ and check if the following equations hold.

$$\begin{aligned} &e(\sigma_6 \cdot g_{n+1-i}^{-x_i}, h_i) = e(W_1, h_n) \cdot e(W_4, h) \\ &e(W_1, W_0) = e(g_1, h), \\ &e(W_1, h) = e(W_2, h_1), \\ &e(W_1, h_{2n}) = e(W_3, h_1), \end{aligned} \tag{4.8}$$

3. If $\mathcal{R} = \mathcal{R}^{IP}$ and $i = 0$, output 1 if

$$e(W, h) = e(\sigma_6, \prod_{i=1}^{n} h_i^{x_i}) \tag{4.9}$$

4. If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$ and $i = 0$, parse $W = (W_0, W_1, W_2, W_3, W_4)$ and check if the following equations hold.

$$\begin{aligned} &e(\sigma_6, \prod_{i=1}^{n} h_i^{x_i}) = e(W_1, h_n) \cdot e(W_4, h) \\ &e(W_1, W_0) = e(g_1, h), \\ &e(W_1, h) = e(W_2, h_1), \\ &e(W_1, h_{2n}) = e(W_3, h_1), \end{aligned} \tag{4.10}$$

BWPSigObtain$(pk, M_{part1}, open_{part1}) \longleftrightarrow$ **MBPSigIssue**$(sk, V_{part1}, M_{part2})$**.** this interactive protocol provides users to acquire signatures from some organizations.

1. the user chooses $r_{part1} \in \mathbb{Z}_p$ and computes a commitment $V_{part1} = g^{r_{part1}} \cdot \prod_{j=1}^{n_1} g_{n+1-j}^{m_j}$ for message M_{part1}=(m_1, ..., m_{n_1}). The the user sets $open_{part1} = M_{part1}$=(m_1, ..., m_{n_1}, r') and computes a witness-indistinguishable proof for knowledge of M_{part1} such that $V_{part1} = g^{r_{part1}} \cdot \prod_{j=1}^{n_1} g_{n+1-j}^{m_j}$.
2. the signature issuer chooses $r^{part2}, c \in Z_p$ and computes

$$
\begin{aligned}
V &= V_{part1} \cdot \prod_{j=n+1}^{n} g_{n+1-j}^{m_j} \\
\sigma_1 &= h^{\gamma/(\omega+c)}, \\
\sigma_{21} &= g^c, \\
\sigma_{22} &= h^c, \\
\sigma_3 &= u^c, \\
\sigma_4 &= (U_0(V \cdot g^{r_{part2}})^{\beta})^c, \\
\sigma_5 &= (V \cdot g^{r_{part2}})^c, \\
\sigma_6 &= V \cdot g^{r_{part2}}
\end{aligned}
\tag{4.11}
$$

   and return $\bar{\sigma} = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r_{part2})$.
3. the user outputs $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$ where $r = r_{part1} + r_{part2}$.

BWPProve1$(params, pk, i, S = \{i\}, m, X, \sigma)$**.** Taken $m = (m_1, ..., m_n)$, $X = (x_1, ..., x_n)$ and $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$. Firstly this proof needs to compute $\{ C_{X_{t,j}} = \text{GSCom}(X_{t,j}, open_{x_{t,j}}) \}_{t \in S, j \in \{1,2,3\}}$ where $\{(X_{t,1}, X_{t,2}, X_{t,2}) = (g_1^{x_t}, g^{x_t}, g_{2n}^{x_t})\}_{t \in S}$. Secondly this proof computes commitments for $\{C_{\sigma_j} = GSCom(\sigma_j, open_{\sigma_j})\}_{j=1...7}$. Thirdly this proof chooses $\sigma_7 \in G$ and $\theta \in Z_p$, computes $C_{\sigma_7} = \text{GSCom}(\sigma_7, open_{\sigma_7})$ and $C_{\sigma_\theta} = \text{GSCom}(\sigma_\theta, open_{\sigma_\theta})$, and then constructs a NIZK proof π_{theta} proving the following equations.

$$
\begin{aligned}
& e(\Upsilon, h) = e(\Omega \times \sigma_{21}, \sigma_1) \wedge \\
& e(\sigma_{21}, u) = e(g, \sigma_3) \wedge \\
& e(\sigma_4, h) = e(U_0, \sigma_{22}) e(\sigma_5, U_1) \wedge \\
& e(\sigma_5, h) = e(\sigma_6, \sigma_{22}) \wedge \\
& e(\sigma_{21}, \sigma_{22}^{-1}) = 1_{\mathbb{G}_T} \wedge \\
& \theta = 1 \wedge \\
& e(g^{\theta}, \Upsilon/\sigma_7) = 1_{\mathbb{G}_T}
\end{aligned}
\tag{4.12}
$$

There are two cases for this proof.

- If $\mathcal{R} = \mathcal{R}^{EQ}$, this proof computes commitment C_W = GSCom(W, $open_W$) where W=BWWitGen(pk, R^{EQ}, i, m, X, σ) and then computes proofs π_{x_i}, $\{\pi_{X_{t,j}}\}_{t\in S,j=1,2}$ such that

$$\begin{aligned} & e(\sigma_6, h_i) = e(X_{i,1}, h_n) \cdot e(W, h) \wedge \\ & e(X_{i,2}, h_1) \cdot e(X_{i,1}, h) \wedge \\ & e(X_{i,2}, h_{2n}) \cdot e(X_{i,3}, h) \end{aligned} \tag{4.13}$$

  The proof is $\pi = (\{C_{x_{t,j}}\}_{t\in S,j=1,2,3}, \{C_{\sigma_j}\}_{j=1,...,8}, C_W, C_\theta, \pi_\theta, \pi_{x_i}, \{\pi_{x_{t,j}}\}_{t\in S,j=1,2},)$.

- If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$, this proof computes commitments $\{C_{W_j}\}_{j=0,...,4}$ to (W_0, W_1, W_2, W_3, W_4) and generates proofs π_{x_i}, π_W satisfying

$$\begin{aligned} & e(\sigma_6, h_i) \cdot e(X_{i,1}, h_n)^{-1} = e(W_1, h_n) \cdot e(W_4, h) \wedge \\ & e(W_1, W_0) = e(g_1, h) \wedge \\ & e(W_1, h) \cdot e(W_2, h_1) \wedge \\ & e(W_1, h_{2n}) = e(W_3, h_1) \end{aligned} \tag{4.14}$$

  and also generates proof $\{\pi_{X_{t,j}}\}_{t\in S,j=1,2}$. The final proof is π=($\{C_{x_{t,j}}\}_{t\in S,j=1,2,3}$, $\{C_{\sigma_j}\}_{j=1,...,8}$, $\{C_{W_j}\}_{j=0,...,4}$, C_θ,π_θ, π_{x_i}, $\{\pi_{x_{t,j}}\}_{t\in S,j=1,2}$).

BWPProve2($params, pk, R, i, m, , X, \sigma$**).** Similar to BWPProve1, firstly this proof computes commitments $\{C_{\sigma_j}\}_{j=1,...,8}$, C_θ, as well as proofs π_θ and π_{x_i}. Then there are four cases in this proof.

- If $\mathcal{R} = \mathcal{R}^{IP}$, compute a commitment C_W = GSCom($W, open_W$) where W = BWPWitGen(pk,R^{IP},0,m,X,σ) and compute a proof π_X such that

$$e(\sigma_6, \prod_{j=1}^{n} h_j^{x_j}) = e(W, h) \tag{4.15}$$

  Finally, the proof π = ($\{C_{\sigma_j}\}_{j=1,...,8}$, C_W, C_θ,π_θ, π_X).

- If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$, compute C_g = GSCom(g, $open_g$), $\{C_{W_j}\}_{j=0,...,4}$ where W_j = BWPWitGen(pk, $\mathcal{R}^{\overline{IP}}$, 0, m, X, σ), and select $T \in mathbbG_2$ and compute a proof π_X such that

$$\begin{aligned} & e(W_1, W_0) = e(g_1, T) \wedge \\ & e(W_1, h) = e(W_2, h_1) \wedge \\ & e(W_1, h_{2n}) = e(W_3, h_1) \wedge \\ & e(g^\theta, T/h) = 1_{\mathbb{G}_T} \wedge \\ & e(\sigma_6, \prod_{j=1}^{n} h_j^{x_j}) = e(W_1, h_n) \cdot e(W_4, h) \end{aligned} \tag{4.16}$$

  Finally, the proof π = ($\{C_{\sigma_j}\}_{j=1,...,8}$, $\{C_{W_j}\}_{j=0,...,4}$, C_θ,π_θ, π_X).

- If $\mathcal{R} = \mathcal{R}^{EQ}$ compute a commitment $C_W = \text{GSCom}(W, open_W)$ where $W = \text{BWPWitGen}(pk, R^{EQ}, i, m, X, \sigma)$, $C_{X_i} = \text{GSCom}(X_i, open_{X_i})$ where $X_i = h_1^{x_i}$, and compute proofs π_W and π_X such that

$$\begin{aligned} & e(\sigma_6, h_i) = e(g_n, X_i) \cdot e(W, h) \wedge \\ & e(g^{\theta}, X_i / h_1^{x_i}) = 1_{\mathbb{G}_T} \end{aligned} \tag{4.17}$$

  Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,...,8}, C_W, C_{X_i}, C_{\theta}, \pi_{\theta}, \pi_W, \pi_X\ )$.

- If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$, compute $\{\ C_{W_j}\ \}_{j=0,...,4}$ where $W_j = \text{BWPWitGen}(pk, R^{\overline{EQ}}, 0, m, X, \sigma)$, $C_{X_i} = \text{GSCom}(X_i, open_{X_i})$ where $X_i = g_1^{x_i}$, and compute proofs $\pi_{X_{i,W}}$, $\{\pi_{W_j}\}_{j=1,...,3}$ π_{X_i}, π_g such that

$$\begin{aligned} & e(\sigma_6, h_i) \cdot e(X_i, h_n)^{-1} = e(W_1, h_n) \cdot e(W_4, h) \wedge \\ & e(W_1, W_0) = e(g_1, h) \wedge \\ & e(W_1, h) = e(W_2, h_1) \wedge \\ & e(W_1, h_{2n}) = e(W_3, h_1) \wedge \\ & e(g^{\theta}, X_i / h_1^{x_i},) = 1_{\mathbb{G}_T} \wedge \\ & e(g^{\theta}, T/h) = 1_{\mathbb{G}_T} \end{aligned} \tag{4.18}$$

  Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,...,8}, \{C_{W_j}\}_{j=0,...,4}, C_{X_i}, C_{\theta}, \pi_{\theta}, \pi_{X_{i,W}}, \{\pi_{W_j}\}_{j=1,...,3}, \pi_{\theta}\ )$.

BWPEqCommProve($params, m, open, open'$). Taken two opens $open_X$ and $open_Y$, two commitments C_X and C_Y can be respectively computed by C_X=GSCom(X, $open_X$) and C_Y=GSCom(Y, $openY$). To prove $X = Y$, we can construct an non-interactive zero-knowledge proof $C_X \cdot C_Y = 1_{\mathbb{G}_T}$ based on $SXDH$ assumption.

Result of our modification on block-wise P-signatures.

Essentially our proposed block-wise P-signatures are based on Izabachène et al.'s idea and extend the public key set with $h_i = h^{(\alpha^i)}$ for $i \in [1, ..., n, n+2, ...2n]$. In this way we successfully untwist the commutable property of Type-1 pairings and make block-wise P-signature be constructed on Type-3 pairings. Figure 4.1 describes group elements used in $\mathbb{G}_1$ and $\mathbb{G}_2$. From the table it is clear that group elements of public key set $pk = (u, \Omega, \Upsilon, U_0, U_1, \{g_i\}_{i\in[1,2n]/\{n+1\}}, \{h_i\}_{i\in[1,2n]/\{n+1\}})$, signature set $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6)$ and witness set $W = (W_0, W_1, W_2, W_3, W_4)$ are now working in two different groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of asymmetric pairings, instead of working in a single group $\mathbb{G}$ of symmetric pairings. From the view point of used group elements, our proposed block-wise P-signatures use more group elements than Izabachène et al.'s construction. However, from the view point of Type-3 pairing implementation, elements in group $\mathbb{G}_1$ is operated on $E(\mathbb{F}_q)$ and elements in group $\mathbb{G}_2$ is operated on $E(\mathbb{F}_{q^k})$; hence our modification can get benefit from the elements working on relatively small field $E(\mathbb{F}_q)$. The detailed efficiency comparison will be discussed in Section 4.3.

Figure 4.1: Elements used in $\mathbb{G}_1$ and $\mathbb{G}_2$ of proposed block-wise P-signatures

Function	elements used in $\mathbb{G}_1$	elements used in $\mathbb{G}_2$
$F-signatures$		
$KeyGen$	g	h
$Sign$	$g, U_0, \{g_i\}_1^n$	h, u
$Verify$	$g, U_0, A, \Omega, \{g_i\}_1^n, \sigma_{21}, \sigma_4, \sigma_5, \sigma_6$	$h, u, U_1, \sigma_1, \sigma_{22}, \sigma_3$
$WitGen-EQ$	$\{g_i\}_1^n$	-
$WitGen-\overline{EQ}$	$g, g_{2n}, \{g_i\}_1^n$	h
$WitGen-IP$	$\{g_i\}_1^n, \{W_i\}_1^n$	-
$WitGen-\overline{IP}$	$g, g_{2n}, \{g_i\}_1^n, \{W_i\}_1^n$	h
$WitVerify-EQ$	g_1, σ_6, W	h, h_i, h_n
$WitVerify-\overline{EQ}$	$g_1, g_{n+1-i}, \sigma_6, W_1, W_2, W_3, W_4$	$h, h_1, h_i, h_n, h_{2n}, W_0$
$WitVerify-IP$	σ_6, W	$h, \{h_i\}_1^n$
$WitVerify-\overline{IP}$	$g_1, \sigma_6, W_1, W_2, W_3, W_4$	$h, h_i, h_{2n}, \{h_i\}_1^n, W_0$

4.2 Security consideration of our proposed block-wise P-signatures

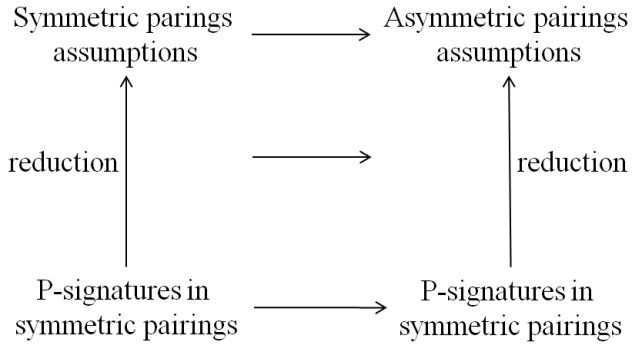
In this section we examine the security of our proposed block-wise P-signatures based on asymmetric pairings. Please refer to Figure 4.2, this graph illustrates the logic of proof reduction of our proposed block-wise P-signatures. Essentially, the left side of this graph shows the way to prove that the original block-wise P-signatures is secure. In Section 3.4, we have reviewed the attack model of block-wise P-signatures given by Izabachène et al. [4], which contains three types of forgeries attacks. The block-wise P-signatures based on symmetric pairings are proved secure against all three kinds of forgeries attacks; to be more specific, Type-I forgeries are proved secure on the security assumption of $q-HSDH$ problem while Type-II forgeries are proved secure on the assumption of $FlexDH$ problem. Moreover, all four different cases of Type-III forgeries are proved secure on the security assumption of $n-FlexDHE$ problem.

