2011

Self-certified digital signatures

Nan Li
University of Wollongong

Self-Certified Digital Signatures

A thesis submitted in fulfillment of the
requirements for the award of the degree

Master of Computer Science

from

UNIVERSITY OF WOLLONGONG

by

Nan Li

School of Computer Science and Software Engineering
September 2011

Dedicated to

My Family

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Nan Li
September 7, 2011

Abstract

Digital signatures are used for proving the authorship of a given message. It is an important primitive of modern cryptography. To verify a signature, a user has to be equipped with a valid public key of the signer. A public key certificate issued by a trusted third party is required for public key authentication. It is necessary to verify the validity of a public key prior to verifying a signature, through a Public Key Infrastructure (PKI). However, the complexity of certificate management [ARP03] is a problem. Although the notion of identity-based signatures [Sha85] is introduced as a solution, key escrow is still an inherent problem.

The efficiency of a signature scheme and the size of a signature are two important aspects in evaluating a signature scheme. Taking PKI-based ring signature schemes as an example, they are inefficient in a large scale of applications [ALSY07]. This is due to the transport and verification costs of public key certificates. Low computational cost for signature signing and verification processes is required in practice. This thesis provides an efficient scheme to solve public key certificate management and key escrow problems, and reduce the communication cost of ring signature schemes.

To eliminate the need of public key certificates from traditional PKI and the problem of key escrow in identity-based cryptography, the concept of *self-certified public keys* was put forth by Girault [Gir91]. In this thesis, we propose an efficient and novel self-certified signature scheme, which requires only one modular multiplication in signing with pre-computation. One of the features of our scheme lies in its batch verification in both single-signer and multi-signer settings. Pairing computations in the batch verification are independent from the number of signatures. Our scheme is proved to be secure in the random oracle model.

Two similar solutions of the certificate management problem and the key escrow problem were proposed in 2003, namely *certificateless public key cryptography* and *certificate-based cryptography*. In the signing process, both of them require two

pieces of information where one is from a trusted third party and the other is chosen by a user itself. The validity of a user's public key is implicitly verified during the signature verification. However, it is different in self-certified signature schemes. The user's public key can not only be implicitly verified in the verification algorithm, but can also be computed explicitly. The computational cost of signature verification is reduced since there is no need to verify the user's public key after a valid key has been recovered. However, in the either certificateless signature schemes or certificate-based signature schemes, public key verifications are indispensable.

We also present a new notion called *self-certified ring signatures* (SCRS), to provide an alternative solution to the certificate management problem in ring signatures and eliminate the key escrow problem in identity-based ring signatures. A precise definition and elaborated security model of SCRS are provided, along with a concrete construction. We prove that our proposed scheme is secure in the random oracle model. This scheme captures all features of ring signatures, and exhibits the advantages of low storage, communication and computation cost.

Acknowledgement

First of all, I am most grateful to my supervisor Professor Yi Mu, for providing precious suggestions and patient guidance during my study. It is impossible to achieve my goals without his support.

I would like to thank my co-supervisor Professor Willy Susilo who gave valuable advices about my research work. My appreciation also goes to all people who have provided helpful discussions and suggestions for my research, including Dr. Wei Wu, Mr Fuchun Guo, and Mr Jinguang Han. I would also thank all staff of Centre for Computer and Information Security Research and School of Computer Science and Software Engineering. I appreciate all anonymous reviewers who review my papers.

I am very thankful to my parents, for their support. Without their help, my research career is unsustainable and this work would never be possible.

I would like to thank my wife Bai Xue, for her understanding and love. She always encourages me and helps me move forward when I encounter problems.

Nan Li

Wollongong, May 2011

Publications

Nan Li, Yi Mu, Willy Susilo and Fuchun Guo. Self-certified ring signatures. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS '11*, pages 396–400. ACM, 2011.

Nan Li, Yi Mu and Willy Susilo. Efficient self-certified signatures with batch verification. *7th China International Conference on Information Security and Cryptology, Inscrypt 2011*. (submitted)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cryptography is originally a study of information hiding which provides the secrecy of data by outputting a secret code. The procedure is called the *data encryption*. To protect the confidentiality of a message, a secret code needs to satisfy a requirement that the plaintext can only be found by a user who holds a valid key. Cryptography has been widely employed for secure communications. However, with the development of more sophisticated techniques of attack and security requirements, more complex structures and types of cryptographic algorithms are required.

In modern cryptography, the security of a cryptographic scheme normally depends on mathematical theory. The notion of mathematical cryptography was firstly put forth by Shannon in 1949 [Sha49]. Basically, most of modern cryptographic algorithms or protocols are used for protecting the security of a communication channel. That is, during a communication between Alice and Bob, deployed cryptographic techniques have to prevent any malicious third party from damaging the transport of data or deciphering secret messages. Generally speaking, four security requirements that are required for secure communication are [Bon04]: 1) No one can read the secret content except an authorized user; 2) A sender can prove his/her identity to a receiver; 3) A message cannot be modified by an unauthorized party, otherwise a user can detect and reject the message; 4) A user who sent a message cannot deny the ownership of the message. To satisfy these security requirements, data encryption and digital signatures, have been widely studied and employed.

Two branches in modern cryptography are symmetric-key cryptography and public key cryptography. In symmetric-key cryptographic schemes, since both a sender and a receiver share the same secret key, the property of message confidentiality is provided. It cannot however provide the identity of a key holder. In other words, symmetric-key algorithms normally do not satisfy the last three security requirements described above. On the other hand, a public key cryptographic schemes

employ two different keys, which are referred to as *public key* and *private key*. A public key is mathematically related to a private key and is bound to a unique user by a trusted third party. The public key can be published, while the private key is kept secretly. The user's identity can be detected by his/her public key. Therefore, public key cryptography is a solution to identify the relationship between a received message and a sender.

Digital signatures which apply the public key cryptography play an important role in modern cryptography. They primarily provide properties of data integrity and authenticity. The notion was proposed by Diffie and Hellman [DH76] in 1976. A digital signature is a receipt of a received message, which means a valid digital signature can prove the genuineness of the relationship between a message and a sender. A user cannot deny a valid signature which is generated by his/her private key. Furthermore, anyone who does not hold the private key cannot forge a valid message-signature pair with a non-negligible advantage in polynomial time. Digital signatures are usually used as *digital fingerprints* in commerce and government. For example, an online auction can employ digital signature schemes to avoid malicious participants. Sending a digital signature along with a bidding, a bidder cannot deny his/her operation. Since the digital signature is unforgeable, smart cards apply digital signatures to provide access control and identification. E-voting, as another example, employs a special kind of digital signature to protect the validity of a ticket, while the voter's identity is hidden.

With different security requirements desired in practice, numerous types of digital signature schemes have been proposed to achieve additional goals. For example, to hide a signer's identity in digital signatures, Rivest, Shamir, and Tauman [RST01] introduced a notion of ring signature. That is, a signer's identity is anonymous to receivers. Moreover, the notion of undeniable signatures was put forth by Chaum and Antwerpen [CA89] in 1989. The main feature is that the validity of a signature is only verifiable by a designated verifier, meanwhile, a signer has ability of proving a given signature is a forgery. In this thesis, we will study a special type of digital signature and describe two signature schemes which have been proved to be secure.

1.1 Public Key Infrastructure

In a public key cryptosystem, to prove that a public key is genuine and authentic, and has not been tampered with or replaced by a malicious third party is the central

problem. Usually certificates are employed to ensure the authenticity of a public key. A public key certificate issued by a certificate authority (CA) is used for public key authentication. Practically, the implementation of certificates needs the support of Public Key Infrastructure (PKI), which defines a set of people, policies, hardware, software and procedures needed to create, store, manage, distribute and revoke public key certificates [TS10, Sta06].

Although the PKI facilitates key cryptosystems, the computational overhead encountered in practice is undesired. Certificate management that includes revocation, storage, distribution and the cost of certificate verification is considered as a problem of the PKI [ARP03, Gut02]. The problem is even more serious in restricted bandwidth communication environments [DGSW02]. Typically, the certificate management process is not considered in signature schemes. A sender who signs a message needs to distribute his/her public key certificate to a receiver along with a message-signature pair. On the other hand, a receiver first checks the certificate to determine the validity of the public key. It is possible to verify a PKI-based digital signature only if the public key is valid. The cost for certificate management cannot be ignored.

Ring signatures [RST01] are signer-ambiguous signatures. A signer chooses a set of ring members including himself and signs a message, however, anyone who receives the signature cannot distinguish from the actual signer in a given set. In traditional PKI-based ring signatures [RST01, RST06], the certificate management problem impacts the efficiency of a system. Both a sender and a receiver have to certify identities of ring members and corresponding public keys. However, ring members are not predefined, which implies a signer needs to transport all members' certificates to a verifier. Considering an extreme example such as e-voting [ALSY07], if there are one million ring members, a large amount of cost is expended to certificate transport and verification has to be consumed. Obviously, it is not practical in the real world, thus, ring signature schemes are not suitable for large scale applications.

1.2 Challenging Issues and Related Work

A solution to the certificate management problem in signatures is identity-based signatures (IBS), which was put forth by Shamir [Sha85]. Identity-based signature schemes [CC03, Wat05] employ user identities as their pubic keys. Since a private key has been mathematically bound to a unique identity during the key generation

phase by a trusted third party, there is no need to use certificates in public key authentications. The trusted third party is referred to as a *Private Key Generator* (PKG). In a signature verification process, a verifier can directly use a user's identity as the public key to check a signature without certifying the validity of the key. Obviously, the certificate management problem has been eliminated by using the notion of identity-based signatures. Unfortunately, a new problem called *key escrow* problem becomes an inherent problem, where a third party who is dishonest can abuse users' private keys.

Green and Hohenberger [GH07] proposed a solution using a blind key extraction in 2007. They described a protocol that a PKG has no secret information about a user after a private key is obtained by a user. In 2009, Yuen, Susilo, and Mu [YSM10] presented a construction of identity-based signature scheme without the key escrow problem. The proposed scheme is escrow-free and a malicious PKG can be detected if it impersonates any user. Different from traditional IBS, a user needs two private pieces of information, namely the identity-based secret key and the user secret key, to sign a message. A PKG that only has the identity-based secret key does not have the ability of generating a valid signature. However, the size of such signature is questionable. Applying threshold signatures can alleviate the key escrow problem [YCK04], but the key can be exposed by collusion.

A certificate is a proof of the relationship of a public key and an identity. Typically, the public key infrastructure is employed to distribute and manage certificates. It is a mature but complex system. The *certificateless public key cryptography* (CL-PKC) is proposed by Al-Riyami and Paterson [ARP03] in 2003. It aims to avoid the use of certificate in data encryption and digital signature schemes. Like identity-based signatures, there is a trusted third party called *Key Generation Center* (KGC) who generates partial private keys for users. A private key is computed by a partial private key and a secret value chosen by the user. Hence, the KGC cannot compromise a user's private key since the secret value is unknown to it. Moreover, user's identities are certified during the partial private key generation process and the validity of the key can be implicitly checked in a signature verification process.

In certificateless public key cryptography, a user key generation process is divided into four parts: Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key and Set-public key [ARP03]. Obtaining a partial private key from a KGC, a user can pick a secret value and generate a pair of private and public keys. Certificateless signature schemes [ARP03, HWZD07, ZZ08] are normally based on costly pairing

computation, the computational efficiency of a scheme has to be considered. An efficient certificateless signature scheme was proposed by Choi, Park, Hwang and Lee [CPHL07] in 2007. The scheme requires only one paring operation and it is provably secure in the random oracle model. Yap, Chow, Heng, and Goi [YCHG07] introduced a security mediated certificateless signature scheme without pairing operations. However, the signing algorithm in the proposed scheme needs the help of an online security mediator (SEM) who is a semi-trusted server. Users are required to communicate with a SEM during the signature generation.

Certificateless public keys are implicitly certified in a signature verification algorithm. However, a verifier cannot believe a public key without a certificate. Once a verification algorithm outputs *false*, a user does not know if a public key or a signature is invalid. Therefore, certificateless public key signature schemes have to consider an attacker who has the ability of replacing a user's public key.

In Eurocrypt 2003, Gentry [Gen03] put forth the notion of *certificate-based encryption* (CBE) to solve the certificate management problem in traditional PKI. Similar to certificateless public key cryptography, the CBE does not have an explicit certificate to prove a user public key. Later, the idea was extended to certificate-based signatures (CBS) [KPH04, LHM$^+$07]. Although it is not necessary to employ the CBS to resolve the certificate management problem [Gen03], the CBS has advantages over the identity-based signatures and certificateless signatures.

A trusted third party in CBS is referred to as *certificate authority* (CA) who is different from the PKG and KGC. To generate a certificate-based signature, a user has to hold secrets, namely a user private key sk and a valid certificate of the corresponding public key. Due to the fact that the private key sk is unknown to the CA, the third party cannot impersonate a user. In addition, a certificate transmission does not need a secure channel which is necessary in both identity-based signatures and certificateless signatures.

Certificate-based signatures [KPH04, LHM$^+$07] are similar to certificateless signatures. But a partial private key in CL-PKC is replaced by a certificate issued by a CA. In certificate-based signatures, a certificate captures all features of public key certificates in the traditional public key infrastructure. It can also be individually published and verifiable. Wu, Mu, Susilo, and Huang [WMSH08, WMSH09] described a generic construction that converts a certificateless signature scheme to a certificate-based signature scheme. Moreover, they presented security levels of certificate-based signature schemes. An efficient certificate-based signature scheme

without pairings was proposed by Liu, Baek, Susilo, and Zhou in 2008 [LBSZ08]. The scheme applied a technique called non-interactive proof-of-knowledge [CS97] to prove the validity of a presented public key. Unfortunately, the size of users' public keys and signatures have to be increased, it might not be suitable for some limited bandwidth applications.

The efficiency of signature schemes is important in some applications. Smart card, as an example, requires a fast signing process since the computing power of a card is restricted. In Crypto 1989, Even, Golreich, and Micali [EGM90] firstly introduced a notion of *online/offline* digital signatures. Their motivation is to improve the speed of signature generation procedure. In such schemes, a signing process is divided into two phases, namely the offline phase and the online phase. Most of heavy computations are pre-computed in the offline phase where the powerful computing environment is provided. The online phase issues a very fast algorithm to sign messages using stored results of offline phase. Golreich, and Micali [EGM90] proposed the first generic construction to convert any signature into an online/offline signature scheme. However, the large size of signatures is a drawback.

In 2001, Shamir and Tauman [ST01] presented an efficient construction for converting signature schemes into online/offline signature schemes. Using the notion of trapdoor hash functions [KR00], they introduced a new method called *hash-sign-switch*. In the proposed construction, the online signing algorithm is efficient and the size of online/offline signatures is only twice as the size of original signatures. Moreover, the security of the original signature scheme is also enhanced.

A notion of *batch verification* was put forth by Fiat [Fia90, Fia97] to save the cost of signature verifications. The aim is to verify a batch of signatures from single or multiple signers at one time. It is useful in centralized applications and repetitive tasks. For example, a bank server can check all transactions simultaneously in the midnight. Ballare, Garay, and Rabin [BGR98] described three techniques of batch verification, which are random subset test, small exponents test, and bucket test. Generally, small exponents test is fast but not suitable for a large number of signatures. Instead, bucket test receives a better result in this case. However, Boyd and Pavlovski [BP00] found general attacks for batch verification schemes [YL95, Har98]. Fortunately, a repair of these attacks is also provided. Actually, many signature schemes [BLS04b, PS06, Wat05] provide the property of batch verification and schemes based on bilinear parings with batch verification were also proposed [GMC08, FGHP09, CHP07, GZ09].

The notion of ring signatures was introduced by Rivest, Shamir, and Tauman [RST01] in Asiacrypt 2001. It was developed from a related notion of group signatures proposed by Chaum and Heyst [CvH91] in 1991. In group signatures schemes, a trusted group manager predefines a group of users and distributes keys to group members. Any user in the group can sign a message on behalf of the group and the signer anonymity is provided. However, the group manager who can revoke the anonymity of group members has the ability to distinguish the signer of a message. Ring signature schemes are simplified group signature schemes that there is no group manage required and users can decide a group themselves. The notion of ring signatures inherits the main feature of group signatures which is the signer anonymity. Moreover, as a group of users are temporarily defined by a user, ring signature schemes provide perfect anonymity that anyone cannot distinguish who is the actual signer from a presented group. A method applied to construct ring signature schemes for RSA-type keys and DL-keys was proposed by Abe, Ohkubo, and Suzuki [AOS02].

A limitation of traditional PKI-based ring signature schemes [RST01, XZF04, SW07, SS10] is the overhead of message signing and signature verification. Because n users participate in group, both signer and verifier have to authenticate the validity of ring members. In other words, the certificate management problem in PKI becomes serious. Hence, some solutions using identity-based signatures, certificateless signatures and certificate-based signatures have been proposed [ZK02, ALYW06, CY07, ZZW07a, ALSY07].

Typically, the size of a ring signature depends on the size of the group. It is undesired in some low bandwidth environment. A constant-size ring signature scheme was presented by Au, Liu, Susilo, and Yuen [ALSY06] in 2006. Additionally, the proposed scheme provides a new property called *Revoke-iff-Linkability*. That is, if a user signs twice, his identity can be detected and revoked by anyone. In some applications, such as ad hoc and e-cash, the size of traditional ring signatures may need to count identities, public keys and certificates of users. The reason is that a sender cannot ensure in an unstable environment a receiver has all related certificates of ring members.

1.3 Self-Certified Public Keys

The notion of *self-certified public keys* was introduced few years after the identity-based signatures was proposed. Girault [Gir91], in Eurocrypt 1991, introduced this idea to bridge a gap between the traditional PKI-based signatures and identity-based signatures. As certificateless signatures and certificate-based signatures, the self-certified signature is also a means of solving certificate management and key escrow problems mentioned above. All three types of signatures achieve the same goal, while in different ways.

In self-certified signatures, a trusted third party (TTP) is employed to generate an entity which is referred to a *witness* for every user. Unlike normal digital signatures, a user's public key is not explicitly given in self-certified signatures. Instead, it is implicitly computed and certified in a signature verification. Nevertheless, a user who holds a witness can also explicitly extract a public key. Thus the cost for public key computation can be saved after a successful certification. Once a signature is accepted by a receiver, he/she can believe that the issued witness is valid. Then receiver applies a key recovery algorithm to find a signer's public key and explicitly use it to verify a signature without public key verification operation.

Girault [Gir91] proposed the first signature scheme using self-certified public keys. A witness issued by a TTP is a RSA signature of a user's identity and a related public key. Anyone who has the witness, user's identity and TTP's public parameters can compute a user's public key. Girault described three security levels of self-certified public keys [Gir91].

- Level-1: A TTP knows the users' private keys and it has the capability to impersonate any user without being detected.
- Level-2: A TTP does not have users' private keys, however, it is still able to forge a witness and impersonate any user without being detected.
- Level-3: A TTP does not know users' private keys and it can be detected if a TTP uses false witness to impersonate any user.

Girault argued that the proposed scheme achieves the highest security level (level-3). However, Saeednia [Sae03] found that, for this scheme, it is possible for dishonest TTP to specially choose RSA modulus which is easy to solve the corresponding discrete logarithm problem in order to compromise users public keys.

A provably secure RSA-based self-certified signature scheme was introduced by Zhou, Cao, and Lu [ZCL04] in 2004. The idea is that a RSA modulus is chosen by a user itself. Although the proposed scheme is secure in the random oracle model, the signature verification process has to be divided into two steps. It is due to the different RSA moduli chosen by a user and a TTP.

Self-certified public keys based on discrete logarithm is proposed by Petersen and Horster [PH97] in 1997. Based on the idea of *weak blind Schnorr signatures* [HMP95], they presented a protocol to distribute users' witnesses. A TTP who generates users' witnesses does not have any knowledge about users private keys. Moreover, a self-certified public key can be hierarchically verified and some applications are given in the paper. However, there is no security proofs for proposed schemes which are claimed secure.

1.4 Aims and Contributions

This thesis focuses on self-certified digital signatures and their formal security proofs. Firstly, our proposed solutions aim at avoiding certificate management problem, key escrow problem, and preventing malicious TTP, as well as capturing all features of self-certified signature schemes. It is reasonable that self-certified public keys are deserving to use in multi-user environments. Secondly, our work intends to boost the speed of self-certified signatures in both signing and verification processes. We consider our main contributions in two aspects.

In digital signatures, the online/offline signature [EGM90] is adopted to speed up the signature generation process and the batch verification [Fia90, Fia97] improves the efficiency of signature verifications. Our proposed efficient self-certified signature scheme naturally enjoys both of these features. Providing pre-computation, the cost of signing for a given message can be largely reduced. In some scenarios, it is required to verify a batch of signatures at one time. The scheme affords efficient batch verifications for both single-signer and multi-signer settings. Moreover, the batch verification in multi-signer setting requires constant pairing calculations defined in Section 2.1.3 regardless of the number of different signers. In addition, we give two methods of verifying a self-certified signature using a witness or a recovered public key respectively. The scheme also inherits advantages of original self-certified signatures. Thus, our scheme is suitable for limited computation power and low bandwidth applications.

We introduce a novel notion of self-certified ring signature schemes. The proposed scheme does not require any certificates to check the validity of ring members' public keys. As a composite element, a witness is applied as a public key, which is implicitly verified in the signature verification. Meanwhile, a witness can be used as a traditional public key certificate that a user's public key can be explicitly extracted from a valid witness. Hence, the certificate management in traditional ring signature schemes is removed. Although identity-based ring signature schemes [LW04, HS04, Her07] have lower cost since user's identity is a public key, the key escrow is a potential security problem. Our scheme is provably secure in the random oracle model.

1.5 Overview of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 provides some background of cryptography, which include cryptographic hash functions, bilinear maps, mathematical complexity problems, definitions of several different digital signature schemes. The forking lemma and random oracle model are also introduced as two tools for security proofs in this thesis.

In Chapter 3, an efficient self-certified signature with batch verification is proposed. We present a formal definition of self-certified signature schemes and their security model. Relationships among the proposed scheme, online/offline signatures and batch verification are presented. A formal security proof in the random oracle is also provided.

In Chapter 4, a novel notion of self-certified ring signatures is introduced. We extend it to combine the conception of self-certified public keys and ring signatures. A formal definition of self-certified ring signature schemes and their security requirements are elaborately proposed. Furthermore, we prove that the proposed scheme is secure in the random oracle model.

Finally, we conclude the thesis in Chapter 5.

Chapter 2

Background

Cryptography background and some mathematical definitions are given in this chapter. Provided notions are the underlying knowledge through the thesis.

2.1 Preliminaries

2.1.1 Cryptographic Hash Functions

Cryptographic hash functions [RS04] are widely used in modern cryptography. An important application of hash functions is to verify the data integrity, such as the message authentication code (MAC). Hence, most of digital signature schemes apply hash functions to prove the message integrity.

Definition 2.1 *A hash function is a deterministic function that takes as input an arbitrary length bit string, outputs a fixed length bit string which is called a hash value. We denote a hash function H as a mapping*

$$H : \{0,1\}^* \rightarrow \{0,1\}^n,$$

where n is the length of a hash value.

A typical paradigm in digital signature schemes is so-called *hash-and-sign* that requires hash functions have the following properties:

- **Pre-image resistance**: Given a hash value h, it is computationally infeasible to find an input m such that $H(m) = h$.
- **Collision resistance**: Basically, the collision is to find two different inputs that output the same hash value. Two types of collision resistance are defined as follows:

- Strong collision resistance: It is computationally infeasible to find a pair of input string (m, m'), where $m \neq m'$, such that $H(m) = H(m')$ [Dam87, BR97].
- Weak collision resistance: Given an input string m, it is computationally infeasible to find another input string m', where $m \neq m'$, such that $H(m) = H(m')$ [NY89].

- **Computational efficiency**: Given an input string m, the hash value $H(m)$ can be found efficiently.

Definition 2.2 *A cryptographic hash function is a hash function that provides above properties.*

Practically, the length of hash values is different among cryptographic hash functions. The output length is normally decided by the security requirement. Generally speaking, a desired cryptographic hash function should have a sufficient large output size. For example, Rivest's well-known hash function MD5 [Riv92] with a 128-bit hash value has been broken. Some other broadly used cryptographic hash functions are proposed in [ZPS92, KR00, AdM04]. In the rest of the thesis, we consider the length of hash values as 160 bits.

2.1.2 Random Oracle Model

In 1993, Bellare and Rogaway [BR93] introduced the *random oracle model* into security proofs. They refer an ideal cryptographic hash function as a random oracle that outputs *uniform* hash values. Once the random oracle model is applied as a technique for security proof, hash functions in the scheme can be replaced by such random oracles (functions). In other words, an adversary has to query hash values due to the random function is unknown to him/her. A party, called the *challenger* in the model, handles random oracles to simulate corresponding outputs for any queries. Since the assumption that output hash values are truly random does not hold in realistic, a scheme which is secure in random oracle model may be insecure in practice [CPS08].

2.1.3 Bilinear Maps

Bilinear maps (pairings) based cryptography is a hot topic in the last decade. Generally, a bilinear map is a function that maps two vector space elements to another vector space element. Two important bilinear parings used in cryptography are Weil pairing [BF01] and Tate pairing [FMR99, BLS04a]. The early application of bilinear parings was to attack elliptic curve cryptography (ECC) in 1993 [MOV93]. Later on, this tool has been used to design cryptographic protocols. Numerous pairing-based encryption and signature schemes are proposed (e.g, [BB04, BLS04a]). We briefly review the definition of bilinear maps as follows:

Definition 2.3 *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of same large prime order p. g is a generator of $\mathbb{G}$. The map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear mapping (pairing) and $(g, p, \mathbb{G}, \mathbb{G}_T, e)$ is a symmetric bilinear group. Some properties of bilinear pairing are as follows:*

- **Bilinearity:** *For all $u, v \in \mathbb{G}$ and for all $a, b \in \mathbb{Z}_p^*$, we have the equation $e(u^a, v^b) = e(u, v)^{ab}$.*
- **Non-Degeneracy:** *For all $g \in \mathbb{G}$, if g is a generator of $\mathbb{G}$, we have $e(g, g) \neq 1$ is a generator of $\mathbb{G}_T$.*
- **Efficiency:** *There is an efficient algorithm to calculate $e(u, v)$ for all $u, v \in \mathbb{G}$.*

2.1.4 Forking Lemma

Pointcheval and Stern [PS96] firstly introduced the concept of *forking lemma* into the security proof for digital signature schemes. The idea is to use the oracle replay attack that outputs two different valid signatures within the same random tape and different random oracles to solve some underlying hard problems of the scheme. Being proved by Pointcheval and Stern [PS96], the forking lemma is a useful method of proving security of digital signature schemes. Unfortunately, it is restricted in the random oracle model. A precondition of using forking lemma is that two different hash functions can be used in a scheme. In fact, hash functions are usually fixed in signature schemes. Hence, the requirement can only be satisfied with the help of random oracles. A generalized version of forking lemma was presented by Bellare and Neven [BN06] in 2006. We review the definition of forking lemma in signatures as follows:

Definition 2.4 *Let $\mathcal{A}$ be an adversary (probabilistic polynomial time turning machine), given public parameters as input, if $\mathcal{A}$ has non-negligible probability to find a tuple (m, r, σ, h), where σ is a valid signature of a message m and h is the oracle output of a random tape r, then $\mathcal{A}$ has non-negligible probability to find another valid tuple (m, r, σ', h') with the same random tape r and different random oracles such that $h \neq h'$.*

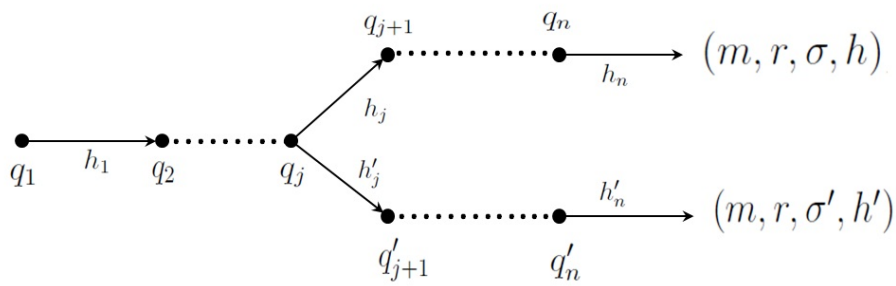

Figure 2.1: Forking Answers to Random Oracle Queries

2.2 Complexity Problems

The security of modern cryptography is based on some underlying mathematical problems which are computationally hard in polynomial time. In computational complexity theory, if there is no deterministic Turing machine that can solve a problem in polynomial time with the error probability bounded by $\delta \in [0, \frac{1}{2})$, we say that the problem is a non-deterministic polynomial time (NP) problem or simply called *hard problem* [Mao04]. On the other hand, we say that a problem is a P problem if there exists a Turing machine that can solve the problem in polynomial time. Most modern cryptographic schemes rely on the widely believed NP problems, even there is no proof that we cannot find a deterministic Turing machine to solve the problem. For example, the large integer factorization and discrete logarithm problems are two basic NP problems and many variants have been developed. In this section, we introduce some hard problems that will be used in the following chapters. For more hard problems and their utilizations in cryptography, please refer to [DBS04].