Due to the fact that our asymmetric block-wise P-signatures are essential extended from the original symmetric ones, the idea of defined attack models and proofs still work on our designed scheme. However, the security assumptions of $q-HSDH$, $FlexDH$ and $n-FlexDHE$ are defined in single group $\mathbb{G}$; [1] to well fit into our requirements in asymmetric pairings, we extend these definitions to asymmetric pairings and prove the equivalence of them.

Definition 4.1. *The* ***q-External Hidden Strong Diffie-Hellman problem (q-XHSDH)*** *This definition extends Definition 2.4 from group security assumption to asymmetric pair-*

[1] Please refer $q-HSDH$ problem to Definition 2.4, $FlexDH$ problem to Definition 2.5, and $n-FlexDHE$ problem to Definition 2.7.

Figure 4.2: Security reduction from symmetric pairings to asymmetric pairings



ings. given two group $\mathbb{G}_1$ and $\mathbb{G}_2$, and define bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. q-XHSDH assumes q-HSDH problem is hard both in $\mathbb{G}_1$ and $\mathbb{G}_2$

Lemma 4.1. *if q-HSDH is secure against forgery, then q-XHSDH is also secure against forgery.*

Proof sketch. Assume an adversary $\mathcal{A}$ want to solve $q - HSDH$ problem with the help of procedure $\mathcal{B}$ that can solve $q - XHSDH$ problem. Without loss of generality we make an assumption that the oracle of $\mathcal{A}$ is set with two group generator such that $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$; in other words, once procedure $\mathcal{B}$ requires a oracle query, oracle $\mathcal{A}$ will chooses $c_i \in \mathbb{Z}_p^*$ and answers $(g^{1/(\omega+c_i)}, g^{c_i}, u_1^{c_i})$ and $(h^{1/(\omega+c_i)}, h^{c_i}, u_2^{c_i})$ by asking his oracle twice. After q times' oracle queries, procedure $\mathcal{B}$ outputs a tuple $(g^{1/(\omega+c)}, g^c, u_1^c, h^{1/(\omega+c)}, h^c, u_2^c)$ where $c \neq c_i$. Then the adversary $\mathcal{A}$ outputs $(g^{1/(\omega+c)}, g^c, u_1^c)$ as a forgery to $q - HSDH$ problem.

Here we argue that making the oracle of $\mathcal{A}$ answer two tuples $(g^{1/(\omega+c_i)}, g^{c_i}, u_1^{c_i})$ and $(h^{1/(\omega+c_i)}, h^{c_i}, u_2^{c_i})$ will not increase the probability of success for procedure $\mathcal{B}$ to output the tuple $(g^{1/(\omega+c)}, g^c, u_1^c, h^{1/(\omega+c)}, h^c, u_2^c)$. The reason is that (g, u_1) and (h, u_2) are randomly chosen from $\mathbb{G}_1$ and $\mathbb{G}_2$, which means procedure $\mathcal{B}$ simultaneously and independently deals with two $q - HSDH$ problems. Therefore, the hardness of solving $q - XHSDH$ problem is equal to solving the problem of $q - HSDH$.

Definition 4.2. *The **Flexible External Diffie-Hellman problem (FlexXDH)**, given two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order p, $(g, g^a, g^b) \in \mathbb{G}_1^3$ and $(h, h^a, h^b) \in \mathbb{G}_2^3$ where $a, b \in \mathbb{Z}_p^*$. finding tuples $(g^\mu, g^{\mu a}, g^{\mu b})$ and $(h^\mu, h^{\mu a}, h^{\mu b})$ such that $\mu \neq 0$.*

Lemma 4.2. *if FlexDH is secure against forgery, FlexXDH is also secure against forgery.*

Proof sketch. Similarly let the adversary $\mathcal{A}$'s oracle with two group elements $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. $\mathcal{A}$ uses a procedure $\mathcal{B}$ which can solve the $FlexXDH$ problem, and answer procedure $\mathcal{B}$'s oracle queries by choosing $a_i, b_i \in \mathbb{Z}_p^*$ and response (g, g^{a_i}, g^{b_i}) and $(h, h^{a_i},$

h^{b_i}). After q times' queries, procedure $\mathcal{B}$ output $\{(g^{\mu}, g^{\mu a}, g^{\mu b}), (h^{\mu}, h^{\mu a}, h^{\mu b})\}$ such that $\mu \neq 0$. Finally the adversary outputs $(g^{\mu}, g^{\mu a}, g^{\mu b})$ as the answer to $FlexDH$ problem.

Definition 4.3. *The **Flexible External n-Diffie-Hellman Exponent Problem** (n-FlexXDHE), given two group $\mathbb{G}_1$ and $\mathbb{G}_1$ of order p, $(g, g_1, ..., g_n, g_{n+2}, ..., g_{2n}) \in \mathbb{G}_1^{2n}$, $(h, h_1, ..., h_n, h_{n+2}, ..., h_{2n}) \in \mathbb{G}_2^{2n}$ such that $g_i = g^{(\alpha^i)}$ and $h_i = h^{(\alpha^i)}$ where $\alpha \in \mathbb{Z}_p^*$ and $i \in [1, 2n] \setminus \{n+1\}$, compute triples $(g^{\mu}, g_{n+1}^{\mu}, g_{2n}^{\mu})$ and $(h^{\mu}, h_{n+1}^{\mu}, h_{2n}^{\mu})$ where $\mu \in \mathbb{Z}_p^*$, $g_{n+1} = g^{\alpha^{n+1}}$ and $h_{n+1} = h^{\alpha^{n+1}}$.*

Lemma 4.3. *if $n - FlexDHE$ is secure against forgery, $n - FlexXDHE$ is also secure against forgery.*

Proof sketch. Similarly let the adversary $\mathcal{A}$'s oracle with two group elements $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. $\mathcal{A}$ uses a procedure $\mathcal{B}$ which can solve the $n - FlexXDHE$ problem, and answer procedure $\mathcal{B}$'s oracle queries by choosing $\alpha_j \in \mathbb{Z}_p^*$ and response $(g, g_1, ..., g_n, g_{n+2}, ..., g_{2n}) \in \mathbb{G}_1^{2n}$ and $(h, h_1, ..., h_n, h_{n+2}, ..., h_{2n}) \in \mathbb{G}_2^{2n}$ such that $g_i = g^{(\alpha_j^i)}$ and $h_i = h^{(\alpha_j^i)}$. After q times' queries, procedure $\mathcal{B}$ output $\{(g^{\mu}, g_{n+1}^{\mu}, g_{2n}^{\mu}), (h^{\mu}, h_{n+1}^{\mu}, h_{2n}^{\mu})\}$ where $\mu \in \mathbb{Z}_p^*$, $g_{n+1} = g^{\alpha^{n+1}}$ and $h_{n+1} = h^{\alpha^{n+1}}$. Finally the adversary outputs $(g^{\mu}, g^{\mu a}, g^{\mu b})$ as the answer to $n - FlexDHE$ problem.

Theorem 4.1. *Our asymmetric block-wise P-signatures are proved secure on the hardness of q-XHSDH problem, FlexXDH problem and n-FXDHE problem.*

Proof sketch. We have extended the security assumptions of $q - HSDH$, $FlexDH$ and $n-FlexDHE$ to security assumptions of $q-XHSDH$, $FlexXDH$ and $n-FlexXDHE$, and prove the security of them. Therefore, our asymmetric pairing based block-wise P-signature can be proven to be secure against all three types of forgery attacks. The prove can be slightly modified from the original proofs provided by [4] and hence we will omit the proofs here. In this way, the reduction proof of right-hand side of Figure 4.2 can be constructed, and our proposed block-wise P-signatures based on asymmetric pairings can be proved secure.

4.3 Efficiency analysis of our proposed block-wise P-signatures

This section focuses on evaluating efficiency of our proposed asymmetric pairing based block-wise P-signatures both in theoretical part and in practical part. To evaluate theoretical performance of proposed asymmetric pairing based block-wise P-signatures, without loss of generality we calculate the group elements used in pairing computation. In terms of practical implementation, we implement block-wise P-signatures both in symmetric pairings and asymmetric pairings, and then measure the running time of them for efficiency comparison.

Figure 4.3: The number of group elements used in Type-1 pairing-based and Type-3 pairing-based block-wise P-signatures – F-signature part

Function	BWPS in Type-1 (# of elements in $\mathbb{G}$)	BWPS in Type-3 (# of elements in $\mathbb{G}_1$)	BWPS in Type-3 (# of elements in $\mathbb{G}_2$)
$F-signatures$			
$KeyGen$	1	1	1
$Sign$	n+3	n+2	2
$Verify$	n+12	n+8	6
$WitGen-EQ$	n	n	0
$WitGen-\overline{EQ}$	n+1	n+2	1
$WitGen-IP$	2n	2n	0
$WitGen-\overline{IP}$	2n+2	2n+2	1
$WitVerify-EQ$	6	3	3
$WitVerify-\overline{EQ}$	12	7	6
$WitVerify-IP$	n+3	2	n+1
$WitVerify-\overline{IP}$	n+8	6	n+4

4.3.1 Theoretical analysis of proposed block-wise P-signatures

Here we compare theoretical performance of block-wise P-signatures. Firstly let us examine the performance of F-unforgeable signatures in block-wise P-signatures. Please refer to Figure 4.3, roughly the result can be classified into three cases. The first case, such as key generation, signature generation, witness verification of $\mathcal{R}^{\overline{EQ}}$ and witness verification of $\mathcal{R}^{\overline{IP}}$, the Type-3 setting use one more group element, $h \in \mathbb{G}_2$, than the Type-1 setting. Next, the second case, such as signature verification, witness generation of $\mathcal{R}^{\overline{EQ}}$, and witness generation of $\mathcal{R}^{\overline{IP}}$, the asymmetric setting use two more group elements, $h, \sigma_{22} \in \mathbb{G}_2$, than the symmetric setting. The third case, such as witness generation of $\mathcal{R}^{EQ}$, witness generation of $\mathcal{R}^{IP}$, witness verification of $\mathcal{R}^{EQ}$, and witness verification of $\mathcal{R}^{IP}$, , Both the asymmetric setting and symmetric setting use the same number of group elements since the asymmetric setting only use one group generator, $g \in \mathbb{G}_1$ or $h \in \mathbb{G}_2$, and do not contain the computation of $\sigma_{21} \in \mathbb{G}_1$ and $\sigma_{22} \in \mathbb{G}_2$. In short, the efficiency of our designed F-unforgeable signature are theoretically competitive to the symmetric one.

In terms of signature proof generation and verification, please refer to Figure 4.4 and Fig 4.5, our designed block-wise P-signatures show significant improvement in theoretical analysis. The major advantage comes from modifying the essential component, Groth-Sahai proof [3] of signature proof generation, from DLIN-based structures to SXDH-based structures. It is evident that the SXDH-based Groth-Sahai proofs use less group elements than their DLIN-based counterpart. [2] Signature proof generation of block-wise

[2]Please see the used group elements comparison between DLIN setting and SXDH setting in Figure 3.3 and Figure 3.2.

Figure 4.4: The number of group elements used in Type-1 pairing-based block-wise P-signatures – Signature proof part[4]

Function	Size (# of elements in $\mathbb{G}$)	Generation (# of MultiExps)	Verification (# of Pairings)
Algorithm $SigProve_1$			
$\mathcal{R} = \mathcal{R}^{EQ}$	80	80· MultiExp	621· Pairing
$\mathcal{R} = \mathcal{R}^{\overline{EQ}}$	101	101· MultiExp	828· Pairing
Algorithm $SigProve_2$			
$\mathcal{R} = \mathcal{R}^{IP}$	65	66· MultiExp	558· Pairing
$\mathcal{R} = \mathcal{R}^{\overline{IP}}$	104	105· MultiExp	828· Pairing
$\mathcal{R} = \mathcal{R}^{EQ}$	77	77· MultiExp	477· Pairing
$\mathcal{R} = \mathcal{R}^{\overline{EQ}}$	107	107· MultiExp	846· Pairing

Figure 4.5: The number of group elements used in Type-3 pairing-based block-wise P-signatures – Signature proof part

Function	Size (# of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$)	Generation (# of MultiExps)	Verification (# of Pairings)
Algorithm $SigProve_1$			
$\mathcal{R} = \mathcal{R}^{EQ}$	63	63· MultiExp	276· Pairing
$\mathcal{R} = \mathcal{R}^{\overline{EQ}}$	77	77· MultiExp	368· Pairing
Algorithm $SigProve_2$			
$\mathcal{R} = \mathcal{R}^{IP}$	53	54· MultiExp	248· Pairing
$\mathcal{R} = \mathcal{R}^{\overline{IP}}$	83	84· MultiExp	368· Pairing
$\mathcal{R} = \mathcal{R}^{EQ}$	61	61· MultiExp	212· Pairing
$\mathcal{R} = \mathcal{R}^{\overline{EQ}}$	85	85· MultiExp	376· Pairing

P-signatures use three different kinds of equations, which are pairing product equations, linear pairing equations and linear multi-scalar multiplication equations. In DLIN setting, pairing product equations cost nine group elements, linear pairing equations cost three group elements, and linear multi-scalar multiplication equations cost two group elements. However, SXDH setting requires less to construct proofs: pairing product equations cost eight group elements, linear pairing equations cost two group elements, and linear multi-scalar multiplication equations cost one group elements. Such the change significant affect the theoretical efficiency of signature proof generation and verification.

For example, let us look the case of $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$. In DLIN setting, we firstly need to make $(3+7+5+1) = 16$ commitments and require $16 \cdot 3 = 48$ group elements in $\mathbb{G}$. And

then the proof generation contains three pairing product equations, eight linear pairing product equations and one linear multi-scalar multiplication equation, which totally cost $(3 \cdot 9 + 8 \cdot 3 + 1 \cdot 2) = 53$ group elements in $\mathbb{G}$. Therefore, constructing signature proofs totally cost $48 + 53 = 101$ group elements in $\mathbb{G}$. However, in SXDH setting, we firstly need to make $(3 + 8 + 5 + 1) = 17$ commitments and require $17 \cdot 2 = 34$ group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. And then the proof generation contains three pairing product equations, nine linear pairing product equations and one linear multi-scalar multiplication equation, which totally cost $(3 \cdot 8 + 9 \cdot 2 + 1 \cdot 1) = 43$ group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. Therefore, constructing signature proofs totally cost $34 + 43 = 77$ group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. The SXDH setting save 24 group elements than DLIN setting, which means nearly 23.76%'s enhancement. In fact, using less group elements means using less multi-exponential computation and multi-exponential computation take the major computation time in most cryptographic algorithms; hence this improvement is enormous. Overall in theoretical analysis, Please refer to Figure 4.6, our new signature proof generation based on SXDH can save nearly 20% of multi-exponential computation.

Figure 4.6: Efficiency comparison between DLIN-based and SXDH-based signature proof generation

Function	(# of MultiExp in DLIN-based proof	(# of MultiExp in SXDH-based proof	Improvement (%)
Algorithm $SigProve_1$			
$\mathcal{R} = \mathcal{R}^{EQ}$	80	63	21.25
$\mathcal{R} = \mathcal{R}^{\overline{EQ}}$	101	77	23.76
Algorithm $SigProve_2$			
$\mathcal{R} = \mathcal{R}^{IP}$	66	54	18.18
$\mathcal{R} = \mathcal{R}^{\overline{IP}}$	105	84	20
$\mathcal{R} = \mathcal{R}^{EQ}$	77	61	20.77
$\mathcal{R} = \mathcal{R}^{\overline{EQ}}$	107	85	20.56

In terms of signature proof verification, take $\mathcal{R} = \mathcal{R}^{EQ}$ for example, naive verification procedure needs 621 pairing computation whether in DLIN setting while SXDH setting only require 276 pairings. The amount of pairing computation between these two settings present a enormous difference.