2.2.1 Discrete Logarithm Problem

The discrete logarithm problem is a basic and widely used hard problem in cryptosystems [McC90]. In abstract algebra, the discrete logarithm is an analogue of ordinary logarithm in the group theory over the real or complex numbers. Let $\mathbb{G}$ be a cyclic multiplicative group and $g \in \mathbb{G}$ is a generator of group $\mathbb{G}$. We say that the solution a, such that let the equation $b = g^a$ hold, is a discrete logarithm to the base g of b in the group $\mathbb{G}$ and denote $a = log_g(b)$. An efficient algorithm [PH78] to solve the discrete logarithm problem requires the complexity as $O(\sqrt{q})$, where $q = |\mathbb{G}|$. However, it is not sufficiently efficient in practice. The discrete logarithm problem can be described as:

Definition 2.5 *Let $\mathbb{G}$ be a multiplicative cyclic group of order p. g is a generator of group $\mathbb{G}$. $\mathbb{Z}_p$ is a finite field. Given $g^a \in \mathbb{G}$, the discrete logarithm (DL) problem on $\mathbb{G}$ is to compute $a \in \mathbb{Z}_p$.*

The DL problem is (t, ϵ)-hard, if there is no probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ can solve the DL problem in time at most t with advantage ϵ if

$$\text{DL Adv}_{\mathcal{A}} = \Pr[a \leftarrow \mathcal{A}(g, g^a) : a \in_R \mathbb{Z}_p] \geq \epsilon.$$

2.2.2 Computational Diffie-Hellman Problem

In 1976, Diffie and Hellman firstly proposed a new mathematical problem called *Diffie-Hellman* (DH) problem [DH76]. The problem is said to be hard as the computational complexity of Diffie-Hellman problem is close to the discrete logarithm problem. That means the DL problem can be solved in polynomial time if there exists an algorithm can solve the DH problem in polynomial time. Typically, some reductions are given from the DL problem to the DH problem [Mau94, BL96]. The variations of Diffie-Hellman problem and their hardness are presented in [BDZ03]. Usually, the Diffie-Hellman problem refers to the *Computational Diffie-Hellman* (CDH) problem. The CDH problem is one of well-known hard problems in cryptography.

Definition 2.6 *Let $\mathbb{G}$ be a multiplicative cyclic group of order p. g is a generator of group $\mathbb{G}$. Given two values $g^a, g^b \in \mathbb{G}$, where a, b are unknown integers that $0 < a, b < p$, the computational Diffie-Hellman problem on $\mathbb{G}$ is to compute g^{ab}.*

The CDH problem is (t, ϵ)-hard, if there is no PPT algorithm $\mathcal{A}$ that can solve the CDH problem in time at most t with advantage ϵ if

$$\text{CDH Adv}_{\mathcal{A}} = \Pr[g^{ab} \leftarrow \mathcal{A}(g, g^a, g^b) : a, b \in_R \mathbb{Z}_p] \geq \epsilon.$$

2.2.3 Weak Computational Diffie-Hellman Problem

Mitsunari, Sakai and Kasahara proposed a new hard problem called *k-weak Computational Diffie-Hellman* (k-wCDH) problem in 2002 [MSK02]. The problem is hard if and only if there is no probabilistic polynomial time algorithm can solve the hard problem *Collusion Attack Algorithm with k traitors* (k-CAA) [MSK02]. We give the definitions for the k-CAA and k-wCDH problems as follows:

Definition 2.7 *Let $\mathbb{G}$ be a multiplicative cyclic group of order p. g is a generator of group $\mathbb{G}$. $\mathbb{Z}_p$ is a finite field. Given an instance*

$$< g, g^a, h_1, \ldots, h_k \in \mathbb{Z}_p, g^{\frac{1}{h_1+a}}, \ldots, g^{\frac{1}{h_k+a}} >,$$

where k is an integer and $a \in_R \mathbb{Z}_p$, the collusion attack algorithm with k traitors on $\mathbb{G}$ is to compute $g^{\frac{1}{h+a}}$ for some $h \notin \{h_1, \ldots, h_k\}$.

The k-CAA problem is (t, ϵ)-hard, if there is no PPT algorithm $\mathcal{A}$ that can solve the k-CAA problem in time at most t with advantage ϵ if

$$\text{CAA Adv}_{k,\mathcal{A}} = \Pr\left[\begin{array}{l} g^{\frac{1}{h+a}} \leftarrow \mathcal{A}(g, g^a, h_1, \ldots, h_k, g^{\frac{1}{h_1+a}}, \ldots, g^{\frac{1}{h_k+a}}) : \\ a \in_R \mathbb{Z}_p, h_i \in \mathbb{Z}_p, h \neq h_i, i \in \{1, \ldots, k\} \end{array}\right] \geq \epsilon.$$

Definition 2.8 *Let $\mathbb{G}$ be a multiplicative cyclic group of order p. g is a generator of group $\mathbb{G}$. $\mathbb{Z}_p$ is a finite field. Given $k+1$ values $< g, g^a, g^{a^2}, \ldots, g^{a^k} >$, where k is an integer and $a \in_R \mathbb{Z}_p$, the k-weak computational Diffie-Hellman problem is to compute $g^{\frac{1}{a}}$.*

The k-wCDH problem is (t, ϵ)-hard, if there is no PPT algorithm $\mathcal{A}$ that can solve the k-wCDH problem in time at most t with advantage ϵ if

$$\text{wCDH Adv}_{k,\mathcal{A}} = \Pr[g^{\frac{1}{a}} \leftarrow \mathcal{A}(g, g^a, \ldots, g^{a^k}) : a \in_R \mathbb{Z}_p] \geq \epsilon.$$

2.2.4 k+1 Exponent Problem

The $k+1$ *exponent problem* (k+1EP) was introduced by Zhang, Safavi-Naini and Susilo [ZSNS04] in 2004. The hardness of $k+1$ exponent problem is proved that it is polynomial time equal to the k-wCDHP. However, we should notice that both k+1EP and k-wCDH problem are no harder than the CDH problem.

Definition 2.9 *Let $\mathbb{G}$ be a multiplicative cyclic group of order p, g is a generator of group $\mathbb{G}$. $\mathbb{Z}_p$ is a finite field. Given $k+1$ values $< g, g^a, g^{a^2}, \ldots, g^{a^k} >$, where k is an integer and $a \in_R \mathbb{Z}_p$, the k+1 exponent problem is to compute $g^{a^{k+1}}$.*

The k+1EP is (t, ϵ)-hard, if there is no PPT algorithm $\mathcal{A}$ can solve the k+1EP in time at most t with advantage ϵ if

$$\text{EP Adv}_{k,\mathcal{A}} = \Pr[g^{a^{k+1}} \leftarrow \mathcal{A}(g, g^a, \ldots, g^{a^k}) : a \in_R \mathbb{Z}_p] \geq \epsilon.$$

2.3 Digital Signatures

The notion of digital signatures was envisioned by Diffie and Hellman [DH76] in 1976. In public key cryptography, the signer holds his/her private key that can generate a signature of a message by a one-way trapdoor function. Any receiver can verify the correctness of a signature via the signer's public key. A digital signature is used to be a proof of the authorship of a message. Hence, digital signatures should satisfy some requirements [RSA78]: 1) A valid signature can prove that the message has been signed by the signer; 2) Only the signer can generate a valid signature by his/her private key; 3) The signer cannot deny a valid signature of a message that signed by his/her private key; 4) A valid signature implies that the message has not been modified.

Definition 2.10 *A digital signature scheme consists of three algorithms:* KeyGen*,* Sign *and* Verify*.*

- KeyGen*: A PPT algorithm run by a user that takes as input a security parameter k, outputs a pair of private and public keys (sk, pk), where*

$$(sk, pk) \leftarrow \mathsf{KeyGen}(k).$$

- Sign*: A PPT algorithm run by a user that takes as input a message m and a private key sk, outputs a signature σ, where*

$$\sigma \leftarrow \mathsf{Sign}(m, sk).$$

- Verify*: A deterministic algorithm that takes as input a message m, a signature σ and the signer's public key pk, outputs true if the signature is valid, otherwise outputs false.*

$$\{true, false\} \leftarrow \mathsf{Verify}(m, \sigma, pk).$$

2.3.1 Security Notion

Unforgeability is the underlying security notion of digital signatures. That is, an adversary is computationally infeasible to find a valid message-signature pair without the signer's private key. Once the pair passes the signature verification algorithm Verify and the pair is not an output of an authorized signer, we say that is a valid forgery. Basically, there are four security levels of digital signatures [GMR88].

- Existential forgery: The adversary has a probability to find a valid forgery for a message.
- Selective forgery: The adversary can find a valid forgery for the chosen message.
- Universal forgery: The adversary can find a valid forgery for any message without the knowledge of the private key.
- Total break: The adversary can recover the signer's private key.

In the security proof of digital signature schemes, we usually consider the strongest security notion *existential unforgeability against adaptive chosen-message attacks* (EUF-CMA) [GMR88]. A signature scheme is said to be secure under EUF-CMA if an attacker who receives valid signatures issued by an oracle for any queried message (where the message may be a special choice based on the previous message-signature pair), he/she cannot generate a new valid forgery.

2.4 Ring Signatures

In 2001, Rivest, Shamir, and Tauman put forth the notion of ring signatures. It simplifies the conception of group signatures [CvH91]. A group of users who are not

predefined in ring signature schemes can be randomly selected. The identity of a user who signs a message is hiding in the group. No one can distinguish from an actual signer in the group. This property is referred to as *anonymity*.

Definition 2.11 *A ring signature scheme consists of three algorithms:* KeyGen, Sign *and* Verify.

- KeyGen*: A PPT algorithm run by a user that takes as input a security parameter k, outputs a pair of private and public keys (sk, pk),*

$$(sk, pk) \leftarrow \mathsf{KeyGen}(\mathsf{k}).$$

- Sign*: A PPT algorithm run by a user that takes as input a message m, a set of identities $\bigcup_{i=0}^{n-1}\{ID_i\}$, the corresponding set of public keys $\bigcup_{i=0,i\neq k}^{n-1}\{pk_i\}$ and the signer's private key sk_k, where $k \in \{0, 1, \ldots, n-1\}$, outputs a ring signature σ, where*

$$\sigma \leftarrow \mathsf{Sign}(m, \bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0,i\neq k}^{n-1}\{pk_i\}, sk_k).$$

- Verify*: A deterministic algorithm run by a user that takes as input a message m, a signature σ, a set of identities $\bigcup_{i=0}^{n-1}\{ID_i\}$ and the corresponding set of public keys $\bigcup_{i=0}^{n-1}\{pk_i\}$, outputs true if the signature is valid, otherwise outputs false.*

$$\{true, false\} \leftarrow \mathsf{Verify}(m, \sigma, \bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0}^{n-1}\{pk_i\}).$$

However, there is a practical drawback in ring signatures. Prior to message signing and signature verification, both the signer and the verifier have to hold all public keys of members in the ring. In traditional public key infrastructure, that means a user needs to get and certify every member's public key before the signing and verification. This is related to the certificate distribution issue. Luckily, some solutions have been proposed, such as identity-based ring signatures [ZK02], certificateless ring signatures [CWMZ09] and certificate-based ring signatures [ALSY07]. More details will be given in Chapter 4.

2.5 Online/Offline Signatures

Online/Offline signature was firstly introduced by Even, Goldreich and Micali [EGM90] in 1990. The motivation is to enhance the security of existing digital signatures and improve the signing efficiency. In online/offline signature schemes, the signing algorithm consists of both offline and online phases. Prior to receiving a message, the offline part is to sign a random bit string using a basic signature scheme. Once the message is given, a user issues the message with a small amount of computations. Hence, online/offline digital signature schemes can be secure under EUF-CMA regardless of the security of the offline part signature scheme. In addition, if a trapdoor hash function is applied during the online phase, the computational cost can be reduced to one multiplication [ST01]. The overhead in online/offline signatures is that the offline part outputs have to be securely stored and ensure that every output is one-time used. We review the definition of online/offline signature schemes introduced by Shamir and Tauman [ST01].

Definition 2.12 *An online/offline signature scheme consists of four algorithms:* KeyGen, $\mathsf{Sign}^{\mathsf{off}}$, $\mathsf{Sign}^{\mathsf{on}}$ *and* Verify.

- KeyGen*: A PPT algorithm run by a user that takes as input a security parameter k, outputs a pair of private and public keys (sk, pk), a pair of hash and trapdoor keys (hk, tk),*

$$\{(sk, pk), (hk, tk)\} \leftarrow \mathsf{KeyGen}(\mathsf{k}).$$

- $\mathsf{Sign}^{\mathsf{off}}$*: A PPT algorithm run by a user in the offline phase that takes as input a random chosen pair (m', r'), a private key sk and a hash key hk, outputs a signature σ' and a hash value $H_{hk}(m', r')$, where*

$$\{\sigma', H_{hk}(m', r')\} \leftarrow \mathsf{Sign}^{\mathsf{off}}(m', r', sk, hk),$$

  the tuple $< m', r', \sigma', H_{hk}(m', r') >$ is stored.

- $\mathsf{Sign}^{\mathsf{on}}$*: A PPT algorithm run by a user in the online phase that takes as input a message m, a tuple $< m', r', \sigma', H_{hk}(m', r') >$, a hash key hk, outputs a signature σ, where*

$$\sigma \leftarrow \mathsf{Sign}^{\mathsf{on}}(m, m', r', H_{hk}(m', r'), hk).$$

- Verify*: A deterministic algorithm run by a user that takes as input a message m, a signature σ, a public key pk and a hash key hk, outputs true if the signature is valid, otherwise outputs false,*

$$\{true, false\} \leftarrow \mathsf{Verify}(\mathsf{m}, \sigma, \mathsf{pk}, \mathsf{hk}).$$

2.6 Identity-Based Signatures

Shamir introduced the notion of *identity-based signatures* (IBS) in 1984 [Sha85]. The IBS is a kind of signature that allows a user to verify a signature through the signer's unique identity information (e.g., user's email address), which is referred to as his/her public key. A trusted third party called *Private Key Generator* (PKG) is desired to generate every user's private key according to his/her identity. Indeed, the PKG has a master secret key which is unknown to users. A user's private key can be seen as a signature of his/her identity information generated by PKG using the master secret key. Once a user expects to verify an identity-based signature, a PKG's master public key is required along with the signer's identity.

Definition 2.13 *An identity-based digital signature scheme consists of four algorithms:* Setup*,* KeyGen*,* Sign *and* Verify*.*

- Setup*: A PPT algorithm run by the PKG that takes as input a security parameter k, outputs a pair of master secret and public keys (msk, mpk) and public system parameters $params$,*

$$(msk, mpk, params) \leftarrow \mathsf{Setup}(\mathsf{k}).$$

- KeyGen*: A PPT algorithm run by the PKG that takes as input a user identity ID, $params$ and a master secret key msk, outputs a user private key sk,*

$$sk \leftarrow \mathsf{KeyGen}(ID, params, msk).$$

- Sign*: A PPT algorithm run by a user that takes as input a message m, $params$ and a private key sk, outputs an identity-based signature σ, where*

$$\sigma \leftarrow \mathsf{Sign}(m, params, sk).$$

- Verify*: A deterministic algorithm run by a user that takes as input a message m, a signature σ, $params$, a master public key mpk and the signer's identity ID, outputs true if the signature is valid, otherwise outputs false,*

$$\{true, false\} \leftarrow \mathsf{Verify}(m, \sigma, params, mpk, ID).$$

Since a public key is derived from the user's identify, the public key distribution infrastructure has been eliminated in IBS. Moreover, certificates are not required as a user's public key is implicitly certified during signature verification process. However, with the knowledge of users' private keys, a PKG can sign a message on behalf of any user. It is an inherited problem called *key escrow* problem in IBS.

2.7 Certificateless Signatures

In traditional PKI-based digital signatures, a certificate is adopted to prove a user's identity. However, it is inefficient in practice. Thus, Al-Riyami and Paterson [ARP03] introduced a conception of certificateless public key cryptography which does not require a public key certificate. A signer holds two secret pieces of information which are partial private key (PPK) and secret value to generate a signature. Since a user's identity has been authenticated in the process of applying a PPK, verifiers only need to use a presented public key to verify a signature.

Definition 2.14 *A certificateless public key digital signature scheme consists of five algorithms:* Setup*,* Partial-Private-Key-Extract*,* UserKeyGen*,* Sign *and* Verify*.*

- Setup*: A PPT algorithm run by the KGC that takes as input a security parameter k, outputs a master secret/public key pair (msk, mpk) and public system parameters $params$,*

$$(msk, mpk, params) \leftarrow \mathsf{Setup}(k).$$

- Partial-Private-Key-Extract*: A PPT algorithm run by the KGC that takes as input an entity's identification ID, $params$, a master public key mpk and a master secret key msk, outputs a partial private key ppk,*

$$ppk \leftarrow \mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}(ID, params, mpk, msk).$$

- UserKeyGen*: A PPT algorithm run by a user that takes as input a partial private key ppk, a user identity ID and params, outputs a pair of private and public keys (sk, pk),*

$$(sk, pk) \leftarrow \mathsf{UserKeyGen}(ppk, ID, params).$$

- Sign*: A PPT algorithm run by a user that takes as input a message m, params and a private key sk, outputs a signature σ, where*

$$\sigma \leftarrow \mathsf{Sign}(m, params, sk).$$

- Verify*: A deterministic algorithm run by a user that takes as input a message m, a signature σ, params, an identity ID and a public key pk, outputs true if the signature is valid, otherwise outputs false,*

$$\{true, false\} \leftarrow \mathsf{Verify}(m, \sigma, params, ID, pk).$$

2.8 Certificate-Based Signatures

The notion of certificate-based signature (CBS) is extended from CBE [Gen03]. In CBS, a trusted third party called certificate authority issues certificates for users. A user who holds a private key and corresponding certificate can sign a message. In a signature verification process, a user's public key is implicitly certified without being checked separated.

Definition 2.15 *A certificate-based digital signature scheme consists of five algorithms:* Setup*,* UserKeyGen*,* CertGen*,* Sign *and* Verify*.*

- Setup*: A PPT algorithm run by the CA that takes as input a security parameter k_1, outputs a master secret key msk, a master public key mpk and public system parameters params,*

$$(msk, mpk, params) \leftarrow \mathsf{Setup}(k_1).$$

- UserKeyGen*: A PPT algorithm run by a user that takes as input a security parameter k_2 and params, outputs a pair of private and public keys (sk, pk),*

$$(sk, pk) \leftarrow \mathsf{UserKeyGen}(k_2, params).$$

- CertGen*: A PPT algorithm run by the CA that takes as input a user public key pk, an identity information ID, $params$, a master public key and a master secret key msk, outputs a certificate $Cert$,*

$$Cert \leftarrow \mathsf{CertGen}(pk, ID, params, mpk, msk).$$

- Sign*: A PPT algorithm run by a user that takes as input a message m, $params$, a certificate $Cert$ and a private key sk, outputs a certificate-based signature σ, where*

$$\sigma \leftarrow \mathsf{Sign}(m, params, Cert, sk).$$

- Verify*: A deterministic algorithm run by a user that takes as input a message m, a signature σ, an identity ID, $params$, a master public key mpk and a user public key pk, outputs true if the signature is valid, otherwise outputs false,*

$$\{true, false\} \leftarrow \mathsf{Verify}(m, \sigma, ID, params, mpk, pk).$$

2.9 Self-Certified Signatures

In order to avoid certificate management problem and key escrow problem, the notion of self-certified public keys was introduced by Girault [Gir91]. Similarly, self-certified signature schemes require a trusted third party. However, a third party has no ability of obtaining user's private keys. Briefly, a user randomly chooses a pair of public and private keys using a set of common parameters published by a trusted third party. Then, a user sends a public key and proves to a third party that he/she knows the related private key. If it is valid, a trusted third party issues a witness which is a signature of a user's public key and identity. In a self-certified signature scheme, a witness can be used to extract a user's public key by anyone. Therefore, it is unnecessary to use additional certificates and the private key is unknown to a third party.

Definition 2.16 *A self-certified signature consists of five algorithms:* Setup*,* KeyGen*,* WitReg*,* Sign *and* Verify*.*

- Setup*: A PPT algorithm run by the TTP that takes as input a security parameter k_1, outputs a pair of master secret and public keys (msk, mpk) and*

public system parameters params,

$$(msk, mpk, params) \leftarrow \mathsf{Setup}(k_1).$$

- KeyGen*: A PPT algorithm run by a user that takes as input a security parameter k_2 and params, outputs a pair of private and public keys (sk, pk),*

$$(sk, pk) \leftarrow \mathsf{KeyGen}(k_2, params).$$

- WitReg*: A PPT algorithm run by the TTP that takes as input an identity ID, a public key pk, a proof of knowledge of private key ppk, $params$ and a master public key, outputs a witness W,*

$$W \leftarrow \mathsf{WitReg}(ID, pk, ppk, params, mpk).$$

- Sign*: A PPT algorithm run by a user that takes as input a message m, pubic system parameters $params$ and a private key sk, outputs a self-certified signature σ, where*

$$\sigma \leftarrow \mathsf{Sign}(m, params, sk).$$

- Verify*: A deterministic algorithm run by a user that takes as input a message m, a signature σ, $params$, a master public key mpk, an identity ID and a witness W, outputs true if the signature is valid, otherwise outputs false,*

$$\{true, false\} \leftarrow \mathsf{Verify}(m, \sigma, params, mpk, ID, W).$$

2.10 Batch Verification

Fiat [Fia90, Fia97] introduced an idea called *batch verification* to improve the efficiency of a signature verification. Originally, his idea is for the RSA verification. In a batch verification scheme, a user who simultaneously verifies a batch of signatures accepts all signed messages if the result of verification turns out to be true. Several techniques for batch verification were described by Ballare, Garay and Rabin [BGR98]. The technique which is referred to as the *small exponents test* is a fast probabilistic test. However, a general attack was found by Boyd and Pavlovski [BP00] in 2000. As mentioned in their solution, to prevent the attack, a test algorithm should satisfy two requirements: 1) The order p of a group $\mathbb{G}$ is prime; 2) Check all chosen elements $\lambda_i \in \mathbb{G}$, where $i \in \{1, \ldots, n\}$. The repaired definition of small exponents test in a cyclic group is as follows:

Definition 2.17 *Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p. g is a generator of a group $\mathbb{G}$. Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i \in \mathbb{Z}_p$ and $y_i \in \mathbb{G}$. l is a security parameter, where $l < |\mathbb{Z}_p|$.*

1. *Check if for all $i \in \{1, \ldots, n\} : y_i = g^{x_i}$.*
2. *Randomly choose a set $\{r_1, \ldots, r_n\}$, where $r_i \in_R \{0,1\}^l$, $i \in \{1, \ldots, n\}$.*
3. *Compute $x = \sum_{i=1}^{n} x_i r_i \mod p$, and $y = \prod_{i=1}^{n} y^{r_i}$.*
4. *Accept all signatures if $y = g^x$, otherwise, reject.*

Chapter 3

Efficient Self-Certified Signatures with Batch Verification

This chapter describes a new construction of self-certified signatures. It is efficient in signing phase with pre-computation. The proposed scheme can be applied to batch verification.

3.1 Introduction

Digital signature is an important primitive in modern cryptography. A valid digital signature can be seen as a receipt of a message from the particular sender and can be applied to many security services such as authentication and non-repudiation. Signature verification relies on public key or signature verification key; therefore, proving the relationship between a public key and its owner is essential for security of signatures. In practice, it relies on the Public Key Infrastructure (PKI). That is, Certificate authority (CA) as a part of PKI issues public key certificates to its users. Nevertheless, PKI might not be desirable. Often, a signature has to be distributed along with its public key certificate. Prior to the signature verification, a signature receiver needs to check the validity of the corresponding certificate and store the certificate for later communications. Certificate distribution, verification and storage add additional cost to communication, computation and storage.

The notion of identity-based signature (IBS) was introduced by Shamir in 1984 [Sha85]. Problems of certificate verification and management are solved by using the signer's identity as his public key. This idea has been applied to various signature schemes, including several multi-user signatures (e.g., [ZK02, GR06]). An identity-based signature scheme secure in the standard model was proposed by Paterson and Schuldt [PS06]. In identity-based signatures, a user's private key is generated by a

trusted authority (TA), as a private key generator (PKG). As a drawback of identity-based systems, PKG can sign a message on behalf of any user. It is referred to as the so-called *key escrow* problem. The problem may be avoided by sharing master secret key among several authorized parties [YCK04], but a potential collusion of the authorities could still be a problem. Some other efforts are also presented in [YSM10, Cho09].

To fill the gap between the PKI based and identity-based signatures, Girault [Gir91] introduced the notion of *Self-certified Public Keys*, where certificate verification and management are not required and the key escrow problem can be eliminated. The idea is that the certificate is replaced by a *witness* and the public key is embedded in it. Anyone who holds a witness along with an attributive identity can recover a correct public key for signature verification. The amount of communication, computation and storage are also reduced. Unlike identity-based schemes, the trusted third party (TTP) cannot extract user's private key. The scheme captures a strong security (level-3) defined by Girault [Gir91]. Notice that IBS only reaches level-1 security.

Saeednia [Sae03] found a problem in the Girault's scheme, namely, a malicious TTP can compromise user private key by using a specific composite modular of RSA. Roughly speaking, the TTP chooses two "small" prime numbers to compute the RSA modulus n and it is helpful to solve the discrete logarithm problem. We refer the readers to [Sae03]. Zhou, Cao and Lu [ZCL04] prevented this attack by utilizing different user chosen modular, whereas the size of signature is increased and the public key recovery must be separated from the signature verification. Self-certified public key generation protocol based on discrete logarithm was also proposed in [PH97].

In this chapter, we propose an efficient and novel self-certified signature (SCS) scheme, which achieves the level-3 security as defined by Girault. The scheme is based on the discrete logarithm rather than RSA. Hence, the private key exposure problem has been resolved. In our scheme, there is no need to separate a certificate and a public key. Instead, we embed user's public key in a witness, which can be seen as a *lightweight* certificate. The public key can be implicitly verified in the signature verification, while anyone who has the user identity and the witness can explicitly extract the public key. We present both cases in our scheme.

The efficiency of a signature scheme is normally evaluated by two aspects: signing efficiency and verification efficiency. In the signing phase, our self-certified signature

scheme only requires one exponent and two multiplication computations with no pairing calculation. We also show that our SCS scheme can be made more efficient by utilizing the idea of pre-computation so that only one multiplication computation is needed. In the verification phase, our scheme requires two pairing computations. However, it is reduced to one pairing computation when the signer's public key has been recovered explicitly. Additionally, we show that our scheme is especially suitable for verifying large number of signatures by batch verification. The result shows that our scheme achieves a constant number of pairing computations in multi-signer setting. We prove that our scheme is secure in the random oracle model.

Related Work. The notion of *certificateless public key cryptography* (CL-PKC) was introduced by Al-Riyami and Paterson [ARP03] in 2003. The idea is similar to self-certified public keys, since the signer is implicitly certified in signature verification and no certificate involved the scheme. Similar with TTP in SCS scheme, an authority called *Key Generation Centre* (KGC) who generates partial private keys for users. An efficient certificateless signature scheme was proposed by Choi, Park, Hwang and Lee [CPHL07] (or CPHL for short). An efficient pairing-free security mediated certificateless signature scheme was proposed by Yap, Chow, Heng and Goi [YCHG07]. While the signing algorithm is an interactive protocol between a signer and an online semi-trusted server. The signature generation needs the help of a third party. Gentry [Gen03] introduced *Certificate-Based Cryptography* (CBC) as another paradigm to remove certificate and solve private key escrow problem. Indeed, the CL-PKC and CBC schemes can easily transfer from the one to the other [WMSH09]. Liu, Baek, Susilo and Zhou [LBSZ08] (or LBSZ for short) proposed a certificate-based signature scheme without pairing computations in random oracle.

The main difference between self-certified signatures and certificateless or certificate-based signatures is the key recoverable property. In self-certified signatures, the user's public key is computable by anyone who has his witness along with a set of public parameters. Once the user's public key has been recovered, the TTP's public key is no longer required. It implies that the cost of key certification and calculation is only needed at the initial stage of a communication as traditional signature schemes. If we treat the witness as a "public key", then it can be used along with the TTP's public key to verify a signature. In certificateless signatures and certificate-based signatures, on the other hand, the signature verification always needs the KGC's public key and the user's public key is uncomputable except the

Table 3.1: Comparison of Existing Schemes. (P: pairing computation; E: exponentiation; M: multiplication; Size: number of elements; SCS-1: our basic scheme; SCS-2: public key is already recovered by a verifier.)

	Signing	Verification	Signature Size	Public Key Size
CPHL	2E	1P+2E+1M	2	1
LBSZ	1E+2M	3E+4M	3	3
Our SCS-1	1E+2M	2P+3E+1M	2	1
Our SCS-2	1E+2M	1P+2E	2	1

KGC.

We compare some efficient schemes in CL-PKC, CBC and SCS models in Table 1. Our proposed scheme has two cases in signature verification. The SCS-1 is used when a user's public key is unknown and the SCS-2 is carried out in the case of the public key has been computed. The result shows that the verification cost can be reduced as in the second case. Both Signature Size and Public Key Size columns indicate the number of required group elements.

Organization of This Chapter. The rest of this chapter is organized as follows. The definition of our scheme and complexity assumptions are given in Section 3.2. A formal security model of our scheme is defined in Section 3.3. Our proposed scheme along with a formal security proof of our scheme is given in Section 3.4. Further discussions on pre-computation and batch verification are presented in Section 3.5 and 3.6, respectively. Finally, Section 3.7 concludes the chapter.