Overall in theoretical analysis, our proposed block-wise P-signatures cost competitive group elements in the part of F-unforgeable signature and achieve significant improvement in the part of signature proof generation and verification.

4.3.2 Practical analysis of asymmetric block-wise P-signatures

After seeing the theoretical efficiency analysis of block-wise P-signatures, in this section, we implement block-wise P-signatures both based on Type-1 pairings and Type-3 pairings.

Our experiment environment uses computer with Intel Core i5-520M processor (3MB cache and 2.40 GHz) and 2.92 GB random access memory (RAM). The operation system is Microsoft Windows 7 32-bit. Moreover, we adopt MIRACL Crypto. SDK to do pairing computation.

MIRACL SDK Library.

MIRACL SDK Library [34], stands for **M**ulti-precision **I**nteger and **R**ational **A**rithmetic **C**ryptographic **L**ibrary, is provided by Certivox and is a powerful library for implementing cryptographic application. MIRACL library is well-known for its supports for state-of-the-art cryptographic primitives, including implementations of elliptic curves and bilinear pairings. In the implementation of bilinear pairings, MIRACL library supports Type-1 pairings and Type-3 pairings respectively. For Type-1 pairings, two kinds of pairing-friendly curves are provided, which are EtaT pairings over $GF(2^m)$ and Tate pairing over $GF(p)$ respectively. And both of two pairings implementation support security degree of AES-80 and AES-128. In terms of Type-3 pairings, there are five pairing-friend curves implemented in MIRACL library, which are

- Cocks-Pinch (CP)[35] curve supports security degree of AES-80.
- Miyaji-Nakabayashi-Takano (MNT)[36] curve supports security degree of AES-80.
- Barreto-Naehrig (BN)[37] curve supports security degree of AES-128 and AES-192.
- Kachisa-Schaefer-Scott (KSS)[38] curve supports security degree of AES-192.
- Barreto-Lynn-Scott (BLS)[39] curve supports security degree of AES-256.

From the above pairing curves provided by MIRACL library, it is evident that Type-3 pairings give more flexibility from different security degree. In fact, for security degree higher than AES-128 there exists no efficient curves for Type-1 pairings.

Experiment result

In our experiment, considering limitation of pairing curves provided by MIRACL library and practical security issue, we choose curves of Tate Pairing-128 and BN-128 respectively to implement Type-1 pairing based block-wise P-signatures and our proposed scheme. And to set a benchmark for every cases in block-wise P-signatures, we choose one message block for the experiment, even though our implemented block-wise P-signatures can support multiple message blocks. Moreover, to eliminate the random bias that could happen in our testing environment, our experiment repeats 20 runs and then do statistical analysis in IBM SPSS software. the detailed experiment results are collected in Appendix A.1 and Appendix A.2, and the standard deviation of our experiments show negligibly small for the performance of block-wise P-signatures based on Type-1 pairings and Type-3 pairings to be evidently distinguished.

Figure 4.7: Efficiency comparison between Type-1 pairing based and Type-3 pairing based block-wise P-signatures (using pairing curves BN-128)

ID	Function	BWPS in Type-1 (/sec.)	BWPS in Type-3 (/sec.)	Improvement (%)
	$F-signatures$			
1	$Sign$	0.3013	0.0492	83.67
2	$Verify$	1.2829	0.7039	45.13
3	$WitGen-EQ$	0.044	0.009	79.54
4	$WitGen-\overline{EQ}$	0.0141	0.0492	85.04
5	$WitGen-IP$	0.043	0.0101	76.51
6	$WitGen-\overline{IP}$	0.1002	0.0174	82.63
7	$WitVerify-EQ$	0.4273	0.2331	45.44
8	$WitVerify-\overline{EQ}$	1.2307	0.6985	43.24
9	$WitVerify-IP$	0.2671	0.1564	41.44
10	$WitVerify-\overline{IP}$	1.1901	0.6931	41.76
	$SigProve_1$			
11	$SigProve_1-EQ$	>189	4.6203	>97.55
12	$SigProve_1-\overline{EQ}$	>252	6.0987	>97.57
13	$SigVerify_1-EQ$	>252	13.4915	>94.64
14	$SigVerify_1-\overline{EQ}$	>336	18.4456	>94.51
	$SigProve_2$			
15	$SigProve_2-EQ$	>152	4.1008	>97.3
16	$SigProve_2-\overline{EQ}$	>224	6.1149	>97.27
17	$SigProve_2-IP$	>128	3.5716	>97.2
18	$SigProve_2-\overline{IP}$	>232	6.4235	>97.23
19	$SigVerify_2-EQ$	>209	12.1482	>94.18
20	$SigVerify_2-\overline{EQ}$	>308	18.4185	>94.01
21	$SigVerify_2-IP$	>176	10.7983	>93.86
22	$SigVerify_2-\overline{IP}$	>319	18.8304	>94.09

The experiment results are collected in Figure 4.7. For the F-unforgeable signatures of block-wise P-signatures, Type-3 pairing-based implementation shows enormous efficiency while comparing with Type-1 pairing-based implementation. From theoretical analysis of F-unforgeable in Figure 4.3, it is clear that Type-3 construction costs slightly more group elements than Type-1 construction; however, the experiment shows extreme efficiency of Type-3 construction. Here we would like to discuss the efficiency of F-unforgeable signatures in two groups. The first group contains signature generation $Sign$ and four cases of witness generation $WitGen-EQ$, $WitGen-\overline{EQ}$, $WitGen-IP$ and $WitGen-\overline{IP}$. The similarity of functions in the first group is that the majority of their computation is on multiplication of group elements. From the experiment result, it is clear that functions

in the first group which are ported to Type-3 construction can save approximately 76% to 85%'s running time, which implicitly illustrates elements multiplication in Type-3 pairings is enormously efficient while comparing to Type-1 pairings. The second group contains signature verification $Verify$ and four cases of witness verification $WitVerify - EQ$, $WitVerify - \overline{EQ}$, $WitVerify - IP$ and $WitVerify - \overline{IP}$. These functions take their major time on doing pairing computation. The experiment result shows transferring F-unforgeable signatures from Type-1 to Type-3 can save nearly 41% to 45%'s running time, which is also a great improvement. The result also implicitly illustrates that doing pairing computation in Type-3 pairings is twice as fast as doing pairing computation in Type-1 pairings. In short, even though it increases several group elements while porting F-unforgeable signatures to Type-3 pairings, efficiency can gain significant benefit from the porting work.

In terms of signature proof generation and verification, Type-3 pairing-based implementation shows tremendous efficiency while comparing with Type-1 pairing-based implementation. The efficiency can be analyzed from a general view, which is to compare the theoretical analysis with practical analysis. From Figure 4.4 and Figure 4.5 we have understood the fundamental techniques of signature proof generation and verification is quite different; Type-1 construction is based on DLIN-based Groth-Sahai proofs while Type-3 construction is based on SXDH-based Groth-Sahai proofs. It is clear that SXDH-based Groth-Sahai proofs require less group elements than DLIN-based one; therefore, Type-3 based signature proof generation and verification are inherently at advantageous position, approximately 20% faster than Type-1 construction in theoretical analysis. Moreover, constructing signature proof and doing verification in block-wise P-signature also inherently contains huge number of element multiplication and pairing computation, which means a great deal of running time in generating proof and verification. From our experiment of even only one message block, signature proof generation and verification take at least 2 minutes to generate a signature proof of $SigProve_2 - IP$ and take at least 3 minutes to do verification of $SigVerify_2 - IP$, which is not an affordable running time in practical application. However, Type-3 construction shows acceptable running time in signature proof generation and verification, which take about 3.5 seconds in signature proof generation of $SigProve_2 - IP$ and about 10.7 seconds in verification procedure. Overall in signature proof generation and verification, the practical experiment result shows in Figure 4.7 is consistent with theoretical analysis shows in Figure 4.6; moreover, Type-3 construction dramatically extend the advantage of efficiency to approximately $93\% - 97\%$ while comparing with Type-1 construction.

Chapter 5

Optimization of Block-wise P-signatures

We have successfully redesigned block-wise P-signatures based on asymmetric pairings, and our proposed new scheme achieves significant improvement in efficiency. Nonetheless, the efficiency of our designed block-wise P-signatures can be further improved by applying optimization techniques in our design. In this chapter we describe the techniques we use to improve asymmetric pairing based block-wise P-signatures and measure the performance of them.

In Chapter 2.4 we have reviewed two optimization techniques, which are fixed argument optimization and calculation of pairing product respectively. These two optimization techniques describe how the fixed elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ result in performance difference. Here we focuses on fixed argument optimization and conclude them into three different cases. The first case is that if both arguments $X \in \mathbb{G}_1$ and $Y \in \mathbb{G}_2$ are fixed, the power of pairing product $e(X,Y)^z$ for some $z \in \mathbb{Z}_p^*$ can be precomputed. The second case shows that if the left-hand side argument $X \in \mathbb{G}_1$ is fixed, the pairing computation $e(X,\cdot)$ can be precomputed. The third case is that if the left-hand side argument $X \in \mathbb{G}_1$ is fixed, the computation of group element multiplication X^z for some $z \in \mathbb{Z}_p^*$ can be precomputed. With these three observation, we try to improve our proposed block-wise p-signatures.

5.1 Optimization of our proposed block-wise P-signature

To apply these three optimization techniques in block-wise p-signatures, it is necessary to examine each function algorithm to find out the repeated pattern appeared in pairing computation, so that our proposed scheme can be further improved by the technique of fixed argument optimization. After fine tuning every function algorithms, an optimized asymmetric pairing based block-wise P-signature is described as follows.

Figure 5.1: Optimization example – Signature verification of block-wise P-signatures before/after optimization

BWPS in Type-3 before optimization	BWPS in Type-3 after optimization
$e(\Upsilon, h) = e(\Omega \cdot \sigma_{21}, \sigma_1)$	$e(g, \Upsilon) = e(\Omega \cdot \sigma_{21}, \sigma_1)$,
$e(\sigma_{21}, u) = e(g, \sigma_3)$	$e(\sigma_{21}, u) = e(g, \sigma_3)$,
$e(\sigma_4, h) = e(U_0, \sigma_{22}) \cdot e(\sigma_5, U_1)$	$e(g, \sigma_4) = e(\sigma_{21}, U_0) \cdot e(U_1, \sigma_5)$,
$e(\sigma_5, h) = e(\sigma_6, \sigma_{22})$	$e(g, \sigma_5) = e(\sigma_{21}, \sigma_6)$,
$e(\sigma_{21}, \sigma_{22}^{-1}) = 1_{\mathbb{G}_T}$	$e(\sigma_{21}, \sigma_{22}^{-1}) = 1_{\mathbb{G}_T}$,

Optimization summary.

Our naively porting to type-3 pairing based block P-signatures leaves a significant space for optimization. The first observation is that group generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$ are repeatedly used in the verification procedure, including signature verification, witness verification and proof verification, which leaves an opportunity for argument pre-computation. However, in implementation of Type-3 pairings $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, pairing computation of fixed elements in $\mathbb{G}_1$ can be sped up with fixed argument pre-computation technique. With this observation, we could align all equations in verification procedures to the form of fixed left-hand side argument. For example in signature verification, please refer to Figure 5.1, $e(\sigma_4, h)$ in the third equation and $e(\sigma_5, h)$ in the fourth equation can be transferred into the form of $e(g, \sigma_4)$ and $e(g, \sigma_5)$, in this way the form of $e(g, \cdot)$ and can be precomputed with fixed argument optimization in implementation. Similarly the case $e(\cdot, \sigma_{22})$ can be also aligned as $e(\sigma_{21}, \cdot)$. However, since our working environment is in asymmetric pairings, which group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ cannot be arbitrarily exchanged, it is necessary to reconsider in which group arguments work. Fortunately the public key extension of our proposed block-wise P-signatures provides us a flexible way to decide in which group the argument work with extra burden; hence the algorithm re-adjusting can be easily achieved.

The second issue is to decide which pairing form, $e(A = g^a, B = h^b)$ or $e(B = g^b, A = h^a)$, can receive most efficiency enhancement for our proposed block-wise P-signatures. To achieve optimized performance of pairing pre-computation, it is critical to identify the group elements that are frequently used for the computation of element multiplication on elliptic curves; so that we can decide in which group the arguments work. It is obvious that generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$ are most frequently used for element multiplication. The generation of public key set $g_i = g^{(\alpha^i)}$ for $i \in [1, ..., n, n+2, ...2n]$ and $h_i = h^{(\alpha^i)}$ for $i \in [1, ..., n, n+2, ...2n]$ are clear examples. In fact, Among all arguments in the optimized block-wise P-signatures, we found arguments g, h, g_1, h_1 and h_{2n} could be the most used arguments for all computation. Therefore, these arguments can be precomputed in implementation.

The last optimization technique is to find pairing form where arguments both in $\mathbb{G}_1$ and $\mathbb{G}_2$ are fixed, so that the power of pairing can be precomputed. In our examination

there shows only one case $e(g_1, h_n)^{x_i}$ where x_i is the $i'th$ message block among all functions is repeatedly used; With this observation, we can conclude the powering pre-computation of $e(g_1, h_n)$ show less efficiency improvement for our proposed block-wise P-signatures.

Our proposed optimized block-wise P-signatures.

With the above three observations we readjust our Type-3 pairing based block-wise P-signatures.

BWPSigSetup(1^k). Taken security parameter k, this function generates $params = (n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, h, e, f)$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order p and generated by generators g and h, e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a bilinear map and $\mathcal{U}=(\mathcal{U}_1, \mathcal{U}_2)$ is common reference string (CRS) for SXDH-based Groth-Sahai proof.

BWPKeyGen($params$). Choosing $\gamma, \omega, \alpha, \beta \in \mathbb{Z}_p$, $u \in \mathbb{G}_2$, $U_0 \in \mathbb{G}_2$, computing $\Omega = g^{\omega}$, $\Upsilon = h^{\gamma}$, $U_1 = g^{\beta}$, $g_i = g^{(\alpha^i)}$ for $i \in [1, ..., n, n+2, ...2n]$ and $h_i = h^{(\alpha^i)}$ for $i \in [1, ..., n, n+2, ...2n]$. The secret-key $sk = (\gamma, \omega, \beta)$ and public-key $pk = (u, \Omega, \Upsilon, U_0, U_1, \{g_i\}_{i\in[1,2n]/\{n+1\}}, \{h_i\}_{i\in[1,2n]/\{n+1\}})$.

BWPSign($params, sk, m$). Taken message $m = (m_1, ..., m_n)$, the signer firstly chooses $r \in \mathbb{Z}_p$ and computes $V = h^r \cdot \prod_{j=1}^{n} h_{n+1-j}^{m_j} = h_n^{m_1} \cdots h_1^{m_n} \cdot h^r$. Secondly the signer chooses $c \in \mathbb{Z}_p$ and computes $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$ where

$$\begin{aligned} \sigma_1 &= h^{\gamma/\omega+c}, \\ \sigma_{21} &= g^c, \\ \sigma_{22} &= h^c, \\ \sigma_3 &= u^c, \\ \sigma_4 &= (U_0 \cdot V^{\beta})^c, \\ \sigma_5 &= V^c, \\ \sigma_6 &= V \end{aligned} \tag{5.1}$$

BWPVerify($params, pk, m, \sigma$). Receiving message $m = (m_1, ..., m_n)$ and $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$, the verifier accepts the signature if $\sigma_6 = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_j}$ and

$$\begin{aligned} & e(g, \Upsilon) = e(\Omega \cdot \sigma_{21}, \sigma_1), \\ & e(\sigma_{21}, u) = e(g, \sigma_3), \\ & e(g, \sigma_4) = e(\sigma_{21}, U_0) \cdot e(U_1, \sigma_5), \\ & e(g, \sigma_5) = e(\sigma_{21}, \sigma_6), \\ & e(\sigma_{21}, \sigma_{22}^{-1}) = 1_{\mathbb{G}_T}, \end{aligned} \tag{5.2}$$

BWPWitGen($params, \mathcal{R}, i, m, X, \sigma$). Taken relation $\mathcal{R}$, index i, $m = (m_1, ..., m_n)$, $X = (x_1, ..., x_n)$ and $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$,

1. If $\mathcal{R} = \mathcal{R}^{EQ}$ and $m_i = x_i$ where $i \in [1, n]$, compute the witness

$$W = h_i^r \cdot \prod_{j=1, j\neq i}^{n} h_{n+1-j+i}^{m_j}. \tag{5.3}$$

2. If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$ and $m_i \neq x_i$, where $i \in [1, n]$, compute the witness $W = (W_0, W_1, W_2, W_3, W_4)$ where

$$\begin{aligned} W_0 &= g^{1/(m_i - x_i)}, \\ W_1 &= h_1^{(m_i - x_i)}, \\ W_2 &= h^{(m_i - x_i)}, \\ W_3 &= h_{2n}^{(m_i - x_i)} \\ W_4 &= h_i^r \cdot \prod_{j=1, j\neq i}^{n} h_{n+1-j+i}^{m_j} \end{aligned} \tag{5.4}$$

3. If $\mathcal{R} = \mathcal{R}^{IP}$, $i = 0$, and , $m \cdot X = 0$, compute the witness

$$\begin{aligned} W &= \prod_{i=1}^{n} W_i^{x_i} \text{ for each} \\ W_i &= h_i^r \cdot \prod_{j=1, j\neq i}^{n} h_{n+1-j+i}^{m_j} \end{aligned} \tag{5.5}$$

4. If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$, $i = 0$, and , $m \cdot X \neq 0$, compute the witness $W = (W_0, W_1, W_2, W_3, W_4)$ where

$$\begin{aligned} W_0 &= g^{1/(m_i - x_i)}, \\ W_1 &= h_1^{(m_i - x_i)}, \\ W_2 &= h^{(m_i - x_i)}, \\ W_3 &= h_{2n}^{(m_i - x_i)} \\ W_4 &= \prod_{i=1}^{n} W_{4,i}^{x_i} \text{ for each} \\ W_{4,i} &= h_i^r \cdot \prod_{j=1, j\neq i}^{n} h_{n+1-j+i}^{m_j} \end{aligned} \tag{5.6}$$

BWPWitVerify$(params, pk, i, X, W, \sigma)$**.** Taken public-key pk, index i, $X = (x_1, ..., x_n)$, witness W and a valid signature $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$,

1. If $\mathcal{R} = \mathcal{R}^{EQ}$ and $i \in [1, n]$, output 1 if

$$e(g_i, \sigma_6) = e(g_1, h_n)^{x_i} \cdot e(g, W)$$

2. If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$ and $i \in [1, n]$, parse $W = (W_0, W_1, W_2, W_3, W_4)$ and check if the following equations hold.

$$\begin{aligned} & e(g_i, \sigma_6 \cdot h_{n+1-i}^{-x_i}) = e(g_n, W_1) \cdot e(g, W_4) \\ & e(W_0, W_1) = e(g_1, h), \\ & e(g, W_1) = e(g_1, W_2), \\ & e(g_{2n}, W_1) = e(g_1, W_3), \end{aligned} \tag{5.7}$$

3. If $\mathcal{R} = \mathcal{R}^{IP}$ and $i = 0$, output 1 if

$$e(g, W) = e(\prod_{i=1}^{n} g_i^{x_i}, \sigma_6)$$

4. If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$ and $i = 0$, parse $W = (W_0, W_1, W_2, W_3, W_4)$ and check if the following equations hold.

$$\begin{aligned} & e(\prod_{i=1}^{n} g_i^{x_i}, \sigma_6) = e(g_n, W_1) \cdot e(g, W_4) \\ & e(W_0, W_1) = e(g_1, h), \\ & e(g, W_1) = e(g_1, W_2), \\ & e(g_{2n}, W_1) = e(g_1, W_3), \end{aligned}$$

BWPSigObtain$(pk, M_{part1}, open_{part1}) \longleftrightarrow$ **MBPSigIssue**$(sk, V_{part1}, M_{part2})$**.** this interactive protocol provides users to acquire signatures from some organizations.

1. the user chooses $r_{part1} \in \mathbb{Z}_p$ and computes a commitment $V_{part1} = h^{r_{part1}} \cdot \prod_{j=1}^{n_1} h_{n+1-j}^{m_j}$ for message M_{part1}=(m_1, ..., m_{n_1}). The the user sets $open_{part1}$ = M_{part1}=(m_1, ..., m_{n_1}, r') and computes a witness-indistinguishable proof for knowledge of M_{part1} such that $V_{part1} = h^{r_{part1}} \cdot \prod_{j=1}^{n_1} h_{n+1-j}^{m_j}$.
2. the signature issuer chooses $r^{part2}, c \in Z_p$ and computes

$$\begin{aligned} & V = V_{part1} \cdot \prod_{j=n+1}^{n} h_{n+1-j}^{m_j} \\ & \sigma_1 = h^{\gamma/(\omega+c)}, \\ & \sigma_{21} = g^c, \\ & \sigma_{22} = h^c, \\ & \sigma_3 = u^c, \\ & \sigma_4 = (U_0(V \cdot g^{r_{part2}})^{\beta})^c, \\ & \sigma_5 = (V \cdot g^{r_{part2}})^c, \\ & \sigma_6 = V \cdot g^{r_{part2}} \end{aligned} \tag{5.8}$$

and return $\bar{\sigma} = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r_{part2})$.

3. the user outputs $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$ where $r = r_{part1} + r_{part2}$.

BWPProve1$(params, pk, i, S = \{i\}, m, X, \sigma)$**.** Taken $m = (m_1, ..., m_n)$, $X = (x_1, ..., x_n)$ and $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6, r)$. Firstly this proof needs to compute $\{ C_{X_{t,j}} = \text{GSCom}(X_{t,j}, open_{x_{t,j}}) \}_{t\in S, j\in\{1,2,3\}}$ where $\{(X_{t,1}, X_{t,2}, X_{t,2}) = (h_1^{x_t}, h^{x_t}, h_{2n}^{x_t})\}_{t\in S}$. Secondly this proof computes commitments for $\{C_{\sigma_j} = GSCom(\sigma_j, open_{\sigma_j})\}_{j=1...7}$. Thirdly this proof chooses $\sigma_7 = \Upsilon \in G$ and $\theta = 1 \in Z_p$, computes $C_{\sigma_7} = \text{GSCom}(\sigma_7, open_{\sigma_7})$ and $C_{\sigma_\theta} = \text{GSCom}(\sigma_\theta, open_{\sigma_\theta})$, and then constructs a NIZK proof π_{theta} proving the following equations.

$$\begin{aligned} & e(g, \sigma_7) = e(\Omega \cdot \sigma_{21}, \sigma_1) \wedge \\ & e(\sigma_{21}, u) = e(g, \sigma_3) \wedge \\ & e(g, \sigma_4) = e(\sigma_{21}, U_0) \cdot e(U_1, \sigma_5) \wedge \\ & e(g, \sigma_5) = e(\sigma_{21}, \sigma_6) \wedge \\ & e(\sigma_{21}, \sigma_{22}^{-1}) = 1_{\mathbb{G}_T} \wedge \\ & \theta = 1 \wedge \\ & e(g^\theta, \Upsilon/\sigma_7) = 1_{\mathbb{G}_T} \end{aligned} \tag{5.9}$$

There are two cases for this proof.

- If $\mathcal{R} = \mathcal{R}^{EQ}$, this proof computes commitment $C_W = \text{GSCom}(W, open_W)$ where $W = \text{BWWitGen}(pk, R^{EQ}, i, m, X, \sigma)$ and then computes proofs π_{x_i}, $\{\pi_{X_{t,j}}\}_{t\in S, j=1,2}$ such that

$$\begin{aligned} & e(g_i, \sigma_6) = e(g_n, X_{i,1}) \cdot e(g, W) \wedge \\ & e(g_1, X_{i,2}) \cdot e(g, X_{i,1}) \wedge \\ & e(g_{2n}, X_{i,2}) \cdot e(g, X_{i,3}) \end{aligned} \tag{5.10}$$

  The proof is $\pi = (\{C_{x_{t,j}}\}_{t\in S, j=1,2,3}, \{C_{\sigma_j}\}_{j=1,...,8}, C_W, C_\theta, \pi_\theta, \pi_{x_i}, \{\pi_{x_{t,j}}\}_{t\in S, j=1,2,})$.

- If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$, this proof computes commitments $\{C_{W_j}\}_{j=0,...,4}$ to (W_0, W_1, W_2, W_3, W_4) and generates proofs π_{x_i}, π_W satisfying

$$\begin{aligned} & e(g_i, \sigma_6) \cdot e(g_n, X_{i,1})^{-1} = e(g_n, W_1) \cdot e(g, W_4) \wedge \\ & e(W_0, W_1) = e(g_1, h) \wedge \\ & e(g, W_1) \cdot e(g_1, W_2) \wedge \\ & e(g_{2n}, W_1) = e(g_1, W_3) \end{aligned} \tag{5.11}$$

  and also generates proof $\{\pi_{X_{t,j}}\}_{t\in S, j=1,2}$. The final proof is π=($\{C_{x_{t,j}}\}_{t\in S, j=1,2,3}$, $\{C_{\sigma_j}\}_{j=1,...,8}$, $\{C_{W_j}\}_{j=0,...,4}$, C_θ,π_θ, π_{x_i}, $\{\pi_{x_{t,j}}\}_{t\in S, j=1,2}$).

BWPProve2$(params, pk, R, i, m, , X, \sigma)$**.** Similar to BWPProve1, firstly this proof computes commitments $\{C_{\sigma_j}\}_{j=1,...,8}$, C_θ, as well as proofs π_θ and π_{x_i}. Then there are four cases in this proof.

- If $\mathcal{R} = \mathcal{R}^{IP}$, compute a commitment $C_W = \text{GSCom}(W, open_W)$ where $W = \text{BWPWitGen}(pk, R^{IP}, 0, m, X, \sigma)$ and compute a proof π_X such that

$$e(\prod_{j=1}^{n} g_j^{x_j}, \sigma_6) = e(g, W) \tag{5.12}$$

Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,\ldots,8},\ C_W,\ C_\theta, \pi_\theta,\ \pi_X\ )$.

- If $\mathcal{R} = \mathcal{R}^{\overline{IP}}$, compute $C_g = \text{GSCom}(g,\ open_g)$, $\{C_{W_j}\}_{j=0,\ldots,4}$ where $W_j = \text{BWPWitGen}(pk,\ \mathcal{R}^{\overline{IP}},\ 0,\ m,\ X,\ \sigma)$, and select $T = h \in mathbbG_2$ and compute a proof π_X such that

$$\begin{aligned} &e(W_0, W_1) = e(g_1, T) \wedge \\ &e(g, W_1) = e(g_1, W_2) \wedge \\ &e(g_{2n}, W_1) = e(g_1, W_3) \wedge \\ &e(g^\theta, T/h) = 1_{\mathbb{G}_T} \wedge \\ &e(\prod_{j=1}^{n} g_j^{x_j}, \sigma_6) = e(g_n, W_1) \cdot e(g, W_4) \end{aligned} \tag{5.13}$$

Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,\ldots,8},\ \{C_{W_j}\}_{j=0,\ldots,4},\ C_\theta, \pi_\theta,\ \pi_X\ )$.

- If $\mathcal{R} = \mathcal{R}^{EQ}$ compute a commitment $C_W = \text{GSCom}(W, open_W)$ where $W = \text{BWPWitGen}(pk, R^{EQ}, i, m, X, \sigma)$, $C_{X_i} = \text{GSCom}(X_i,\ open_{X_i})$ where $X_i = h_1^{x_i}$, and compute proofs π_W and π_X such that

$$\begin{aligned} &e(g_i, \sigma_6) = e(g_n, X_i) \cdot e(g, W) \wedge \\ &e(g^\theta, X_i / h_1^{x_i}) = 1_{\mathbb{G}_T} \end{aligned} \tag{5.14}$$

Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,\ldots,8},\ C_W,\ C_{X_i},\ C_\theta, \pi_\theta,\ \pi_W, \pi_X\ )$.

- If $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$, compute $\{\ C_{W_j}\ \}_{j=0,\ldots,4}$ where $W_j = \text{BWPWitGen}(pk,\ R^{\overline{EQ}},\ 0,\ m,\ X,\ \sigma)$, $C_{X_i} = \text{GSCom}(X_i,\ open_{X_i})$ where $X_i = g_1^{x_i}$, and compute proofs $\pi_{X_{i,W}}$, $\{\pi_{W_j}\}_{j=1,\ldots,3}$ π_{X_i}, π_g such that

$$\begin{aligned} &e(g_i, \sigma_6) \cdot e(g_n, X_i)^{-1} = e(g_n, W_1) \cdot e(g, W_4) \wedge \\ &e(W_0, W_1) = e(g_1, h) \wedge \\ &e(g, W_1) = e(g_1, W_2) \wedge \\ &e(g_{2n}, W_1) = e(g_1, W_3) \wedge \\ &e(g^\theta, X_i / h_1^{x_i},) = 1_{\mathbb{G}_T} \wedge \\ &e(g^\theta, T/h) = 1_{\mathbb{G}_T} \end{aligned} \tag{5.15}$$

Finally, the proof $\pi = (\ \{C_{\sigma_j}\}_{j=1,\ldots,8},\ \{C_{W_j}\}_{j=0,\ldots,4},\ C_{X_i}, C_\theta,\ \pi_\theta,\ \pi_{X_{i,W}},\ \{\pi_{W_j}\}_{j=1,\ldots,3},\ \pi_\theta\ )$.

BWPEqCommProve($params, m, open, open'$)**.** Taken two opens $open_X$ and $open_Y$, two commitments C_X and C_Y can be respectively computed by C_X=GSCom(X, $open_X$) and C_Y=GSCom(Y, $openY$). To prove $X = Y$, we can construct an non-interactive zero-knowledge proof $C_X \cdot C_Y = 1_{\mathbb{G}_T}$ based on SXDH assumption.