3.2 Definitions

Digital signature schemes are basically consisted of three algorithms: key generation (**KeyGen**), signing algorithm (**Sign**) and verification algorithm (**Verify**). Besides the basic algorithms, a self-certified signature scheme has two additional algorithms: the system setup algorithm (**Setup**) for generating system parameters and the witness registration algorithm (**WitReg**) for registering a user. The five algorithms in SCS are defined as follows:

- **Setup**(k_1): is a PPT algorithm run by a Trusted Third Party (TTP) that takes as input a security parameter k_1, outputs the public system parameters *param* and a master secret key msk.

- **KeyGen**(k_2): is a PPT algorithm run by a user that takes as input a security parameter k_2, outputs a pair of public and private keys (pk, sk).
- **WitReg**(ID, pk, v): is a PPT algorithm run by the TTP that takes as input a user's identity ID, public key pk and the proof of the knowledge of private key v, outputs a witness W if the proof v is valid, otherwise rejects.
- **Sign**(m, sk): is a PPT algorithm that takes as input a message m, private key sk, outputs a signatures $\sigma = (u, t)$.
- **Verify**(m, σ, ID, W): is a deterministic algorithm that takes as input a message m, a signature σ, user's identity ID and the witness W, outputs *true* if it is valid, otherwise outputs *false*.

3.3 Security Models

Goldwasser, Micali and Rivest [GMR88] introduced the strongest security notion of digital signature schemes: existential unforgeability against adaptive chosen-message attacks (EUF-CMA). A self-certified signature scheme needs to satisfy EUF-CMA as normal signature schemes. However, there are some differences according to the use of self-certified public keys. Girault [Gir91] defined the security of self-certified public keys as three levels: 1) The TTP knows a user's private key; 2) The attacker cannot know a user's private key, but it can forge a false witness without being detected by users; 3) Anyone cannot know a user's private key and cannot forge a witness without being detected. Hence, the identity-based signature schemes are only reach the level 1. A self-certified signature scheme should satisfy the level 3. Following this notion, we define a security model of self-certified signature schemes. There are two cases in our security model and the SCS scheme is EUF-CMA iff it is secure in both cases.

- *Type I adversary* $(\mathcal{A}_I)$: plays as a malicious user who does not get a valid witness from the TTP. The adversary tries to forge a witness that cannot be detected in the verification phase.
- *Type II adversary* $(\mathcal{A}_{II})$: is considered as a corrupted TTP who tries to reveal the user's private key.

The security of self-certified signatures is defined by two games.

Game 1: This is a game defined as *Type I attack*. The challenger runs **Setup** and gives public parameters to $\mathcal{A}_I$. $\mathcal{A}_I$ has an ability to access user private keys, but the master secret key is unknown. The adversary makes **Corruption**, **WitReg**, **Sign** queries and outputs a forgery.

- **Setup**: The challenger $\mathcal{C}$ runs the algorithm **Setup** to generate public parameters *param* and returns to $\mathcal{A}_I$.
- **Queries**: $\mathcal{A}_I$ has the ability to adaptively submit three types of query defined as follows.
  - **Corruption Query**: On $\mathcal{A}_I$'s query ID, $\mathcal{C}$ returns the corresponding private key. $\mathcal{A}_I$ can make this query at most q_1 times.
  - **WitReg Query**: On $\mathcal{A}_I$'s query (ID, pk, v), $\mathcal{C}$ runs the algorithm **WitReg** and returns a valid witness W. $\mathcal{A}_I$ can make this query at most q_2 times.
  - **Sign Query**: On $\mathcal{A}_I$'s query (m, ID), $\mathcal{C}$ runs the algorithm **Sign** and returns a signature σ of message m. $\mathcal{A}_I$ can make this query at most q_3 times.
- **Forgery**: $\mathcal{A}_I$ outputs a signature $\sigma^* = (u^*, t^*)$ of a message m^* that the pair (m^*, ID^*) is not queried in **Sign Query** and W^* is not an output of **WitReg Query**. $\mathcal{A}_I$ wins the game if the **Verify**$(m^*, \sigma^*, ID^*, W^*)$ =*true*. The advantage of $\mathcal{A}_I$ is defined as

$$Adv_{\mathcal{A}_I} = \Pr[\mathcal{A}_I \ wins].$$

Definition 3.1 *A self-certified signature scheme is $(t, q_1, q_2, q_3, \epsilon)$-secure against adaptively chosen message Type I attack, if there is no $\mathcal{A}_I$ who wins Game 1 in polynomial time t with advantage at least ϵ after q_1, q_2, q_3 queries.*

Game 2: This is a game defined as *Type $\mathcal{A}_{II}$ attack*. The challenger runs **Setup** and gives public parameters to $\mathcal{A}_{II}$. Due to $\mathcal{A}_{II}$ is considered as a dishonest TTP, a master secret key is also returned, but $\mathcal{A}_{II}$ has no ability to access user private key. Then the adversary makes **Public-Key**, **Sign** queries and outputs a forgery.

- **Setup**: The challenger runs the algorithm **Setup**, outputs public parameters *param* and a master secret key msk. $\mathcal{C}$ gives *param* and msk to the adversary.

- **Public-Key Query**: On $\mathcal{A}_{II}$'s query ID, the challenger $\mathcal{C}$ runs the algorithm **KeyGen** and returns a public key. $\mathcal{A}_{II}$ can make this query at most q_1 times.
- **Sign Query**: On $\mathcal{A}_{II}$'s query (m, ID), $\mathcal{C}$ runs the algorithm **Sign** and returns a signature σ of a message m. $\mathcal{A}_{II}$ can make this query at most q_2 times.
- **Forgery**: $\mathcal{A}_{II}$ outputs a signature $\sigma^* = (u^*, t^*)$ of a message m^* that the pair (m^*, ID^*) is not queried in **Sign Query**. $\mathcal{A}_{II}$ wins the game if the **Verify**$(m^*, \sigma^*, ID^*, W^*)$=*true*. The advantage of $\mathcal{A}_{II}$ is defined as

$$Adv_{\mathcal{A}_{II}} = \Pr[\mathcal{A}_{II}\ wins].$$

Definition 3.2 *A self-certified signature scheme is (t, q_1, q_2, ϵ)-secure against adaptively chosen message Type II attack, if there is no $\mathcal{A}_{II}$ who wins Game 2 in polynomial time t with advantage at least ϵ after q_1, q_2 queries.*

3.4 The Proposed Scheme

In PKI based schemes, a certificate can be seen as a part of a signature when the two parties initiate a communication. The verification of a certificate is required prior to the signature verification. For stable partners who communicate frequently, the cost of certificate transmission and verification are negligible. However, in most cases, the participants barely know each other personally, and hence, the verification process becomes essential. We present a novel and efficient self-certified signature scheme that the cost of computations, transmission and storage are all reduced.

3.4.1 Construction

Setup: Select a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where the order of group $\mathbb{G}$ and $\mathbb{G}_T$ are the same prime p. Let g be a generator of $\mathbb{G}$. The TTP then chooses two collision-resistant cryptographic hash functions that $h_1 : \{0,1\}^* \to \mathbb{G}$, $h_2 : \{0,1\}^* \to \mathbb{Z}_p^*$. Randomly select a number $\alpha \in_R \mathbb{Z}_p^*$, set $msk = \alpha$ and the master public key $mpk = g^\alpha$. The public parameters are $(\mathbb{G}, \mathbb{G}_T, g, p, e, h_1, h_2, mpk)$.
KeyGen: Randomly chooses $x \in_R \mathbb{Z}_p^*$ and computes $e(g,g)^x$. Sets the public and private keys as $(pk, sk) = (e(g,g)^x, x)$.
WitReg: A user interact with a TTP in this algorithm as follows.

- The user computes a proof of knowledge of private key $v = g^{\alpha x}$, where x is the user private key, and sends (ID, pk, v) to TTP.
- TTP verifies the equation $e(v, g) \stackrel{?}{=} pk^{\alpha}$, if it holds, then generates a witness

$$W = (v^{\frac{1}{\alpha}} h_1(ID))^{\frac{1}{\alpha}}.$$

- The user accepts the witness if the following equations holds:

$$\begin{aligned} & e(W, mpk)e(h_1(ID)^{-1}, g) \\ & = e(v^{\frac{1}{\alpha}} h_1(ID), g)e(h_1(ID)^{-1}, g) \\ & = pk. \end{aligned} \tag{3.1}$$

Sign: To sign a message $m \in \{0,1\}^*$, the signer randomly selects $r \in_R \mathbb{Z}_p^*$ and computes

$$\begin{aligned} \sigma &= (g^r, \frac{1 - rh_2(m||g^r)}{x}) \\ &= (u, t). \end{aligned}$$

Verify: On input a signature $\sigma = (u, t)$ on a message m under a witness W of the identity ID, the verifier checks whether

$$e(W^t, mpk)e(u^{h_2(m||u)} h_1(ID)^{-t}, g) \stackrel{?}{=} e(g, g) \tag{3.2}$$

or

$$e(g, g)^{xt} e(u^{h_2(m||u)}, g) \stackrel{?}{=} e(g, g). \tag{3.3}$$

Outputs *true* if the equation holds, otherwise outputs *false*. The equation (3.3) is to utilize once the user public key was recovered as in (3.1).

Correctness: Our self-certified signature scheme is correct as shown in following:

$$\begin{aligned} & e(W^t, mpk)e(u^{h_2(m||u)} h_1(ID)^{-t}, g) \\ & = e((v^{\frac{1}{\alpha}} h_1(ID))^t, g)e(u^{h_2(m||u)} h_1(ID)^{-t}, g) \\ & = e(g^{xt} g^{rh_2(m||u)}, g) \\ & = e(g, g). \end{aligned}$$

3.4.2 Security Analysis

A self-certified signature is unforgeable if it is secure against two types of attacks defined in Section 3.3. We show that our signature scheme is secure under the strongest security notion for signature schemes (EUF-CMA).

Theorem 3.1 *Our SCS scheme is $(t, q_{h_1}, q_2, q_3, \epsilon)$-secure against existential forgery under Type I chosen message attack, q_{h_1} is the number of queries on h_1 hash function, assuming that the (k+1)-exponent assumption is (t', ϵ')-hard, where,*

$$\epsilon' \geq \frac{1}{q_2} \cdot (1 - \frac{1}{q_2 + 1})^{q_2+1} \cdot \epsilon, \qquad t' = t + O(q_{h_1} + q_2 + q_3).$$

Proof: Suppose a *Type I adversary* $\mathcal{A}_I$ who can $(t, q_1, q_2, q_3, \epsilon)$-break our SCS scheme. We can construct an algorithm $\mathcal{B}$ run by the challenger to use $\mathcal{A}_I$ to solve the $(k+1)$-exponent problem. The algorithm $\mathcal{B}$ is given the $(k+1)$-EP instance $(g, g^a, g^{a^2}, g^{a^3})$, where $k = 3$, and the goal is to output g^{a^4}. $\mathcal{B}$ interacts with $\mathcal{A}_I$ in game 1 as follows.

Setup: $\mathcal{B}$ sets g^a as the generator of a group $\mathbb{G}$ and the master public key $mpk = g$. Let the master secret key $msk = a^{-1}$, which is unknown to $\mathcal{B}$. $\mathcal{B}$ maintains four lists $L_{h1} = \{< ID, b, coin \in \{0, 1\} >\}$, $L_{h2} = \{< M, c >\}$, $L_c = \{< ID, sk >\}$ and $L_w = \{< ID, pk, v, W >\}$, which are initially empty.

h_1 **Query**: $\mathcal{A}_I$ issues an h_1 query on input ID_i at most q_{h_1} times, where $1 \leq i \leq q_{h_1}$. $\mathcal{B}$ outputs $h_1(ID_i)$ if ID_i is in the list L_{h1}. Otherwise, $\mathcal{B}$ tosses a coin with the probability $\Pr[coin = 1] = \xi$ ($\Pr[coin = 0] = 1 - \xi$), selects $b_i \in_R \mathbb{Z}_p^*$ and answers the query as follows.

$$\begin{cases} coin_i = 0: \ h_1(ID_i) = g^{ab_i}, \\ \\ coin_i = 1: \ h_1(ID_i) = g^{a^3 b_i}, \end{cases}$$

$\mathcal{B}$ outputs $h_1(ID_i)$ and adds $< ID_i, b_i, coin_i >$ in the list L_{h1}

h_2 **Query**: $\mathcal{A}_I$ issues an h_2 query on input string M_i at most q_{h_2} times, where $1 \leq i \leq q_{h_2}$. $\mathcal{B}$ outputs $h_2(M_i)$ if M_i is in the list L_{h2}. Otherwise, $\mathcal{B}$ randomly selects $c_i \in_R \mathbb{Z}_p^*$ and sets $h_2(M_i) = c_i$. Then, $\mathcal{B}$ outputs $h_2(M_i)$ and adds $< M_i, c_i >$ into the list L_{h2}.

Corruption Query: $\mathcal{A}_I$ issues a corruption query on input identity ID_i, where $1 \leq i \leq q_1$. $\mathcal{B}$ outputs sk_i if ID_i is in the list L_c. Otherwise, $\mathcal{B}$ outputs a random choice $sk_i \in_R \mathbb{Z}_p^*$ and adds $< ID_i, sk_i >$ in the list L_c.

WitReg Query: $\mathcal{A}_I$ issues a witness query on input (ID_i, pk_i, v_i), where $1 \leq i \leq q_2$. $\mathcal{B}$ outputs a witness W_i if ID_i is in the list L_w. Otherwise, $\mathcal{B}$ retrieves the private key sk_i and b_i in L_c and L_{h1}, respectively. If $coin_i = 0$, $\mathcal{B}$ sets and outputs witness W_i as

$$W_i = g^{a^2(sk_i+b_i)}.$$

$\mathcal{B}$ adds $< ID_i, pk_i, v_i, W_i >$ into the list L_w. If $coin_i = 1$, $\mathcal{B}$ outputs FAIL and aborts the simulation.

Sign Query: $\mathcal{A}_I$ issues a signing query on input (m_i, ID_i), where $1 \leq i \leq q_3$. $\mathcal{B}$ retrieves the private key sk_i from the list L_c. If it exists, runs the algorithm **Sign** and outputs a signature σ on message m_i. Otherwise, $\mathcal{B}$ runs **Corruption Query** first, then generates a signature as before.

Forgery: Eventually, $\mathcal{A}_I$ outputs a forgery $\sigma^* = (u^*, t^*)$ on message m^* under the witness W^* of identity ID^*. $\mathcal{A}_I$ wins the game if **Verify**$(m^*, \sigma^*, ID^*, W^*)$ outputs *true*, the pair (m^*, ID^*) does not be an input of **Sign Query** and W^* is not an output of **WitReg Query**. We assume that sk^* and b^* are in L_c and L_{h1}, respectively. $\mathcal{B}$ computes a solution of $(k+1)$-exponent problem $(k = 3)$ as follows

$$g^{a^4} = (W^* g^{-a^2 sk^*})^{\frac{1}{b^*}}.$$

Probability: The simulator $\mathcal{B}$ outputs FAIL only if $coin_i = 1$ when the adversary queries a witness. Hence, the challenger can solve the $(k+1)$-exponent problem in condition of the simulation is success and the forgery witness is related to the index i. The probability is $\epsilon' \geq \frac{1}{q_2} \cdot (1 - \frac{1}{q_2+1})^{q_2+1} \cdot \epsilon$ and the reduction process is as [BLS04b]. The time of an exponentiation in each query is denoted as $O(1)$, so the simulation time is $t' = t + O(q_{h_1} + q_2 + q_3)$. □

Theorem 3.2 *Our SCS scheme is (t, q_1, q_2, ϵ)-secure against existential forgery under Type II chosen message attack, assuming that the DL assumption is (t', ϵ')-hard, where*

$$\epsilon' = \epsilon, \qquad t' \geq t + O(q_{h_1} + q_1 + 2q_2).$$

Proof: Suppose a *Type II adversary* $\mathcal{A}_{II}$ who can (t, q_1, q_2, ϵ)-break our SCS scheme. We can construct an algorithm $\mathcal{B}$ run by the challenger to use $\mathcal{A}_{II}$ to solve the DL problem. The algorithm $\mathcal{B}$ is given the DL instance (g, g^a), and the goal is to output a. $\mathcal{B}$ interacts with $\mathcal{A}_{II}$ in game 2 as follows.

Setup: $\mathcal{B}$ sets g as the generator of a group $\mathbb{G}$ and the master public key $mpk = g^\alpha$. Let the master secrete key $msk = \alpha$ and give it to $\mathcal{A}_{II}$. $\mathcal{B}$ maintains three list

$L_{h1} = \{< ID, b >\}$, $L_{h2} = \{< M, c >\}$ and $L_{pk} = \{< ID, pk, s >\}$, which are initially empty.

h_1 **Query**: $\mathcal{A}_{II}$ issues an h_1 query on input ID_i at most q_{h_1} times, where $1 \leq i \leq q_{h_1}$. $\mathcal{B}$ outputs $h_1(ID_i)$ if ID_i is in the list L_{h1}. Otherwise, $\mathcal{B}$ randomly chooses $b_i \in \mathbb{Z}_p^*$ and sets $h_1(ID_i) = g^{b_i}$. Then, $\mathcal{B}$ outputs $h_1(ID_i)$ and adds $< ID_i, b_i >$ in the list L_{h1}.

h_2 **Query**: $\mathcal{A}_{II}$ issues an h_2 query on input string M_i at most q_{h_2} times, where $1 \leq i \leq q_{h_2}$. $\mathcal{B}$ answers the query as h_2 **Query** in game 1 and adds $< M_i, c_i >$ in the list L_{h2}.

Public-key Query: $\mathcal{A}_{II}$ issues a public-key query on input ID_i, where $1 \leq i \leq q_1$. $\mathcal{B}$ outputs pk_i if ID_i is in the list L_{pk}. Otherwise, $\mathcal{B}$ randomly chooses $s_i \in_R \mathbb{Z}_p^*$ and computes public key

$$pk_i = e(g^a, g)^{s_i}.$$

$\mathcal{B}$ then outputs pk_i and adds $< ID_i, pk_i, s_i >$ into the list L_{pk}.

Sign Query: $\mathcal{A}_{II}$ issues a signing query on input (m_i, ID_i), where $1 \leq i \leq q_2$. $\mathcal{B}$ answers queries as follows:

- If ID_i is not in L_{pk}, $\mathcal{B}$ runs **Public-Key Query**.
- Otherwise, $\mathcal{B}$ randomly selects $c_i, r_i \in_R \mathbb{Z}_p^*$ and computes

$$u_i = g^{\frac{1}{c_i}}(g^a)^{-s_i r_i}, \quad t_i = c_i r_i.$$

  Let $M_i = m_i || u_i$ and $h_2(M_i) = c_i$, $\mathcal{B}$ adds $< M_i, c_i >$ into the list L_{h_2} and outputs the signature $\sigma_i = (u_i, t_i)$.

Forgery: Eventually, $\mathcal{A}_{II}$ outputs a forgery $\sigma^* = (u^*, t^*)$ on message m^* under a witness W^* of the identity ID^*. $\mathcal{A}_{II}$ wins the game if **Verify**$(m^*, \sigma^*, ID^*, W^*)$ outputs *true* and the pair (m^*, ID^*) is never queried to the **Sign Query**. Then, $\mathcal{B}$ can run the same random tape and a different h_2 to output another valid signature $\sigma^{*\prime} = (u^{*\prime}, t^{*\prime})$. The outputs of two h_2 hash functions are respectively c^* and $c^{*\prime}$, where $c^* \neq c^{*\prime}$. We assume that s^* is in the list L_{pk}. $\mathcal{B}$ can compute

$$\begin{cases} 1 - r^* c^* = a s^* t^*, \\ \\ 1 - r^* c^{*\prime} = a s^* t^{*\prime}, \end{cases} \qquad a = \frac{r^*(c^{*\prime} - c^*)}{s^*(t^* - t^{*\prime})} \quad ,$$

as a solution of DL problem.

Probability: The simulator $\mathcal{B}$ does not outputs FAIL in any queries. The challenger can solve the DL problem in condition of the successful simulation. Hence, the probability is $\epsilon' = \epsilon$. The time consuming of an exponentiation is considered as $O(1)$. Therefore, the simulation time is $t' = t + O(q_{h_1} + q_1 + 2q_2)$. □

3.5 Self-Certified Signatures with Precomputations

Even, Goldreich and Micali introduced the notion of online/offline signatures [EGM90] to improve the signature generation efficiency. Their main idea is to split the signature generation into two stages, namely offline stage and online stage. Most heavy computations are carried out in the offline stage prior to the availability of the message. Once the message is received, the algorithm can output a signature quickly by conducting the online stage. They proposed a method which converts any signature schemes into an online/offline signature scheme. However, it is impractical. Subsequently, Shamir and Tauman presented an efficient "*hash-sign-switch*" paradigm [ST01]. The signature size is largely reduced while the efficiency is maintained.

Our scheme provides pre-computation in the signing stage as some other schemes mentioned in [ST01]. It is easy to partition our scheme into two parts: offline stage and online stage. In the offline stage, the signer picks a random choice r', where $r' \in_R \mathbb{Z}_p^*$. Then he/she computes $u' = g^{r'}$ and $t' = \frac{r'}{x}$. The pair (u', t') should be securely stored. In the online stage, the signer retrieves a pair (u', t'), and computes $u = u', t = x^{-1} - t'h_2(m||u')$ as a signature on the message m. Hence, in the online signature operations, it only requires a modular multiplication and a subtraction, provided that the signer stores the inverse of his private key x^{-1}. In addition, the length of our self-certified signature scheme is as short as [ST01].

3.6 Batch Verification

The notion of batch verification was introduced by Fiat in 1989 [Fia90]. Generally, the motivation of batch verification is to improve the verification efficiency when verifying large number of signatures. According to the three paradigms of batch verification scheme proposed in [BGR98], we apply the *Small Exponent Test* in this chapter. The length l of the exponent is a security parameter that depends on the security requirement in practice. Batch verification for single-signer and multi-signer

settings are both provided in this section.

3.6.1 Single-Signer Batch Verification

In the single-signer setting, there is no need to implicitly verify signer public keys in all signatures, since all public keys are the same. Therefore, we assume that the signer's public key has been recovered and the equation (3) is used in the verification. Nevertheless, the equation (2) can be used in a similar way if the public key is not computed.

Let $(\mathbb{G}, \mathbb{G}_T, g, p, e, h_1, h_2, mpk)$ be public parameters and $k = |\mathbb{G}| = |\mathbb{G}_T|$. Given a set of signatures $S = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, where $\sigma_i = (u_i, t_i)$, on messages $M = \{m_1, m_2, \ldots, m_n\}$ from the same singer in which $pk = e(g, g)^x$. The verifier checks S as follows.

- If $u_i \notin \mathbb{G}$, where $i = 1, 2, \ldots, n$, rejects all signatures and outputs *false*.
- Otherwise, randomly selects l-bits elements $(\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathbb{Z}_p^n$, where $l < k$, and computes:

$$T = \lambda_1 t_1 + \lambda_2 t_2 + \ldots + \lambda_n t_n = \sum_{i=1}^{n} \lambda_i t_i \ ,$$

$$U = u_1^{\lambda_1 h_2(m_1||u_1)} \cdot u_2^{\lambda_2 h_2(m_2||u_2)} \ldots u_i^{\lambda_i h_2(m_i||u_i)} = \prod_{i=1}^{n} u_i^{\lambda_i h_2(m_i||u_i)} \ ,$$

$$C = \lambda_1 + \lambda_2 + \ldots + \lambda_i = \sum_{i=1}^{n} \lambda_i.$$

Accepts all signatures and outputs *true* if the equation holds

$$e(g, g)^{xT} e(U, g) = e(g, g)^C.$$

Correctness

$$\begin{aligned}
&e(g, g)^{xT} e(U, g) \\
&= e(g, g)^{x \sum_{i=1}^{n} \lambda_i t_i} e(g, g)^{\sum_{i=1}^{n} r_i \lambda_i h_2(m_i||u_i)} \\
&= e(\prod_{i=1}^{n} g^{\lambda_i - r_i \lambda_i h_2(m_i||u_i)}, g) e(\prod_{i=1}^{n} g^{r_i \lambda_i h_2(m_i||u_i)}, g) \\
&= e(g, g)^C.
\end{aligned}$$

Let A be a modular addition in $\mathbb{Z}_p^*$ and Pa is a pairing calculation. Mul_s is a modular multiplication in group s. An l-bits exponentiation in group s is denoted as

$Ex_s(l)$ and a test of a group member is Gt. Computational cost of has functions in both types of verification are ignored since they are the same. The cost of native verification and batch verification on n signatures in single-signer setting are respectively,

$$nEx_{\mathbb{G}}(k) + nEx_{\mathbb{G}_T}(k) + nPa + nMul_{\mathbb{G}_T}$$

and

$$nGt + 2nMul_{\mathbb{Z}_p^*} + 2(n-1)A + nEx_{\mathbb{G}}(k) + 1Ex_{\mathbb{G}_T}(k) + 1Pa + 1Ex_{\mathbb{G}_T}(l) + (n-1)Mul_{\mathbb{G}} + 1Mul_{\mathbb{G}_T}.$$

Theorem 3.3 *The batch verification of our self-certified signature scheme in single-signer setting is secure, if there is no adversary with probability at least 2^{-l}, where l is the length of a small exponent.*

Proof: Suppose that an adversary outputs a forgery (M^*, S^*) accepted by batch verification under identity ID. We show that the probability of a valid forgery depends on the length l of a small exponent.

Without losing generality, we assume that the public key $pk = e(g,g)^x$ has been recovered from (1). A signature $\sigma_i^* = (u_i^*, t_i^*)$ can be considered as

$$\sigma_i^* = (g^{r_i}, \ \frac{1 - r_i h_2(m_i^* || g^{r_i}) + k_i}{x}),$$

where $r_i, k_i \in_R \mathbb{Z}_p^*$. If $k_i = 0$, the signature is valid. Otherwise, it is invalid. Then, we can compute that

$$T^* = \lambda_1 t_1^* + \lambda_2 t_2^* + \ldots + \lambda_n t_n^* = \sum_{i=1}^{n} \lambda_i t_i^*, \quad U^* = U, \quad C^* = C.$$

If the following equation holds

$$\begin{aligned}
&e(g,g)^{xT^*} e(U^*, g) \\
&= e(g^x, g)^{\sum_{i=1}^n \lambda_i t_i^*} e(g,g)^{\sum_{i=1}^n r_i \lambda_i h_2(m_i^* || u_i)} \\
&= e(\prod_{i=1}^{n} g^{\lambda_i - r_i \lambda_i h_2(m_i^* || u_i) + \lambda_i k_i}, g) e(\prod_{i=1}^{n} g^{r_i \lambda_i h_2(m_i^* || u_i)}, g) \\
&= e(g,g)^{C^*},
\end{aligned}$$

then $\sum_{i=1}^n \lambda_i k_i \equiv 0 \pmod p$. Assuming that at least one signature σ_j^* is invalid. It implies the adversary can find a k_j such that

$$\lambda_j \equiv -k_j^{-1} \sum_{i=1, i \neq j}^{n} \lambda_i k_i \pmod p, \ k_j \neq 0.$$

However, small exponents λ_i, where $i = 1, 2, \ldots, n$, are l-bits random choices selected by the verifier. Hence, the probability of an adversary break the batch verification is equal to the probability of the equation hold, where

$$\Pr\left[\lambda_j \equiv -k_j^{-1} \sum_{i=1, i\neq j}^{n} \lambda_i k_i \pmod p \,\middle|\, \sum_{i=1}^{n} \lambda_i k_i \equiv 0 \pmod p\right] \leq 2^{-l}.$$

□

3.6.2 Multi-Signer Batch Verification

Generally speaking, the batch verification in a single-signer setting is a special case of that in a multi-signer setting. The amount of pairing computations normally depend on the number of signers in the multi-signer batch verification. However, we show that our scheme only needs constant pairing computations.

Suppose that public keys have not been recovered in this case. Let $(\mathbb{G}, \mathbb{G}_T, g, p, e, h_1, h_2, mpk)$ be public parameters and $k = |\mathbb{G}| = |\mathbb{G}_T|$. Given a set of signatures $S = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, where $\sigma_i = (u_i, t_i)$, on messages $M = \{m_1, m_2, \ldots, m_n\}$ with witnesses $WT = \{W_1, W_2, \ldots, W_n\}$ under identity $I = \{ID_1, ID_2 \ldots, ID_n\}$, respectively. The verifier checks S as follows.