The result after optimization

Figure 5.2: Elements used in $\mathbb{G}_1$ and $\mathbb{G}_2$ of our proposed optimized block-wise P-signatures

Function	elements in $\mathbb{G}_1$	elements in $\mathbb{G}_2$
$F - signatures$		
$KeyGen$	g	h
$Sign$	$g, U_0, \{g_i\}_1^n$	h, u
$Verify$	$g, U_0, U_1, \Omega, \{g_i\}_1^n, \sigma_{21}, \sigma_4$	$h, A, u, \sigma_1, \sigma_{22}, \sigma_3, \sigma_5, \sigma_6$
$WitGen - \overline{EQ}$	-	$\{h_i\}_1^n$
$WitGen - \overline{EQ}$	g	$h, h_{2n}, \{h_i\}_1^n$
$WitGen - IP$	-	$\{h_i\}_1^n, \{W_i\}_1^n$
$WitGen - \overline{IP}$	g	$h, h_{2n}, \{h_i\}_1^n, \{W_i\}_1^n$
$WitVerify - \overline{EQ}$	g, g_1, g_i	σ_6, h_n, W
$WitVerify - \overline{EQ}$	$g_1, g_i, g_n, g_{2n}, \sigma_6, W_0$	$h, h_{n+1-i}, W_1, W_2, W_3, W_4$
$WitVerify - IP$	$g, \{g_i\}_1^n$	σ_6, W
$WitVerify - \overline{IP}$	$g_1, g_i, g_{2n}, \{g_i\}_1^n, W_0$	$h, \sigma_6, W_1, W_2, W_3, W_4$

Let us examine the difference between our proposed two block-wise P-signatures. Please compare Figure 4.1 and Figure 5.2, which collect elements used in two group $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The first observation is that these two schemes use the same group elements, which means the fundamental idea of asymmetric-based block-wise P-signatures remains unchanged and hence our optimization may not jeopardize the security level of our proposed scheme. Secondly, it is also clear that the optimized block-wise P-signatures dispatch more elements of signature set $\sigma = (\sigma_1, \sigma_{21}, \sigma_{22}, \sigma_3, \sigma_4, \sigma_5, \sigma_6)$ and witness set $W = (W_0, W_1, W_2, W_3, W_4)$ to group $\mathbb{G}_2$ while comparing with the naive porting version of block-wise P-signatures introduced in last chapter. The advantage of this modification is that while doing verification procedure, we can construct maximal repeated pattern of fixed left-hand side argument in pairing computation, which can be pre-computed in implementation. Lastly, our modification moves the dynamic elements such as σ and W to the group $\mathbb{G}_2$ of pairing computation, which can hence get the benefit of relatively small field computation E($\mathbb{F}_q$) while comparing to E($\mathbb{F}_{q^k}$) of group $\mathbb{G}_1$ computation.

In our modification, we are also interested in knowing how many repeated pairing patterns we can have to do pairing pre-computation. Figure 5.3 collects all the repeated appearance of pairing form after the optimization. In the first view of this table, it is clear that except for $WitVerify - IP$, all verification procedure can get benefit from the

Figure 5.3: Collection of repeated pairing forms in block-wise P-signatures before/after optimization

Function	Pairing form	# before optimization	# after optimization
$Verify$	$e(g,\cdot)$	1	4
	$e(\sigma_{21},\cdot)$	1	4
$WitVerify-EQ$	$e(g,\cdot)$	0	1
	$e(g_1,\cdot)$	1	2(Max)
$WitVerify-\overline{EQ}$	$e(g,\cdot)$	0	2
	$e(g_1,\cdot)$	1	5(Max)
	$e(g_n,\cdot)$	0	2(Max)
$WitVerify-IP$	$e(g,\cdot)$	0	1
$WitVerify-\overline{IP}$	$e(g,\cdot)$	0	2
	$e(g_1,\cdot)$	1	3
$SigVerify_1-EQ$	$e(g,\cdot)$	1	8
	$e(\sigma_{21},\cdot)$	1	4
	$e(g_1,\cdot)$	0	2(Max)
	$e(g_n,\cdot)$	0	2(Max)
$SigVerify_1-\overline{EQ}$	$e(g,\cdot)$	1	7
	$e(\sigma_{21},\cdot)$	1	4
	$e(g_1,\cdot)$	1	3
	$e(g_n,\cdot)$	0	2
$SigVerify_2-EQ$	$e(g,\cdot)$	2	7
	$e(\sigma_{21},\cdot)$	1	4
	$e(g_n,\cdot)$	1	2(Max)
$SigVerify_2-\overline{EQ}$	$e(g,\cdot)$	2	10
	$e(\sigma_{21},\cdot)$	1	4
	$e(g_1,\cdot)$	1	4(Max)
	$e(g_n,\cdot)$	0	2
$SigVerify_2-IP$	$e(g,\cdot)$	1	5
	$e(\sigma_{21},\cdot)$	1	4
$SigVerify_2-\overline{IP}$	$e(g,\cdot)$	2	8
	$e(\sigma_{21},\cdot)$	1	4
	$e(g_1,\cdot)$	1	2

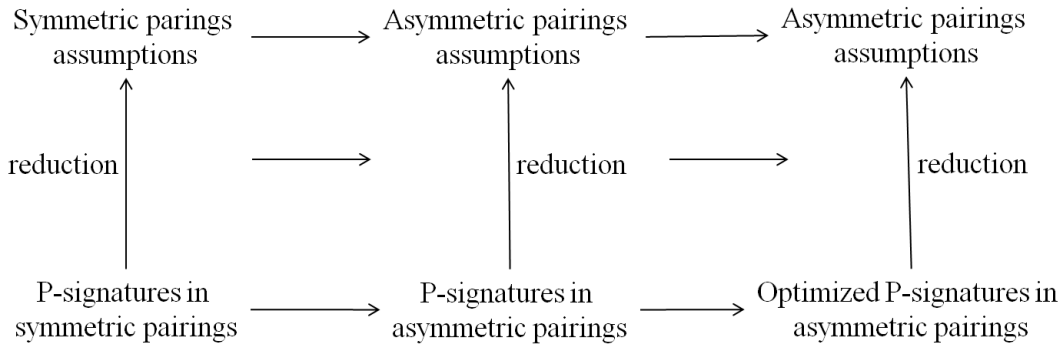
optimization. It is desired to notice that the number of pairing form $e(g,\cdot)$ is dramatically increased, which indirectly illustrates the advantage of our optimization on asymmetric pairing based block-wise P-signatures. Other pairing forms such as $e(g_1,\cdot)$, $e(g_n,\cdot)$ and $e(\sigma_{21}$ are also gain the benefit from our optimization. In short, these repeated pairing pattern can be precomputed and speed up the computation of our optimized block-wise

P-signature scheme.

5.1.1 Security consideration of optimized P-signatures

To optimize the proposed block-wise P-signatures we have significantly readjusted the relative position of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. It is necessary to examine whether the modified block-wise P-signature scheme is still secure against forgery attack.

Figure 5.4: Security reduction from asymmetric pairings P-signatures to optimized P-signatures



In Section 4.2 we define the security assumption of $q-XHSDH$ problem, $FlexXDH$ problem and $n-XFDHE$ problem for asymmetric pairings and all these three security are essentially extended from $q-HSDH$ problem, $FlexDH$ problem and $n-FDHE$ problem of single group $\mathbb{G}$ into asymmetric pairing groups $\mathbb{G}_1$ and $\mathbb{G}_2$. With these security assumption we can reduce the security of our proposed block-wise P-signatures to $q-XHSDH$ problem, $FlexXDH$ problem and $n-XFDHE$ problem.

In terms of our optimization on block-wise P-signatures, we also need to carefully examine if our modification jeopardize the security. Please refer to 5.4, it is necessary to examine whether the security assumption of $q-XHSDH$ problem, $FlexXDH$ problem and $n-XFDHE$ problem still works in our optimized block-wise P-signatures. Firstly, the defined $q-XHSDH$ problem are essentially simultaneously running the $q-HSDH$ with generator $g \in \mathbb{G}_1$ and generator $h \in \mathbb{G}_2$ and try to find a tuple $(g^{1/(\omega+c)}, g^c, u_1^c, h^{1/(\omega+c)}, h^c, u_2^c)$ given $(g^{1/(\omega+c_i)}, g^{c_i}, u_1^{c_i})$ and $(h^{1/(\omega+c_i)}, h^{c_i}, u_2^{c_i})$ where $c_i \in \mathbb{Z}_p^*$. This security assumption is essentially symmetric except for the underlying group elements g, u_1, h, and u_2; in other words, the exponential argument ω and c_i are the same. Such definition give us convenience to transfer one secret from group $\mathbb{G}_1$ to group $\mathbb{G}_2$. For example, please refer to first equation in Figure 5.1.

$$\begin{aligned} e(\Upsilon, h) &= e(\Omega \cdot \sigma_{21}, \sigma_1)\text{before optimization} \\ e(g, \Upsilon) &= e(\Omega \cdot \sigma_{21}, \sigma_1)\text{after optimization} \end{aligned} \tag{5.16}$$

Our modification move the fixed argument γ from $e(\Upsilon = g^{\gamma}, h) = e(g, h)^{\gamma}$ to $e(g, \Upsilon = h^{\gamma})$ $= e(g, h)^{\gamma}$. This example is a model technique we used to optimize block-wise P-signature. It is clear that such the modification will not change the fundamental idea of defined security assumption, but simply adjust the argument form one group to the other group. Therefore, $q - XHSDH$ problem still works in our modification. Similarly, $FlexXDH$ problem and $n - XFDHE$ also show the same case as $q - XHSDH$, which adjust the exponent argument from one group to the other group. With this observation we can conclude that $q - XHSDH$ problem, $FlexXDH$ problem and $n - XFDHE$ problem are still suitable for the optimized block-wise P-signatures.

Theorem 5.1. *The optimized asymmetric block-wise P-signatures are secure on the hardness of q-XHSDH problem, FlexXDH problem and n-XFDHE problem.*

Proof sketch. The optimized block-wise P-signature can be proven secure against all three types of forgery attacks. The prove can be slightly modified from the original proofs provided by [4] and hence we will omit the proofs here. In this way, the reduction proof of right-hand side of Figure 5.4 can be constructed.

5.2 Efficiency of optimized block-wise P-signatures

In this section we evaluate how much benefit the optimization can gain in block-wise P-signatures. Let us examine the theoretical analysis between asymmetric pairing based block-wise P-signatures without optimization and one with optimization. Please refer to Figure 5.5, The first observation is that our optimization do not decrease used group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ even though Our modification on the proposed block-wise P-signatures is huge. Secondly in the witness generation of case $\mathcal{R} = \mathcal{R}^{IP}$ and case $\mathcal{R} = \mathcal{R}^{\overline{IP}}$ which take the most number of group elements among all cases, we transfer the cost from group $\mathbb{G}_1$ to group $\mathbb{G}_2$, which can hence get benefit from the small field computation. Overall the result is acceptable because our optimization focuses on alignment of pairing arguments and does not increase or decrease group elements. Therefore, from the view of theoretical analysis the performance enhancement is relatively small while comparing the improvement from porting block-wise P-signature from symmetric pairings to asymmetric pairings.

5.2.1 Pairing pre-computation in MIRACL library

Fortunately MIRACL library has supported a variety of optimization techniques for pairings computation, which are described as follows.

1. Powering pre-computation of bilinear pairings. This function supports pre-computation of fixed arguments $X \in \mathbb{G}_1$ and $Y \in \mathbb{G}_2$ in pairing computation $e(X, Y)$, so that $e(X, Y)^z$ can be sped up for some $z \in \mathbb{Z}_p^*$. One can use the function to pre-compute and store the value

$$\text{precomp\_for\_power}(X,Y) \tag{5.17}$$

Figure 5.5: Comparison of used group elements before/after optimization – F-signature part

Function	BWPS before optimization (# in $\mathbb{G}_1$)	(# in $\mathbb{G}_2$)	BWPS after optimization (# in $\mathbb{G}_1$)	(# in $\mathbb{G}_2$)
$F - signatures$				
$KeyGen$	1	1	1	1
$Sign$	n+2	2	n+2	2
$Verify$	n+8	6	n+7	7
$WitGen - EQ$	n	0	0	n
$WitGen - \overline{EQ}$	n+2	1	1	n+2
$WitGen - IP$	2n	0	0	2n
$WitGen - \overline{IP}$	2n+2	1	1	2n+2
$WitVerify - EQ$	3	3	3	3
$WitVerify - \overline{EQ}$	7	6	7	6
$WitVerify - IP$	2	n+1	n+1	2
$WitVerify - \overline{IP}$	6	n+4	n+4	6

2. Multiplication pre-computation of group elements. This function supports multiplication pre-computation of element $X \in \mathbb{G}_1$ or elements $Y \in \mathbb{G}_2$ in element multiplication, so that X^z or Y^z can be sped up for some $z \in \mathbb{Z}_p^*$. One can use the function to pre-compute and store the value

$$\begin{matrix} \text{precomp\_for\_mult}(X) \\ \text{precomp\_for\_mult}(Y) \end{matrix} \tag{5.18}$$

3. Pairing pre-computation of group elements. This function supports pre-computation of fixed arguments $X \in \mathbb{G}_1$ in pairing computation $e(X, \cdot)$, so that $e(X, Y)$ can be sped up for some $Y \in \mathbb{G}_2$. One can use function to pre-compute and store the value

$$\text{precomp\_for\_pairing}(X) \tag{5.19}$$

4. Fast multiple pairing products technique. This function speeds up multiple pairing products by sharing the same Miller variable and final exponentiation. One can use the following function to compute result

$$\text{multi\_pairing}(CollectionX, CollectionY) \tag{5.20}$$

In our implementation, the first pre-computation technique provided MIRACL library can less benefit on our optimized scheme since block-wise P-signatures simply contain one powering computation of bilinear pairings. Also there is no computation of multiple pairing product in block-wise P-signatures. Therefore, we mainly adopt the second and the third pre-computation provided by MIRACL library to speed up block-wise P-signatures.

The detailed experiment results are collected in Appendix A.2 and Appendix A.3, and the standard deviation of our experiments show negligibly small for the performance of block-wise P-signatures of naive Type-3 pairing porting and performance of optimized block-wise P-signatures to be evidently distinguished.

Figure 5.6: Efficiency comparison of block-wise P-signatures without/with optimization (using pairing curve BN-128)

ID	Function	BWPS in Type-3 without optimization(/sec.)	BWPS in Type-3 with optimization (/sec.)	Improvement (%)
	$F - signatures$			
1	$Sign$	0.0492	0.0302	38.61
2	$Verify$	0.7039	0.6554	6.89
3	$WitGen - EQ$	0.009	0.0041	54.44
4	$WitGen - \overline{EQ}$	0.0492	0.0114	19.14
5	$WitGen - IP$	0.0101	0.0048	52.47
6	$WitGen - \overline{IP}$	0.0174	0.0131	24.71
7	$WitVerify - EQ$	0.2331	0.212	9.05
8	$WitVerify - \overline{EQ}$	0.6985	0.6086	12.87
9	$WitVerify - IP$	0.1564	0.1501	4.02
10	$WitVerify - \overline{IP}$	0.6931	0.6191	10.67
	$SigProve_1$			
11	$SigProve_1 - EQ$	4.6203	4.4409	3.88
12	$SigProve_1 - \overline{EQ}$	6.0987	5.882	3.55
13	$SigVerify_1 - EQ$	13.4915	13.2608	1.70
14	$SigVerify_1 - \overline{EQ}$	18.4456	17.5681	4.75
	$SigProve_2$			
15	$SigProve_2 - EQ$	4.1008	3.9838	2.85
16	$SigProve_2 - \overline{EQ}$	6.1149	5.8713	3.98
17	$SigProve_2 - IP$	3.5716	3.4304	3.95
18	$SigProve_2 - \overline{IP}$	6.4235	6.1146	4.80
19	$SigVerify_2 - EQ$	12.1482	11.8556	2.40
20	$SigVerify_2 - \overline{EQ}$	18.4185	17.5311	4.81
21	$SigVerify_2 - IP$	10.7983	10.4557	3.17
22	$SigVerify_2 - \overline{IP}$	18.8304	17.994	4.44

5.2.2 Experiment result.

Please refer to Figure 5.6, the data are collected from 20 runs of experiment with single block of message. This figure illustrates how the optimization affect the efficiency of block-

wise P-signatures, which can be roughly divided into two part: F-unforgeable signatures and signature proof generation and verification.

Firstly let us examine the performance of optimization in F-unforgeable signatures. Overall it is clear that efficiency enhancement depends on how large the proportion of function codes can be precomputed. For the signature generation, it is evident that three of seven equations in the algorithm can be sped up by pre-computing multiplication of group elements g and h, which is expected to receive 42.85%'s improvement in efficiency. Our experiment shows 38.61%'s enhancement, which is consistent to the expected value. In other words, the result implicitly shows the performance of pre-computing elements multiplication can show a positive relation with the proportion of codes that can be pre-computed. In terms of signature verification, our revised scheme possess five of nine pairing computation can be sped up by the pre-computation of $e(g, \cdot)$, which approximately 41.66 %'s improvement in efficiency. However, our experiment can only get about 7%'s improvement, which illustrates that the fixed argument pre-computation cannot provide consistent percentage while comparing with expected improvement.

Next let us consider the performance of witness generation and verification. For signature generation of cases $\mathcal{R}$=$\mathcal{R}^{EQ}$ and $\mathcal{R}$=$\mathcal{R}^{IP}$, the results show half improvement in efficiency. Essentially the enormous improvement also depends on the the proportion of precomputed codes; hence in our experiment with only one message block, the improvement is expected to receive half improvement in running time. The other two cases, $\mathcal{R}$=$\mathcal{R}^{\overline{EQ}}$ and $\mathcal{R}$=$\mathcal{R}^{\overline{IP}}$, also presents about 20%'s enhancement in efficiency. However, the receiving efficiency of these two cases will decrease with the increase of message blocks. In terms of witness verification, the performance of optimization is low. The experiment result shows approximately 10%'s improvement in the cases $\mathcal{R}$=$\mathcal{R}^{EQ}$, $\mathcal{R}$=$\mathcal{R}^{\overline{EQ}}$ and $\mathcal{R}$=$\mathcal{R}^{\overline{IP}}$, however, there is only 4% in the case of $\mathcal{R}$=$\mathcal{R}^{IP}$. This phenomenon again supports the statement that pairing pre-compution of fixed argument can provides lower gain in efficiency enhancement.

Lastly in terms of signature proof generation and verification, none of all cases in our experiment can get efficiency benefit over 5%, which cannot regarded as a desirable performance. The situation may result from the complexity of Groth-Sahai proofs. Firstly, Groth-Sahai commitments essentially is an process of encryption, which means they needs to use fresh random value to conceal the element to be committed and leaves less space for pairing pre-computation. Secondly, the verification of Groth-Sahai proofs require hundreds of pairing computation and the group elements involved in pairing computation are usually unshown during the verification procedure. These two unfavorable factors result in less efficiency gain in signature proof generation and verification of block-wise P-signatures.

In summary, our optimization in block-wise P-signatures receive significant performance in F-signatures part, and signature proof generation and verification can get less efficiency enhancement in our optimization.

5.3 Other practical efficiency analysis of block P-signatures

In last section we focus on evaluating the performance of applying optimization techniques in our revised block-wise P-signatures. All the experiment data are collected on BN curves; however, it is desirable to know the performance of block-wise P-signature on different pairing curves. In this section, we discuss and compare how the different pairing curves affect the performance of block-wise P-signatures.

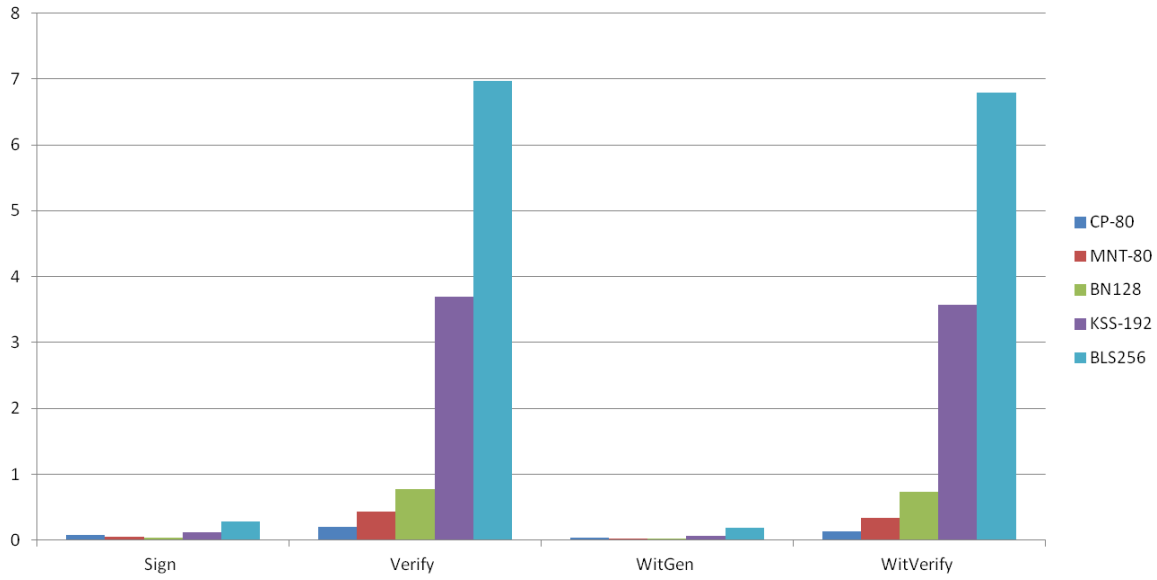
5.3.1 Efficiency analysis on different pairing curves

The choice of pairing curve is also regarded as an important issue for pairing computation. Due to the unique design of different pairing curves, the performance of pairing computation can present unique result. In this section we try to examine how the different pairing curves affect the performance of block-wise P-signatures. Here the experiment is simplified and mainly focus on measuring the performance of case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$ of F-unforgeable signatures among different pairing curves. In terms of pairing curves, we adopt CP-80, MNT-80, BN-128, KSS-192, BLS-256, which left-hand side of the notation is the curve name, and right-hand side represents the security degree. The code of F-unforgeable signatures we adopted is optimized and contains pairing pre-computation. Moreover, we increase the number of repeating experiment to 100 and extend the message to 20 blocks, which can further increase the precision of this experiment. The detailed experiment results are collected in Appendix A.4, and the standard deviation of our experiments show negligibly small for the performance of block-wise P-signatures on different pairing curves to be evidently distinguished.

From a general view, please refer to Figure 5.7, it is interesting that running time of signature verification and witness verification is increasing with the raise of security degree; however, performance of signature generation and witness generation is close among different curves with different security degree. As we have understood, signature verification and witness verification comprise pairing computation that can only get a little benefit from pairing pre-computation; therefore, it is not surprising that the pairing curves with higher security degree will take more time on pairing computation. Moreover, there are two remarkable points showing in this comparison. Please refer to Figure 5.8, firstly it is noticeable that CP-80 take less time doing verification than MNT-80, which shows CP-80 take more advantage than MNT-80 on pairing computation of same security degree. Secondly there is a small increase from CP-80 to BN-128; however, the running time dramatically increase from BN-128 to BLS-256, which illustrate that BN-128 shows significant performance in pairing computation.

In terms of signature generation and witness generation, the experiment result is impressing. Firstly no matter what pairing curves and security degree we adopt, all these five pairing curves present excellent performance. In F-unforgeable signatures, signature generation and witness generation comprise a series of multiplication of group elements on elliptic curves; hence the experiment result shows that all five pairing curves show remarkable performance on multiplication on curve points. In spite of high performance of all

Figure 5.7: Efficiency comparison among different pairing curves - F-signatures part



pairing curves, there are still two noticeable points. Please refer to Figure 5.8 which give the detailed data of this experiment, the first point is that MNT-80 perform better than CP-80 in signature generation and witness generation, which is different from the result of signature verification and witness verification. This phenomenon illustrates that MNT curve is better at doing point multiplication than CP curve. The second point is that BN-128 presents the best performance in signature generation and witness generation, even better than CP-80 and MNT-80 with lower security degree. This result implicitly suggest that BN curves possesses tremendous efficiency in doing point multiplication.

In short our experiment provide a reference for application developers to choose which curves is better for their applications. Considering the security requirement, most cryptographic products require at least AES-128; hence BN-128 can be an appropriate choice. If applications focus on computational performance and only need lower security degree, developers can evaluate their applications' requirement and choose CP-80 or MNT-80. In practical situation, signature verification can be operated many times and there is only one signature generation; hence in this situation CP-80 can provide better performance than MNT-80.

5.3.2 Further efficiency analysis on BN curves

In this section we would like to analyze the performance of BN curves of different security degree. From the experiment result of BN-128 we have understood that BN curves show

Figure 5.8: Detailed data of efficiency comparison among different pairing curves - F-signatures part

Function	CP-80	MNT-80	BN-128	KSS-192	BLS-256
$Sign$	0.077	0.047	0.04	0.118	0.2774
$Verify$	0.2003	0.4333	0.7785	3.6986	6.9683
$WitGen - \overline{EQ}$	0.0351	0.0289	0.0189	0.0668	0.1932
$WitVerify - \overline{EQ}$	0.132	0.3357	0.7316	3.5669	6.7946

well balance on security degree and performance in efficiency. Essentially MIRACL library allows BN curves being built for the security degree of AES-128 and AES-192, which inspires us to measure the performance of BN curves of these two security degrees. To facilitate the comparison between these two curve settings, we set the same experiment parameters as the environment we set for block-wise P-signatures: 20 runs of one message block. Please see the experiment result in Figure 5.9.

Firstly let us consider the result of F-signatures part of block-wise P-signatures in different security degree. Please refer to Figure 5.10, the result indicates that in the signature generation and witness generation, BN-192 setting present competitive efficiency, which means these operations can be computed in a fairly short time. However, in terms of signature verification and witness verification, the running time of BN-192 setting dramatically increase. It is remarkable that in the case $\mathcal{R}$=$\mathcal{R}^{EQ}$ and $\mathcal{R}$=$\mathcal{R}^{IP}$ of $SigVerify_2$, the verification procedures take roughly one second, which is acceptable for the majority of applications; but, the others operations arise to nearly 3.5 seconds, which is a significant difference while comparing to BN-128-setting. In short, for the F-signatures part of block-wise P-signatures, BN-192 setting present an acceptable result.

Secondly we examine the result of signature proof generation and verification of block-wise P-signatures. Please refer to Figure 5.11, it is clear that signature proof generation and verification take terribly long time to complete the computation. Roughly the data can be divided into two parts, signature proof generation (Function ID 11, 12, 15, 16, 17, 18) and signature proof verification (Function ID 13, 14, 19, 20, 21, 22). The proof generation part can be done in 40 seconds while the proof verification part take up to 90 seconds. Comparing to BN-128 setting, BN-192 setting cannot be regards to a nice choice for constructing block-wise P-signatures.

Finally let us compare the efficiency improvement of pre-computation between BN-128 setting and BN-196 setting. Please refer to Figure 5.12, it is surprising that the efficiency improvement present consistently decline in BN-192 setting while comparing to BN-128 setting. This phenomenon illustrates that the pre-computation for both element multiplication and pairing computation in security degree of AES-128 can take more advantage than in AES-192. It is reasonable to suppose that the cause comes from the unique property of BN curve, and may not happen on other pairing curves.

Figure 5.9: Efficiency comparison of block-wise P-signature without/with optimization (using pairing curve BN-192)

ID	Function	BWP in Type-3 without optimization(/sec.)	BWP in Type-3 with optimization (/sec.)	Improvement (%)
	$F - signatures$			
1	$Sign$	0.3253	0.2164	33.47
2	$Verify$	3.6861	3.4822	5.53
3	$WitGen - EQ$	0.051	0.028	45.09
4	$WitGen - \overline{EQ}$	0.0868	0.0735	15.32
5	$WitGen - IP$	0.0484	0.0239	50.61
6	$WitGen - \overline{IP}$	0.1065	0.0811	23.84
7	$WitVerify - EQ$	1.2265	1.1321	7.69
8	$WitVerify - \overline{EQ}$	3.5844	3.2391	9.63
9	$WitVerify - IP$	0.7813	0.7667	1.86
10	$WitVerify - \overline{IP}$	3.5371	3.2696	7.56
	$SigProve_1$			
11	$SigProve_1 - EQ$	26.6617	25.9066	2.83
12	$SigProve_1 - \overline{EQ}$	34.3566	33.4852	2.53
13	$SigVerify_1 - EQ$	68.8043	67.2217	2.30
14	$SigVerify_1 - \overline{EQ}$	90.8369	88.7359	2.31
	$SigProve_2$			
15	$SigProve_2 - EQ$	23.8078	23.3271	2.01
16	$SigProve_2 - \overline{EQ}$	34.4066	33.6923	2.07
17	$SigProve_2 - IP$	19.5668	18.8773	3.52
18	$SigProve_2 - \overline{IP}$	35.1501	34.0464	3.13
19	$SigVerify_2 - EQ$	61.4989	60.8401	1.07
20	$SigVerify_2 - \overline{EQ}$	91.4734	89.4296	2.23
21	$SigVerify_2 - IP$	52.7504	51.6206	2.14
22	$SigVerify_2 - \overline{IP}$	94.3742	91.279	3.27

Figure 5.10: Performance comparison of block-wise P-signatures on pairing curves BN-128 and BN-196 - F-signatures part (please refer x-axis to the ID shows in Figure 5.9)

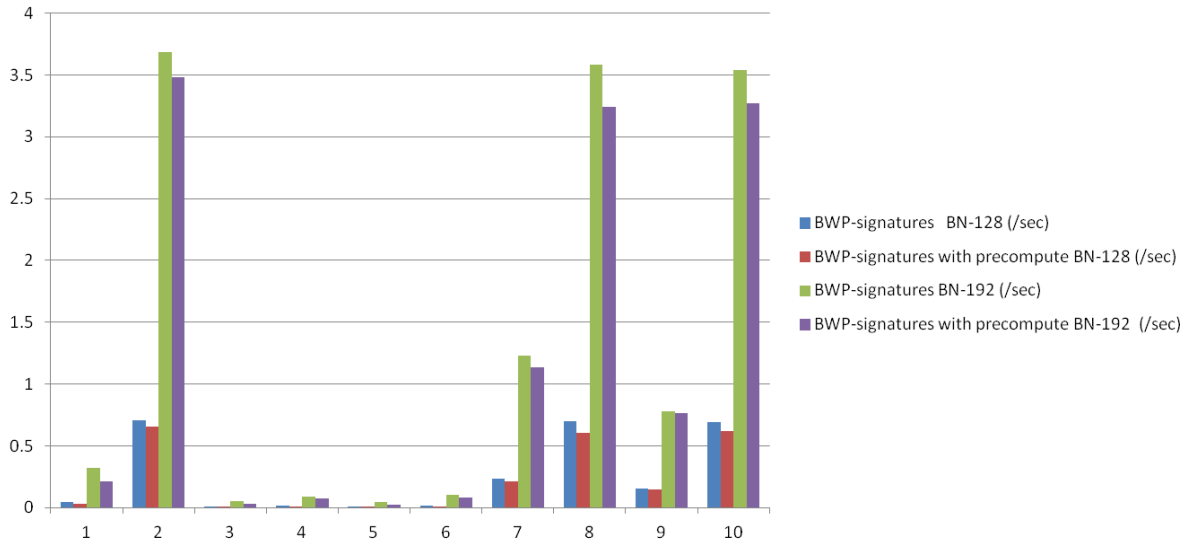


Figure 5.11: Performance comparison of block-wise P-signatures on pairing curves BN-128 and BN-196 - whole part (please refer x-axis to the ID shows in Figure 5.9)