- If $u_i \notin \mathbb{G}$, where $i = 1, 2, \ldots, n$, rejects all signatures and outputs *false*.
- Otherwise, randomly selects l-bits elements $(\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathbb{Z}_p^n$, where $l < k$, and computes:

$$\begin{aligned}
T &= W_1^{\lambda_1 t_1} \cdot W_2^{\lambda_2 t_2} \ldots W_n^{\lambda_n t_n} = \prod_{i=1}^{n} W_i^{\lambda_i t_i}, \\
U &= (u_1^{h_2(m_1||u_1)} h_1(ID_1)^{-t_1})^{\lambda_1} \cdot (u_2^{h_2(m_2||u_2)} h_1(ID_2)^{-t_2})^{\lambda_2} \\
&\quad \ldots (u_i^{h_2(m_i||u_i)} h_1(ID_i)^{-t_i})^{\lambda_i} \\
&= \prod_{i=1}^{n} (u_i^{h_2(m_i||u_i)} h_1(ID_i)^{-t_i})^{\lambda_i}, \\
C &= \lambda_1 + \lambda_2 + \ldots + \lambda_i = \sum_{i=1}^{n} \lambda_i.
\end{aligned}$$

Accepts all signatures and outputs *true* if the equation holds

$$e(T, mpk)e(U, g) = e(g, g)^C.$$

Correctness

$$
\begin{aligned}
& e(T, mpk)e(U, g) \\
&= e(g^{x_i} h(ID_i), g)^{\sum_{i=1}^{n} \lambda_i t_i} e(\prod_{i=1}^{n} (u_i^{h_2(m_i||u_i)} h_1(ID_i)^{-t_i})^{\lambda_i}, g) \\
&= e(g, g)^{\sum_{i=1}^{n} x_i \lambda_i t_i} e(g, g)^{\sum_{i=1}^{n} r_i \lambda_i h_2(m_i||u_i)} \\
&= e(\prod_{i=1}^{n} g^{\lambda_i - r_i \lambda_i h_2(m_i||u_i)}, g) e(\prod_{i=1}^{n} g^{r_i \lambda_i h_2(m_i||u_i)}, g) \\
&= e(g, g)^C.
\end{aligned}
$$

The cost of the original verification and the batch verification on n signatures in a multi-signer setting are respectively,

$$3nEx_{\mathbb{G}}(k) + nMul_{\mathbb{G}} + 2nPa + nMul_{\mathbb{G}_T}$$

and

$$nGt + nMul_{\mathbb{Z}_p^*} + 3nEx_{\mathbb{G}}(k) + (3n-2)Mul_{\mathbb{G}} + nEx_{\mathbb{G}}(l) + (n-1)A + 2Pa + 1Mul_{\mathbb{G}_T} + 1Ex_{\mathbb{G}_T}(l).$$

Theorem 3.4 *The batch verification of our self-certified signature scheme in multi-signer setting is secure, if there is no adversary with probability at least 2^{-l}, where l is the length of a small exponent.*

Proof: Suppose that an adversary outputs a forgery (M^*, S^*) accepted by batch verification under a set of identities I^* with their corresponding witnesses WT^*. We show that the probability of a valid forgery depends on the length l of a small exponent.

Different from single-signer setting, we assume that users public keys have not been recovered. Let x_i, where $i = 1, \ldots, n$, be the ith user private key. A signature $\sigma_i^* = (u_i^*, t_i^*)$ can be considered as

$$\sigma_i^* = (g^{r_i}, \ \frac{1 - r_i h_2(m_i^* || g^{r_i}) + k_i}{x_i}),$$

where $r_i, k_i \in_R \mathbb{Z}_p^*$. If $k_i = 0$, the signature is valid. Otherwise, it is invalid. Then,

we can compute that

$$
\begin{aligned}
T^* &= W_1^{\lambda_1 t_1^*} \cdot W_2^{\lambda_2 t_2^*} \ldots W_n^{\lambda_n t_n^*} = \prod_{i=1}^{n} W_i^{\lambda_i t_i^*}, \\
U^* &= (u_1^{h_2(m_1||u_1)} h_1(ID_1)^{-t_1^*})^{\lambda_1} \cdot (u_2^{h_2(m_2||u_2)} h_1(ID_2)^{-t_2^*})^{\lambda_2} \\
&\quad \ldots (u_i^{h_2(m_i||u_i)} h_1(ID_i)^{-t_i^*})^{\lambda_i} \\
&= \prod_{i=1}^{n} (u_i^{h_2(m_i||u_i)} h_1(ID_i)^{-t_i^*})^{\lambda_i} \ , \\
C^* &= \lambda_1 + \lambda_2 + \ldots + \lambda_i = \sum_{i=1}^{n} \lambda_i.
\end{aligned}
$$

If the following equation holds

$$e(T^*, mpk)e(U^*, g) = e(g, g)^{C^*},$$

then $\sum_{i=1}^{n} \lambda_i k_i \equiv 0 \pmod{p}$. Assuming that at least one signature σ_j^* is invalid. It implies the adversary can find a k_j such that

$$\lambda_j \equiv -k_j^{-1} \sum_{i=1, i \neq j}^{n} \lambda_i k_i \pmod{p}, \ k_j \neq 0.$$

However, small exponents λ_i, where $i = 1, 2, \ldots, n$, are randomly picked by a user in a batch verification process. Therefore, the probability of an adversary break the batch verification is equal to the probability of the equation hold, where

$$\Pr\left[\lambda_j \equiv -k_j^{-1} \sum_{i=1, i \neq j}^{n} \lambda_i k_i \pmod{p} \,\middle|\, \sum_{i=1}^{n} \lambda_i k_i \equiv 0 \pmod{p}\right] \leq 2^{-l}.$$

□

3.7 Conclusion

In this chapter, we proposed an efficient and novel self-certified signature scheme. With pre-computation, our scheme requires only one modular multiplication for signature generation. Our scheme allows the batch verification in both single-singer and multi-signer settings. We showed that in the multi-signer setting, the verification of n signatures requires only two pairing computations regardless of the size of n. Our self-certified signature scheme was proven secure in the random oracle model.

Chapter 4

Self-Certified Ring Signatures

This chapter describes a novel notion of self-certified ring signatures along with its precise definition, security model and formal security proofs.

4.1 Introduction

The notion of ring signature was first proposed by Rivest, Shamir, and Tauman in 2001 [RST01]. Ring signatures are group-oriented digital signatures, which achieve signer anonymity as a major feature. It differs to a group signature as there is no anonymity revocation provided and no group setup stage. Ring signatures allow the user to sign on behalf of a group which is not predefined. Hence, any user can freely choose a set of users that include himself as a group and generate a ring signature. Verifiers believe that someone in the group signed the message, but cannot know who is the actual signer.

In the traditional public key infrastructure, a signer's public key is certified with a signature of the certificate authority. Although the public key certificates can be used to authenticate user public keys, they increase the computation and communication cost, especially for a large group. This issue has been a concern for the application of ring signatures (e.g., [ALSY07, CYH05]). Furthermore, the complexity of certificates management is also a drawback.

Shamir [Sha85] introduced the notion of identity-based signature (IBS) in 1984. The idea of the IBS is to eliminate the certificate verification and management problems by using the signer's identity as the public key. This idea was later applied to ring signatures (e.g., [ZK02, CLHY05, ALYW06]). The identity-based ring signatures (IBRS) exhibit a better applicability. Unfortunately, the main disadvantage of IBS and IBRS are user private keys are known by the trusted authority (TA) who generates the private keys for users. Therefore, TA can impersonate any user

to generate his/her signatures. This problem is referred to as *private key escrow*.

Girault [Gir91] introduced the concept of self-certified public keys as a solution to certificate management and private key escrow. The main feature of self-certified model (SCM) is that the user public key is computable through the witness and the public key of the trusted third party (TTP). As the public key is compressed into the witness, there is no need to verify the user's public key. Hence, compared with PKI, the self-certified model presents advantages about the amount of storage, communication and computation. SCM captures the level-3 security defined by Girault as the private key escrow problem is also eliminated, while a normal IBS only reaches the level-1.

However, Saeednia [Sae03] found a problem in the Girault's algorithm in that the TTP could still compromise the user private key via the selected composite modular of RSA (a product of two special primes that helps to solve discrete log problem). A potential solution given in [Sae03] is to increase the size of primes, but the size of witness will also be increased. Although it is not a problem in [ZCL04], the public key recovery must be separated from the signature verification and an additional computation is necessary. There exist some other work in self-certified system (e.g., [PH97, Sae97]).

In this chapter, we propose the first self-certified ring signature (SCRS) scheme, which reaches the level-3 security and fixes a problem in the original SCM. Our scheme captures all the features of self-certified model and ring signature schemes. Intuitively, in our scheme, the user's public key is embedded in a witness. The user only has to provide the identity and the witness. The verifier can compute the public key during the signature verification. We present the precise definition of self-certified ring signatures. We also present a concrete scheme where the ring is formed with the three-move model introduced in [AOS02].

Our scheme achieves the level-3 security defined by Girault [Gir91]. We provide a security model of self-certified ring signatures and prove that our SCRS is secure under this model. Different from the Girault's algorithm, our scheme does not rely on the RSA assumption. Therefore, the private key leakage problem is eliminated.

Related Work. Different but similar approach to the self-certificate cryptography is certificateless public key cryptography (CL-PKC), which was introduced by Al-Riyami and Paterson [ARP03] in 2003. It can also eliminate the key escrow and certificate management problems. In CL-PKC, the user gets a certificate from the

	Escrow Free	Secure Channel*	Public Key Recovery	Components of Private key
SCRS	Yes	No	Yes	1
CLRS	Yes	Yes	No	2
CBRS	Yes	No	No	2

Table 4.1: Properties of Existing Paradigms. (*A secure channel is required during the certificate/PPK transmission.)

key generation center (KGC). It is seen as a partial private key (PPK). Then, the actual private key is composed of the PPK and a secret value chosen by the user. The user then generates the public key and it can be verified without a certificate. In 2007, the notion of ring signatures was applied to CL-PKC by Zhang, Zhang and Wu [ZZW07b]. Independently, another certificateless ring signature scheme is proposed in [CY07].

Another related paradigm called *certificate-based cryptography* (CBC) [Gen03] was introduced to solve the same problems in PKI and IBS. In CBC, the private key and the corresponding public key are decided before getting a certificate. But, the same as in CL-PKC, the certificate is used to sign. CBC is very similar to CL-PKC and Wu *et al.* [WMSH09] presented a generic construction to convert a certificateless signature to a certificate-based signature. CBC is also applied to ring signatures. Au *et al.* [ALSY07] proposed the first certificate-based (linkable) ring signature. We compare some features of certificateless ring signature (CLRS), certificate-based ring signature (CBRS) and SCRS in Table 1. We can find their similarities and differences.

Organization of This Chapter. The rest of this chapter is organized as follows. In Section 4.2, we give the definitions of self-certified ring signature schemes, the security model, and some mathematical definitions. The concrete scheme was presented in Section 4.3. The security analysis of our scheme is given in Section 4.4. Finally, we conclude the chapter in Section 4.5.

4.2 Definitions

We give a definition of self-certified ring signatures. As introduced in [Gir91], our SCRS scheme has a special registration phase where each user gets a witness form the TTP. The SCRS security model is also given in this section.

4.2.1 Self-Certified Ring Signature

A self-certified ring signature is composed of the five algorithms: **SysSetup, KeyGen, WitReg, Sign** and **Verify**.

- **SysSetup**(λ_1)**:** Taking as input a security parameter λ_1, the algorithm returns public parameters Params and a master secret key msk.
- **KeyGen**(λ_2): Taking as input a security parameter λ_2, the algorithm returns the public and private keys (PK, SK).
- **WitReg**(ID, PK, Q): Taking as input the identity ID, public key PK and the proof of knowledge of private key Q, the algorithm returns the witness W if the Q is valid, otherwise rejects.
- **Sign**($m, \bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0,i\neq k}^{n-1}\{W_i\}, SK_k$): Taking as input a set of identities $\bigcup_{i=0}^{n-1}\{ID_i\}$, a set of witnesses $\bigcup_{i=0,i\neq k}^{n-1}\{W_i\}$, a private key SK_k, $k \in \{0, 1, \ldots, n-1\}$ and the message m, the algorithm outputs a self-certified ring signature σ.
- **Verify**($m, \sigma, \bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0}^{n-1}\{W_i\}$): Taking as input a signature σ, the message m and a set of identities $\bigcup_{i=0}^{n-1}\{ID_i\}$ with the corresponding witnesses $\bigcup_{i=0}^{n-1}\{W_i\}$, the algorithm returns *true* if it is valid, otherwise returns *false*.

4.2.2 SCRS Unforgeability

According to [Gir91], the security of a self-certified signature scheme is defined as three levels: 1) the TTP knows the user's private key; 2) the attacker cannot know user's private key, but it can forge a false witness without being detected by users; 3) anyone cannot know the user's private key and cannot generate a false witness without being detected. Level 3 is the highest security level of self-certified scheme.

We expand this notion and define a security model of self-certified ring signature schemes. In this model, the self-certified ring signature scheme must be existentially unforgeable against adaptive chosen-message attacks in two cases. For each type, a game is given to describe the related attack.

- *Type I attack***:** The *Type I attacker* is an illegal user who does not get a valid witness from the TTP. The attacker tries to forge a witness that cannot be detected in the self-certified ring signature verification phase. Let *Type I attacker* be $\mathcal{A}_I$ for short.

- *Type II attack*: The *Type II attacker* represents a dishonest TTP who tries to compromise the user's private key in the witness registration phase. Let *Type IIattacker* be $\mathcal{A}_{II}$ for short.

Game 1: In this game, we let the adversary be an uncertified user (*Type I attacker*) who tries to forge a valid self-certified ring signature with a forged witness.

Setup: The challenger $\mathcal{C}$ runs **SysSetup** to generate public parameters Params and the master secret key. Then, $\mathcal{C}$ gives Params to the adversary.

Queries: $\mathcal{A}_I$ can adaptively issue the **Wit-Query** and **Signature-Query** queries to $\mathcal{C}$. These queries are answered as follows:

- **Wit-Query**: The adversary makes a witness query on (ID, PK, Q), $\mathcal{C}$ responds a valid witness by running **WitReg** algorithm. Let q_w be the number of witness queries in this phase.

- **Signature-Query**: $\mathcal{A}_I$ can query the signature for its choice $(m, \bigcup_{i=0}^{n-1}\{ID_i\})$. $\mathcal{C}$ generates witnesses and returns a signature σ on the message m. Let q_s be the number of signature queries in this phase.

Forgery: $\mathcal{A}_I$ outputs a message m^*, a signature σ^*, a set of identities and a set of witnesses such that $(m^*, \bigcup_{i=0}^{n-1}\{ID_i^*\})$ is not used in queries and the forged witness W^* is not generated by $\mathcal{C}$. The adversary wins the game if Verify$(m^*, \sigma^*, \bigcup_{i=0}^{n-1}\{ID_i^*\}$, $\bigcup_{i=0}^{n-1}\{W_i^*\})$ returns *true*. We denote the advantage of $\mathcal{A}_I$ as:

$$Adv_{\mathcal{A}_I} = \Pr\left[\begin{array}{l} Verify(m^*, \sigma^*, \bigcup_{i=0}^{n-1}\{ID_i^*\}, \\ \bigcup_{i=0}^{n-1}\{W_i^*\}) = true : \\ (PK_i, s_i) \xleftarrow{R} KeyGen(\lambda); \\ W_i \leftarrow WitReg(ID_i, PK_i, Q_i); \\ W^* \neq W_i \ for \ i \in \{1 \ldots, q_w\}; \\ m^* \neq m_i, \ for \ i \in \{1 \ldots, q_s\}; \\ (m^*, \sigma^*) \leftarrow \mathcal{A}_I(\bigcup\{ID^*\}, \bigcup\{W^*\}); \end{array}\right].$$

Definition 4.1 *We say that a self-certified ring signature scheme is (t, q_w, q_s, ϵ)-secure against Type I attack if there is no Type I attacker who wins Game 1 in t-time with advantage at least ϵ after q_w, q_s queries.*

Game 2: In this game, we let the adversary be a malicious TTP (*Type II attacker*) who tries to forge a self-certified ring signature using the chosen identity and the corresponding witness.