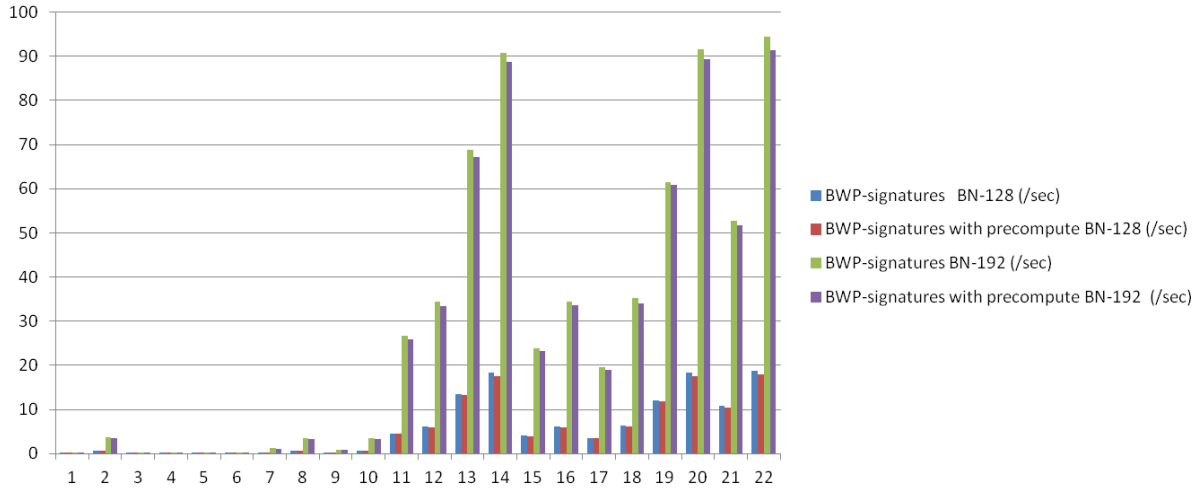
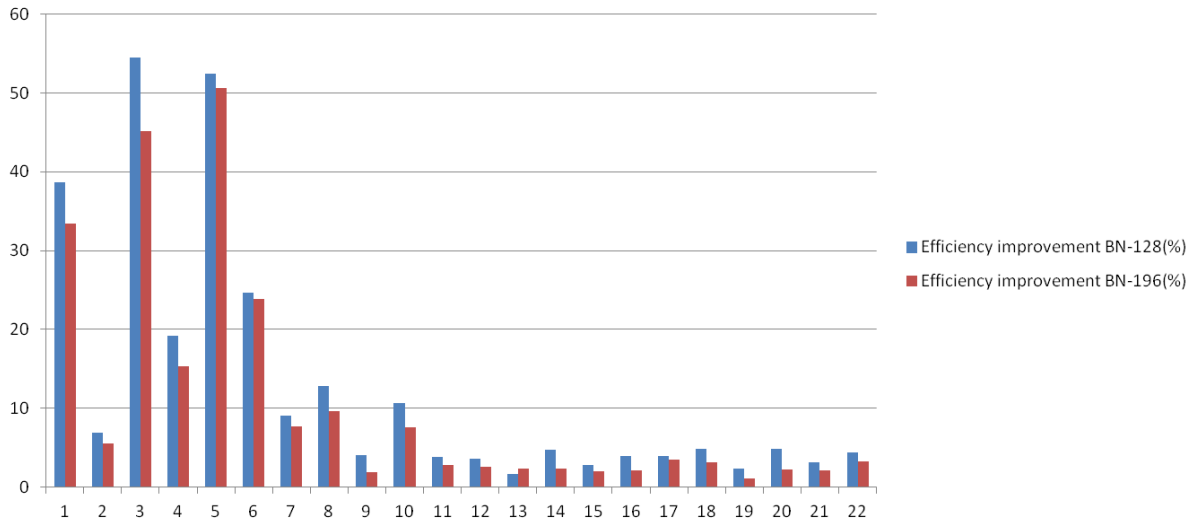


Figure 5.12: Comparison of efficiency enhancement between block-wise P-signatures in pairing curves BN-128 and BN-196 (please refer x-axis to the ID shows in Figure 5.9)



Chapter 6

Conclusions

In this dissertation we make an comprehensive analysis on block-wise P-signatures from a variety of efficiency perspectives. For the theoretical efficiency analysis of block-wise P-signatures, we proposed the first block-wise P-signature scheme based on asymmetric pairings, which is theoretically regarded to possess efficiency advantage over ones based on symmetric pairings. Moreover, our proposed block-wise P-signatures can be proven secure on the assumptions of q-XHSDH problem, Flex-XDH problem and n-XFDHE problem, which are firstly proposed in our works. Comparing with ordinary block-wise [4] introduced by Izabachène et al, our proposed scheme can provide competitive efficiency in group elements used in signature generation and verification; furthermore, our proposed scheme can significantly reduce used group elements in signature proofs generation and verification.

In terms of practical implementation, to verify the correctness of our proposed scheme and evaluate the performance difference between our designs and Izabachène et al.'s works. We both firstly implement block-wise P-signatures based on symmetric pairings [1]. and ones based on asymmetric pairings [2]. Our proposed block-wise P-signatures can get 79%-85%'s efficiency improvement on signature generation and witness generation, as well as 41%-45%'s efficiency enhancement on signature verification and witness verification. Moreover, the signature proof generation and verification based on SXDH construction can get get nearly 93%-97%'s improvement over ones based on DLIN construction.

In spite of enormous efficiency improvement of our proposed block-wise p-signatures, we further seek for higher efficiency enhancement by applying optimization techniques in our proposed schemes. Comparing with our proposed schemes, the optimized block-wise P-signatures can 19%-52%'s improvement on signature generation and witness generation while signature verification and witness verification can hence get 4%-12%' efficiency enhancement. Moreover, signature proof generation and verification can get benefit of 1%-4%'s efficiency improvement.

[1] Type-1 pairings in specific is adopted and signature proof generation and verification adopt DLIN-based Groth-Sahai proof systems.

[2] Type-3 pairings in specific is adopted and signature proof generation and verification adopt SXDH-based Groth-Sahai proof systems.

Finally, we evaluate the performance of block-wise P-signatures on different pairing curves and different security degree. In short, Barreto-Naehrig (BN) curve presents well balance both in fast pairing computation and acceptable security degree for the implementation of block-wise P-signatures.

In conclusion, our proposed block-wise P-signatures can bring significant efficiency benefit to the research of P-signatures; moreover, the efficiency analysis on block-wise P-signatures can also be the reference for the implementation of pairing-based cryptographic applications.

6.1 Future work of block-wise P-signatures

Block-wise P-signatures are relatively complex digital signatures since they combine a variety of cryptographic primitives, including digital signatures, attribute-based cryptographic primitives, predicate cryptographic primitives and non-interactive zero-knowledge proofs. Although our proposed block-wise P-signatures with optimization have received significant progress on improving efficiency of them, there are two possible research directions for the future works.

Efficiency improvement of signature proofs verification

Figure 6.1: Number of pairings computation per proof verification, where n and m stand for the number of different types of variables. [5]

proof equation	Naive verification	Batch verification
SXDH		
Pairing product equation	$5m + 3n + 16$	$m + 2n + 8$
Multi-scalar multiplication equation in $\mathbb{G}_1$	$8m + 2n + 14$	$\min(2n + 9,\ 2m + n + 7)$
Multi-scalar multiplication equation in $\mathbb{G}_2$	$8n + 2m + 14$	$\min(2m + 9,\ 2n + m + 7)$
Quadratic equation	$8m + 8n + 12$	$2\ \min(m,n)+8$
SXDH		
Pairing product equation	$12n + 27$	$3n + 6$
Multi-scalar multiplication equation in $\mathbb{G}$	$9n + 12m + 27$	$3n + 3m + 6$
Quadratic equation	$18n + 24$	$3n + 6$

Observing the efficiency of our proposed block-wise P-signatures, it is evident that signature proofs generation and verification still take long time to complete the procedures, especially on the signature proofs verification.[3] To improve the performance of signature proofs generation and verification, it is clear that we should fundamentally improve the

[3]Please refer to Figure 4.4 and Figure 4.5

performance of Groth-Sahai proof systems. Recently Blazy et al. [5] introduced the technique that doing batch verification of Groth-Sahai proofs, which can enormously decrease the number of pairing computation. Please see the performance enhancement of batch verification in Figure 6.1. The techniques batch verification adopt are three: Moving the exponent into pairing computation, Moving the product into pairing computation, and Switching the product. In other words, the batch verification should rearrange the equations of signature proof verification of block-wise P-signatures by using group element multiplication, so that the batch verification dramatically decreases necessary pairing computation. In fact, Izabachène et al. [4] also suggested to use this batch techniques on their block-wise P-signatures and they expected to use less than 20 pairing computation to finish one signature proof verification procedure. Comparing with naive verification, the technique of batch verification indeed provides a great solution for block-wise P-signatures.

Application of block-wise P-signatures

Block-wise P-signatures are proposed for achieving anonymous credentials systems, which can be widely used in a variety of application domains: from large system applications in cloud computing, to small devices such as mobile phones and Oyster cards. From our experiment result, up to now it is evident that computing block-wise P-signatures still require machines with powerful computation ability. Motivated by this observation, there are two directions for the development of block-wise P-signatures. Firstly, considering the required computing power of block-wise P-signatures, one can focus on constructing proxy architectures to alleviate the computation burden for portable devices such as mobile phones. Another possible research direction can focus on the reconstruction of light-weight block-wise p-signatures, so that the new construction can be used in smart cards or other devices without powerful computing ability.

Bibliography

[1] Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics **156**(16) (2008) pp. 3113–3121

[2] Scott, M.: On the efficient implementation of pairing-based protocols. In: Cryptography and Coding - 13th IMA International Conference - IMACC 2011, Lecture Notes in Computer Science vol. 7089, Springer-Verlag (2011) pp.296–308

[3] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Advances in Cryptology - EUROCRYPT 2008, Lecture Notes in Computer Science vol. 4965, Springer-Verlag (2008) pp.415–432

[4] Izabachène, M., Libert, B., Vergnaud, D.: Block-wise p-signatures and non-interactive anonymous credentials with efficient attributes. In: Cryptography and Coding - 13th IMA International Conference - IMACC 2011, Lecture Notes in Computer Science vol. 7089, Springer-Verlag (2011) pp.431–450

[5] Blazy, O., Fuchsbauer, G., Izabachène, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch groth-sahai. In: Applied Cryptography and Network Security - 8th International Conference, ACNS 2010, Lecture Notes in Computer Science vol. 6123, Springer-Verlag (2010) pp.218–235

[6] Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Commun. ACM **28**(10) (1985) pp. 1030–1044

[7] Damgård, I.: Payment systems and credential mechanisms with provable security against abuse by individuals. In: Advances in Cryptology - CRYPTO 1988, Lecture Notes in Computer Science vol. 403, Springer-Verlag (1988) pp.328–335

[8] Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Selected Areas in Cryptography - SAC 1999, Lecture Notes in Computer Science vol. 1758, Springer-Verlag (1999) pp.184–199

[9] Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Advances in Cryptology - EUROCRYPT 2001, Lecture Notes in Computer Science vol. 2045, Springer-Verlag (2001) pp.93–118

[10] Camenisch, J., Herreweghen, E.V.: Design and implementation of the *idemix* anonymous credential system. In: ACM Conference on Computer and Communications Security - CCS2002, ACM (2002) pp.21–30

[11] Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Security in Communication Networks, Third International Conference - SCN 2002, Lecture Notes in Computer Science vol. 2576, Springer-Verlag (2002) pp.268–289

[12] Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Advances in Cryptology - CRYPTO 2004, Lecture Notes in Computer Science vol. 3152, Springer-Verlag (2004) pp.56–72

[13] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Theory of Cryptography, Fifth Theory of Cryptography Conference - TCC 2008, Lecture Notes in Computer Science vol. 4948, Springer-Verlag (2008) pp.356–374

[14] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: Compact e-cash and simulatable vrfs revisited. In: Pairing-Based Cryptography - Pairing 2009, Lecture Notes in Computer Science vol. 5671, Springer-Verlag (2009) pp.114–131

[15] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Advances in Cryptology - CRYPTO 2009, Lecture Notes in Computer Science vol. 5677, Springer-Verlag (2009) pp.108–125

[16] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: 27th Annual Symposium on Foundations of Computer Scienc - FOCS 86, IEEE Computer Society (1986) pp.174–187

[17] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology - CRYPTO 1986, Lecture Notes in Computer Science vol. 263, Springer-Verlag (1986) pp.186–194

[18] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, Proceedings of the 1st ACM Conference on Computer and Communications Security (1993) pp.62–73

[19] Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th Symposium on Foundations of Computer Science - FOCS 2003, IEEE Computer Society (2003) pp.102–113

[20] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited, J. ACM, vol. 51 (2004) pp.557–594

[21] Camenisch, J., Groß, T.: Efficient attributes for anonymous credentials (extended version). IACR Cryptology ePrint Archive **http://eprint.iacr.org/2010/496** (2010) pp.496

[22] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Advances in Cryptology - CRYPTO 2004, Lecture Notes in Computer Science vol. 3152, Springer-Verlag (2004) pp.41–55

[23] Scott, M.: Authenticated id-based key exchange and remote log-in with simple token and pin number. **http://eprint.iacr.org/2002/164.pdf** (2002)

[24] Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Public Key Cryptography - PKC 2007, Lecture Notes in Computer Science vol. 4450, Springer-Verlag (2007) pp.1–15

[25] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Advances in Cryptology - CRYPTO 2005, Lecture Notes in Computer Science vol. 3621, Springer-Verlag (2005) pp.258–275

[26] NIST: Nist recommendation for key management part 1: General, nist special publication. **http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf** (2005)

[27] Lenstra, A.K.: Handbook of Information Security. Volume 2. Wiley (2005)

[28] ECRYPT: Ecrypt yearly report on algorithms and keysizes. **http://www.ecrypt.eu.org/documents/D.SPA.10-1.1.pdf** (2005)

[29] Scott, M.: Computing the tate pairing. In: The Cryptographers' Track at the RSA Conference - CT-RSA 2005, Lecture Notes in Computer Science vol. 3376, Springer-Verlag (2005) pp.293–304

[30] Costello, C., Stebila, D.: Fixed argument pairings. In: Progress in Cryptology - LATINCRYPT 2010, Lecture Notes in Computer Science vol. 6212, Springer-Verlag (2010) pp.92–108

[31] Granger, R., Smart, N.P.: On computing products of pairings. IACR Cryptology ePrint Archive **http://eprint.iacr.org/2006/172** (2006) pp.172

[32] Ghadafi, E., Smart, N.P., Warinschi, B.: Groth-sahai proofs revisited. In: Public Key Cryptography, Lecture Notes in Computer Science vol. 6056, Springer-Verlag (2010) pp.177–192

[33] Boneh, D., Boyen, X.: Short signatures without random oracles. In: Advances in Cryptology - EUROCRYPT 2004, Lecture Notes in Computer Science vol. 3027, Springer-Verlag (2004) pp.56–73

[34] Certivox: Miracl crypto sdk. **http://certivox.com/index.php/solutions/miracl-crypto-sdk/**

[35] Blake, I.F., Seroussi, G., Smart, N.P.: Advances in Elliptic Curve Cryptography. Volume 2. Cambridge University Express (2005)

[36] Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for fr-reduction. In: IEICE transactions on Fundamentals. Volume E84-A(5). (2001) pp.1234–1243

[37] Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Selected Areas in Cryptography - SAC 2005, Lecture Notes in Computer Science vol. 3897, Springer-Verlag (2005) pp.319–331

[38] Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In: Pairing-Based Cryptography - Pairing 2008, Lecture Notes in Computer Science vol. 5209, Springer-Verlag (2008) pp.126–135

[39] Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Security in Communication Networks, Third International Conference - SCN 2002, Lecture Notes in Computer Science vol. 2576, Springer-Verlag (2002) pp.257–267

Appendix A

Experiment Results

A.1 Experiment result of Type-1 pairing based P-signatures

Figure A.1: Type-1 pairings - case $\mathcal{R} = \mathcal{R}^{EQ}$

Statistics

		Setup	Sign	Verify	WitnessGen_EQ	WitnessVerify_EQ
N	Valid	20	20	20	20	20
	Missing	0	0	0	0	0
Mean		.92158515	.30135285	1.28288775	.04400175	.42733400
Std. Error of Mean		.013368242	.008020846	.018755627	.001224593	.010681330
Std. Deviation		.059784594	.035870316	.083877713	.005476548	.047768360
Minimum		.853013	.274918	1.184911	.038776	.378189
Maximum		1.074047	.421742	1.485180	.061696	.552517

Figure A.2: Type-1 pairings - case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$

Statistics

		Setup	Sign	Verify	WitnessGen_NEQ	WitnessVerify_NEQ
N	Valid	20	20	20	20	20
	Missing	0	0	0	0	0
Mean		.90151540	.29194295	1.26527760	.09428765	1.23067780
Std. Error of Mean		.011955331	.004377100	.014664369	.000656535	.015734753
Std. Deviation		.053465867	.019574987	.065581053	.002936112	.070367956
Minimum		.838311	.272262	1.182660	.089257	1.152632
Maximum		1.068521	.365864	1.454494	.101444	1.461125

Figure A.3: Type-1 pairings - case $\mathcal{R} = \mathcal{R}^{IP}$

Statistics

		Setup	Sign	Verify	WitnessGen_IP	WitnessVerify_IP
N	Valid	20	20	20	20	20
	Missing	0	0	0	0	0
Mean		.89177740	.32581135	1.29256415	.04304805	.26705675
Std. Error of Mean		.008490105	.005218063	.013832456	.000539083	.003418591
Std. Deviation		.037968903	.023335889	.061860623	.002410853	.015288405
Minimum		.837268	.301010	1.216610	.039268	.248397
Maximum		.993322	.398153	1.494926	.049604	.310074

Figure A.4: Type-1 pairings - case $\mathcal{R} = \mathcal{R}^{\overline{IP}}$

Statistics

		Setup	Sign	Verify	WitnessGen_NIP	WitnessVerify_NIP
N	Valid	20	20	20	20	20
	Missing	0	0	0	0	0
Mean		.88747815	.29465815	1.25835560	.10020485	1.19012015
Std. Error of Mean		.016556059	.006281588	.021226990	.000940950	.013005810
Std. Deviation		.074040946	.028092115	.094929984	.004208056	.058163749
Minimum		.830375	.274548	1.175634	.095741	1.128129
Maximum		1.182779	.406483	1.562926	.109185	1.314017

A.2 Experiment result of Type-3 pairing based P-signatures

Figure A.5: Type-3 pairings - case $\mathcal{R} = \mathcal{R}^{EQ}$

Statistics

		Setup	Sign	Verify	WitnessGen_EQ	WitnessVerify_EQ	CSR_Setup	SigProve1_EQ	SigVerify1_EQ	SigProve2_EQ	SigVerify_EQ
N	Valid	20	20	20	20	20	20	20	20	20	20
	Missing	0	0	0	0	0	0	0	0	0	0
Mean		.05027265	.04743380	.67413230	.00909035	.23314415	.04405120	4.62035425	13.49154145	4.10086950	12.14829320
Std. Error of Mean		.000578046	.000532926	.005593251	.000130337	.002033351	.000827417	.069125120	.070725968	.021582124	.081859046
Std. Deviation		.002585098	.002383316	.025013777	.000582885	.009093421	.003700323	.309136932	.316296144	.096518193	.366084783
Minimum		.046658	.044321	.650778	.008211	.225571	.040376	4.391355	13.101480	4.000276	11.786126
Maximum		.056185	.053313	.735688	.010429	.260429	.054758	5.629051	14.149736	4.390399	13.235085

Figure A.6: Type-3 pairings - case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$

Statistics

		Setup	Sign	Verify	WitnessGen_NEQ	WitnessVerify_NEQ	CRS_Setup	SigProve1_NEQ	SigVerify1_NEQ	SigProve2_NEQ	SigVerify2_NEQ
N	Valid	20	20	20	20	20	20	20	20	20	20
	Missing	0	0	0	0	0	0	0	0	0	0
Mean		.05087980	.04762895	.68472235	.01416910	.69851735	.04415715	6.09871725	18.44566995	6.11498430	18.41857030
Std. Error of Mean		.000506227	.000378164	.003861331	.000239914	.005921944	.000586273	.027385498	.071802265	.036260725	.070637385
Std. Deviation		.002263916	.001691200	.017268399	.001072929	.026483738	.002621891	.122471669	.321109491	.162162890	.315899988
Minimum		.047876	.045320	.662056	.013371	.662329	.041465	5.917422	17.954811	5.865433	17.887451
Maximum		.057025	.051035	.724997	.017432	.763590	.053603	6.346308	18.992123	6.546158	18.960619

Figure A.7: Type-3 pairings - case $\mathcal{R} = \mathcal{R}^{IP}$

Statistics

		Setup	Sign	Verify	WitnessGen_IP	WitnessVerify_IP	CRS_Setup	SigProve2_IP	SigVerify2_IP
N	Valid	20	20	20	20	20	20	20	20
	Missing	0	0	0	0	0	0	0	0
Mean		.05300030	.05057225	.70636760	.01015900	.15640745	.04554395	3.57160655	10.79838575
Std. Error of Mean		.000894603	.000991682	.004738942	.000393582	.003071550	.000662869	.016419793	.041706643
Std. Deviation		.004000788	.004434935	.021193193	.001760152	.013736388	.002964439	.073431547	.186517776
Minimum		.049326	.046158	.666205	.008606	.143812	.041792	3.503885	10.533720
Maximum		.066041	.064579	.769584	.014774	.209497	.054491	3.821653	11.300502

Figure A.8: Type-3 pairings - case $\mathcal{R} = \mathcal{R}^{\overline{IP}}$

Statistics

		Setup	Sign	Verify	WitnessGen_NIP	WitnessVerify_NIP	CRS_Setup	SigProve2_NIP	SigVerify2_NIP
N	Valid	20	20	20	20	20	20	20	20
	Missing	0	0	0	0	0	0	0	0
Mean		.05107920	.04873170	.69710250	.01749110	.69313570	.04570510	6.42353105	18.83043820
Std. Error of Mean		.000427034	.000494034	.005441113	.000419610	.010129301	.000693566	.032285106	.069674685
Std. Deviation		.001909755	.002209385	.024333399	.001876552	.045299610	.003101722	.144383383	.311594664
Minimum		.048004	.045515	.661646	.015840	.650566	.042091	6.186861	18.335826
Maximum		.055115	.053548	.776395	.025074	.856911	.056336	6.739728	19.374175

A.3 Experiment result of Optimized P-signatures

Figure A.9: Optimized Type-3 pairings - case $\mathcal{R} = \mathcal{R}^{EQ}$

Statistics

		Setup	Sign	Verify	WitnessGen_EQ	WitnessVerify_EQ	CRS_Setup	SigProve1_EQ	SigVerify1_EQ	SigProve2_EQ	SigVerify2_EQ
N	Valid	20	20	20	20	20	20	20	20	20	20
	Missing	0	0	0	0	0	0	0	0	0	0
Mean		.19153445	.03023230	.65549075	.00417675	.21204490	.07018285	4.44097725	13.26083370	3.98384055	11.85562630
Std. Error of Mean		.001094000	.000359615	.001925503	.000136053	.001311476	.000737204	.017052025	.049958108	.005570816	.011838947
Std. Deviation		.004892515	.001608249	.008611111	.000608445	.005865099	.003296878	.076258972	.223419450	.024913447	.052945379
Minimum		.183796	.028097	.640766	.003519	.199792	.066527	4.388279	13.118826	3.948245	11.745511
Maximum		.206356	.033864	.675146	.005472	.228818	.079729	4.757067	14.183754	4.036066	11.971007

Figure A.10: Optimized Type-3 pairings - case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$

Statistics

		Setup	Sign	Verify	WitnessGen_NEQ	WitnessVerify_NEQ	CSR_Setup	SigProve1_NEQ	SigVerify1_NEQ	SigProve2_NEQ	SigVerify_NEQ
N	Valid	20	20	20	20	20	20	20	20	20	20
	Missing	0	0	0	0	0	0	0	0	0	0
Mean		.19109270	.02994315	.65370005	.01142425	.60867875	.06845690	5.88200035	17.56810610	5.87139670	17.53119690
Std. Error of Mean		.001548253	.000533641	.002161029	.000143809	.003099185	.000484578	.024187773	.050156458	.020730763	.007462251
Std. Deviation		.006924000	.002386515	.009664418	.000643131	.013859977	.002167098	.108171008	.224306497	.092710789	.033372202
Minimum		.183136	.027983	.640491	.010769	.595726	.065835	5.802990	17.408878	5.776844	17.462205
Maximum		.214259	.038382	.675255	.012948	.655193	.073102	6.311577	18.495545	6.186870	17.579083

Figure A.11: Optimized Type-3 pairings - case $\mathcal{R} = \mathcal{R}^{IP}$

Statistics

		Setup	Sign	Verify	WitnessGen_IP	WitnessVerify_IP	CRS_Setup	SigProve2_IP	SigVerify2_IP
N	Valid	20	20	20	20	20	20	20	20
	Missing	0	0	0	0	0	0	0	0
Mean		.19665275	.03100980	.67792265	.00482510	.15017390	.07129385	3.43048765	10.45579780
Std. Error of Mean		.000656048	.000240844	.004558152	.000139474	.000716935	.000412192	.013755844	.014977999
Std. Deviation		.002933936	.001077087	.020384677	.000623746	.003206232	.001843379	.061518004	.066983646
Minimum		.188319	.030004	.648987	.004272	.141075	.067356	3.344815	10.266152
Maximum		.202171	.033532	.756858	.007037	.157091	.076071	3.654472	10.543908

Figure A.12: Optimized Type-3 pairings - case $\mathcal{R} = \mathcal{R}^{\overline{IP}}$

Statistics

		Setup	Sign	Verify	WitnessGen_NIP	WitnessVerify_NIP	CRS_Setup	SigProve2_NIP	SigVerify2_NIP
N	Valid	20	20	20	20	20	20	20	20
	Missing	0	0	0	0	0	0	0	0
Mean		.19138835	.03098820	.66186590	.01312140	.61914430	.06916595	6.11469865	17.99474615
Std. Error of Mean		.001003260	.000336886	.003317167	.000160652	.003614226	.000566525	.019748311	.034329971
Std. Deviation		.004486717	.001506600	.014834824	.000718460	.016163311	.002533579	.088317131	.153528297
Minimum		.184207	.028680	.638814	.011965	.600985	.065886	6.035866	17.880903
Maximum		.201465	.034076	.696090	.014711	.669767	.074707	6.425287	18.614412

A.4 Experiment result of P-signatures on different curves

Figure A.13: CP-80 - case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$

Statistics

		Setup	Sign_NEQ	Verify_NEQ	WitGen_NEQ	WitVerify_NEQ
N	Valid	100	100	100	100	100
	Missing	0	0	0	0	0
Mean		.94699607	.04703891	.43333754	.02892057	.33577150
Std. Error of Mean		.009129348	.000795823	.006292754	.000760795	.004124346
Std. Deviation		.091293479	.007958231	.062927541	.007607947	.041243456
Minimum		.830621	.038567	.368242	.023475	.284600
Maximum		1.419946	.072260	.632807	.082563	.538995

Figure A.14: MNT-80 - case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$

Statistics

		Setup	Sign_NEQ	Verify_NEQ	WitGen_NEQ	WitVerify_NEQ
N	Valid	100	100	100	100	100
	Missing	0	0	0	0	0
Mean		.94699607	.04703891	.43333754	.02892057	.33577150
Std. Error of Mean		.009129348	.000795823	.006292754	.000760795	.004124346
Std. Deviation		.091293479	.007958231	.062927541	.007607947	.041243456
Minimum		.830621	.038567	.368242	.023475	.284600
Maximum		1.419946	.072260	.632807	.082563	.538995

Figure A.15: BN-128 - case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$

Statistics

		Setup	Sign_NEQ	Verify_NEQ	WitGen_NEQ	WitVerify_NEQ
N	Valid	100	100	100	100	100
	Missing	0	0	0	0	0
Mean		.63585043	.04005798	.77853187	.01894745	.73169164
Std. Error of Mean		.006340467	.000689660	.007831064	.000253905	.007481758
Std. Deviation		.063404667	.006896596	.078310639	.002539053	.074817576
Minimum		.556614	.033674	.683132	.015339	.640395
Maximum		.921575	.080574	1.295758	.026817	1.049150

Figure A.16: KSS-192 - case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$

Statistics

		Setup	Sign_NEQ	Verify_NEQ	WitGen_NEQ	WitVerify_NEQ
N	Valid	100	100	100	100	100
	Missing	0	0	0	0	0
Mean		2.76751633	.11803778	3.69863770	.06688943	3.56690182
Std. Error of Mean		.013109996	.001043453	.015000657	.000775039	.017614290
Std. Deviation		.131099963	.010434528	.150006573	.007750391	.176142898
Minimum		2.534189	.105004	3.420859	.058391	3.277613
Maximum		3.410596	.175729	4.254405	.100527	4.234271

Figure A.17: BLS-256 - case $\mathcal{R} = \mathcal{R}^{\overline{EQ}}$

Statistics

		Setup	Sign_NEQ	Verify_NEQ	WitGen_NEQ	WitVerify_NEQ
N	Valid	100	100	100	100	100
	Missing	0	0	0	0	0
Mean		8.25163567	.27742023	6.96834772	.19329811	6.79467358
Std. Error of Mean		.057380459	.002752793	.045915838	.002122112	.051680485
Std. Deviation		.573804592	.027527928	.459158383	.021221117	.516804848
Minimum		7.485995	.238961	6.296145	.165196	6.053818
Maximum		11.372930	.398193	9.522171	.267669	9.307346