Setup: The challenger runs **SysSetup** to generate public parameters Params and master secret key msk. $\mathcal{C}$ gives Params and msk to the adversary.

Queries: $\mathcal{A}_{II}$ can adaptively issue the **Public-key-Query** and **Signature-Query** queries to $\mathcal{C}$. These queries are answered as follows:

- **Public-key-Query**: $\mathcal{A}_{II}$ makes a public key query on ID_i. $\mathcal{C}$ generates (PK_i, SK_i) and returns PK_i. Let q_p be the number of public key queries in this phase.

- **Signature-Query**: $\mathcal{A}_{II}$ makes a signature query on $(m, \bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0}^{n-1}\{W_i\})$. $\mathcal{C}$ responds a valid signature σ by running Sign algorithm. Let q_s be the number of signature queries in this phase.

Forgery: $\mathcal{A}_{II}$ forges a self-certified ring signature and wins if Verify$(m^*, \sigma^*, \bigcup_{i=0}^{n-1}\{ID_i^*\}, \bigcup_{i=0}^{n-1}\{W_i^*\})$ returns *true* where $(m^*, \bigcup_{i=0}^{n-1}\{ID_i^*\}, \bigcup_{i=0}^{n-1}\{W_i^*\})$ does not appear in Signature-Query. We denote the advantage of this adversary as:

$$Adv_{\mathcal{A}_{II}} = \Pr\left[\begin{array}{l} Verify(m^*, \sigma^*, \bigcup_{i=0}^{n-1}\{ID_i^*\}, \\ \bigcup_{i=0}^{n-1}\{W_i^*\}) = true: \\ W_i \leftarrow WitReg(ID_i, PK_i, Q_i); \\ W^* \neq W_i \ for \ i \in \{1 \ldots, q_w\}; \\ (m^*, \sigma^*) \leftarrow \mathcal{A}_{II}(ID, \bigcup_{i=0}^{n-1}\{W_i\}); \\ m^* \neq m_i, \ for \ i \in \{1 \ldots, q_s\}; \end{array}\right].$$

Definition 4.2 *We say that a self-certified ring signature scheme is (t, q_p, q_s, ϵ)-secure against Type II attack if there is no Type II attacker who wins Game 2 in t-time with advantage at least ϵ after q_p, q_s queries.*

4.2.3 SCRS Anonymity

Anonymity is the main feature of ring signatures. It requires that the adversary cannot tell which member in the group generates the signature in polynomial-time with the probability greater than $\frac{1}{n}$, where n is the number of ring members. We define a stronger security model of anonymity of self-certified ring signatures and a powerful adversary $\mathcal{A}_p$. The adversary holds all members' private keys while he/she makes a decision. The game is constructed as follows:

Game 3: In the game, we let $\mathcal{A}_p$ be an adversary who tries to guess the actual singer of a given signature with all users' keys and witnesses.

Setup: The challenger runs the **SysSetup** algorithm to generate public parameters Params and a master secret key x. $\mathcal{C}$ gives Params to the adversary.

Query: $\mathcal{A}_p$ makes a signature query of its choice $(\bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0}^{n-1}\{W_i\}, \bigcup_{i=0}^{n-1}\{SK_i\})$. $\mathcal{C}$ chooses a signer and runs the Sign algorithm to generate and return a signature σ. Let q_s be the number of signature queries in this phase.

Guess: $\mathcal{A}_p$ guesses the actual singer of a given signature and wins if $\mathcal{A}_p$ has successfully found the index of the singer in the set of identities. We denote the advantage of this adversary as:

$$Adv_{A_p} = \left| \Pr \left[\begin{array}{l} A_p^{guess}(m, \bigcup_{i=0}^{n-1} ID_i, \bigcup_{i=0}^{n-1} W_i, \\ \bigcup_{i=0}^{n-1} SK_i) = j : \\ j \in \{0, ...n-1\}; \\ (SK_i, PK_i) \xleftarrow{R} KeyGen(\lambda); \\ W_i \leftarrow WitReg(ID_i, PK_i, Q_i); \\ \sigma \xleftarrow{R'} Sign(m, \bigcup_{i=0}^{n-1} ID_i, \\ \bigcup_{i=0}^{n-1} W_i, \bigcup_{i=0}^{n-1} SK_i); \end{array} \right] - \frac{1}{n} \right|,$$

where elements in all sets are indexed as $0, \ldots, n-1$ and j is the index of the singer in the set. We define $\xleftarrow{R'}$ as "randomly select" a user to be the signer.

Definition 4.3 *We say that a self-certified signature scheme is (t, q_s, ϵ)-anonymous if there is no adversary who wins Game 3 in t-time with advantage at least ϵ after q_s queries.*

4.3 The Proposed Scheme

In this section, we present our self-certified ring signature scheme. Like CLRS and CBRS, it contains an interactive phase where the user requests a witness from the TTP. Since the certificate (witness) is used as a PPK in CLRS, the interaction must be protected by a secure channel that increases the cost and potential security problems. Otherwise, any one gets a certificate can generate a valid signature. Although the CBRS is no need to protect the certificate transmission, it still uses the certificate as a part of private key. In most CLRS and CBRS schemes, the user must keep these two elements. However, the witness in our scheme is a public parameter. The signing algorithm only requires the private key to be chosen by user. Normally, the length of private key in SCRS is half of that in CLRS and CBRS. In

addition, the signature and witness can be generated in parallel. It is useful in some potential applications. While the public key in our scheme is implicitly calculated in the verification, it can be explicitly recovered from the witness and the TTP's public key.

4.3.1 Construction

SysSetup: The TTP chooses a symmetric bilinear group defined in Section 2.1.3 $(g, q, \mathbb{G}, \mathbb{G}_T, e)$ and two collision-resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$. It also randomly selects $x, y \in_R \mathbb{Z}_q^*$ and sets $msk = (x, y)$, public keys $(U, V) = (g^x, g^{\frac{y}{x}})$. Finally, the TTP gives a set of the public parameters $\mathsf{Params}= (e, \mathbb{G}, \mathbb{G}_T, g, q, U, V)$.

KeyGen: The user randomly chooses a private key $s \in_R \mathbb{Z}_q^*$ and let $SK = s$. It computes the corresponding public key $PK = e(g, g)^s$.

WitReg: Let $(PK, SK) = (e(g, g)^s, s)$ be the user's public and private keys and do as follows:

- The user computes the $Q = V^s$ and then sends (ID, PK, Q) to the TTP.
- The TTP first verifies the user's Q. If the equation $e(Q, U^{\frac{1}{y}}) = PK$ holds, the TTP generates a witness as:

$$W = H_1(ID)^{\frac{1}{x}} Q^{\frac{1}{y}}.$$

- Upon receiving the witness, the user checks if the following equation holds. If so, the user accepts and publishes the witness; otherwise, rejects it. Remark that the user only publishes his/her identity and the witness.

$$\begin{aligned}
&e(W, U)e(H_1(ID), g)^{-1} \\
&= e(H_1(ID)^{\frac{1}{x}} Q^{\frac{1}{y}}, U)e(H_1(ID), g)^{-1} \\
&= e(H_1(ID)^{\frac{1}{x}} V^{\frac{s}{y}}, U)e(H_1(ID), g)^{-1} \\
&= e(H_1(ID)^{\frac{1}{x}}, g^x)e(g^{\frac{s}{x}}, g^x)e(H_1(ID), g)^{-1} \\
&= e(H_1(ID), g)e(g^s, g)e(H_1(ID), g)^{-1} \\
&= e(g^s, g) \\
&= PK.
\end{aligned}$$

Sign: Let the signer be the kth of the selected set of users. The user Takes as input a message $m \in \{0,1\}^*, \bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0,i\neq k}^{n-1}\{W_i\}$ and s_k, the user generates the self-certified ring signature as follows:

- Randomly chooses a number $\alpha \in_R \mathbb{Z}_q^*$ to compute, $c_{k+1} = H_2(L||m||e(g,g)^\alpha)$, where L is the list of selected IDs (include the signer), such that $L = ID_0||ID_1 ||\ldots||ID_{n-1}$.
- Randomly selects $r_i \in_R \mathbb{Z}_q^*$, for $i = k+1, k+2, \ldots, n-1, 0, \ldots, k-1$, then compute each c_{i+1} by

$$c_{i+1} = H_2(L||m||e(g^{r_i}H_1(ID_i)^{-c_i}, g)e(W_i^{c_i}, U)).$$

- To form a ring, the user uses the private key (s_k) and calculates,

$$r_k = \alpha - s_k c_k \pmod q.$$

  The signature of m is $\sigma = (c_0, r_0, r_1, \ldots, r_{n-1})$.

Verify: Taking as input $(m, \sigma, \bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0}^{n-1}\{W_i\})$, the user computes c_{i+1} as above, for $i = 0, 1, \ldots, n-1$. Accept the signature if $c_0 = c_n$, otherwise reject it.

4.3.2 Correctness

Our self-certified ring signature scheme is correct as the following equation holds:

$$\begin{aligned}
c_{k+1} &= H_2(L||m||e(g^{r_k}H_1(ID_k)^{-c_k}, g)e(W_i^{c_k}, U)) \\
&= H_2(L||m||e(g^{r_k}H_1(ID_k)^{c_k}, g)e(H_1(ID_k)^{\frac{1}{x}}Q^{\frac{1}{y}}, g^x)^{c_k}) \\
&= H_2(L||m||e(g^{\alpha - s_k c_k}, g)e(g^{s_k c_k}, g) \\
&= H_2(L||m||e(g,g)^\alpha).
\end{aligned}$$

4.4 Security Analysis

The security of self-certified ring signatures contains two parts, the unforgeability and the anonymity. Different from certificateless and certificate-based ring signatures, the public key replacement attack in [HSMZ05] and [LHM+07] is no longer valid in self-certified signatures. Our scheme is secure if there is no adversary who wins any of the following games.

4.4.1 Game 1 Security

Theorem 4.1 *Our self-certified ring signature scheme is $(t, q_1, q_w, q_s, \epsilon)$-secure against Type I attack if the k+1EP is (t', ϵ')-hard, where*

$$\epsilon' \geq (1-p)^{q_w+q_s} p\epsilon, \quad t' \leq t + (q_1 + (3n-1)q_s + 3q_w)t_{pa} + 2(n-1)q_s t_{exp}.$$

Here, q_1 is number of queries on H_1 hash function, t_{exp} is the running time of one exponentiation on a group and t_{pa} is the running time of one bilinear paring operation.

Proof: Suppose there exists a *Type I attacker* who can (t, q_w, q_s, ϵ)-break our SCR signature scheme. We construct an algorithm $\mathcal{B}$ run by the challenger to solve the k+1 exponent problem. The algorithm $\mathcal{B}$ receives the instance $(g^{\frac{1}{a}}, g, g^a, g^{a^2})$ of k+1 exponent problem ($k = 3$) and aims to output g^{a^3} using the algorithm $\mathcal{A}_I$. Science P is a generator of a group $\mathbb{G}_1$ with a prime order, we let $g' = g^{\frac{1}{a}}$, then it is exactly an instance of k+1EP. The interaction between $\mathcal{B}$ and an adversary $\mathcal{A}_I$ is as follows.

Setup: In Game 1, $\mathcal{B}$ sets the public key $U = g^{\frac{1}{a}}$ and $V = g^{ya}$, where $y \in_R \mathbb{Z}_q^*$. Then it gives Params to $\mathcal{A}_I$ and the master secret key $\frac{1}{a}$ is unknown to $\mathcal{B}$. $\mathcal{B}$ creates and maintains four lists $L_{H_1} = \{((ID, con \in \{0,1\}), u)\}$, $L_{H_2} = \{(L, m, \theta, R)\}$, $L_{W_1} = \{(ID, PK, Q, W)\}$ and $L_{W_2} = \{(ID, SK, W)\}$, which are initially empty.

H_1 **Query**: $\mathcal{A}_I$ can query the result of H_1 on ID_i at most q_1 times, where $1 \leq i \leq q_1$. If the queried ID_i is in the list L_{H_1}, $H_1(ID_i)$ will be returned as the answer. Otherwise, $\mathcal{B}$ chooses $u_i \in_R \mathbb{Z}_q^*$ and responds as follows:

- Tosses a coin with the probability such that:

  $$\Pr[con = 1] = p, \ \ \Pr[con = 0] = 1 - p$$

  and sets the result as con_i.

  $$\begin{cases} con_i = 0 : H_1(ID_i) = g_i^u, \\ \\ con_i = 1 : H_1(ID_i) = g^{u_i a^2}. \end{cases}$$

- Outputs $H_1(ID_i)$ as the answer and adds $((ID_i, con_i), u_i)$ into the list L_{H_1}.

H_2 **Query**: $\mathcal{A}_I$ can query the result of H_2 on input (L_j, m_j, θ_j) at most q_2 times, where $\theta_j \in \mathbb{G}_2$ and $1 \leq j \leq q_2$. If (L_j, m_j, θ_j) is in the list L_{H_2}, R_j will be returned as the answer. Otherwise, $\mathcal{B}$ randomly chooses $R_j \in_R \mathbb{Z}_q^*$ and sets $H_2(L_j||m_j||\theta_j) = R_j$. Then $\mathcal{B}$ responds with R_j and adds $(L_j, m_j, \theta_j, R_j)$ into the list L_{H_2}.

Queries:

- **Wit-Query**: Let (PK_i, SK_i) be the $\mathcal{A}_I$'s selected public and private keys where $PK_i = e(g,g)^{s_i}$ and $SK_i = s_i$. $\mathcal{A}_I$ can query a witness on (ID_i, PK_i, Q_i) at most q_w times. If Q_i is valid, $\mathcal{B}$ responds the query and maintains the list L_{W_1} as follows:
  - If ID_i is not in L_{H_1}, $\mathcal{B}$ runs H_1 **Query**.
  - If $con = 1$, $\mathcal{B}$ aborts the simulation and returns FAIL.
  - If ID_i is already in the list $L_{W_1} \cup L_{W_2}$, $\mathcal{B}$ returns W_i and PK_i cannot be replaced.
  - Otherwise, $\mathcal{B}$ computes and returns a witness as

    $$W_i = (V^{u_i} Q_i)^{\frac{1}{y}},$$

    and adds (ID_i, PK_i, Q_i, W_i) into the list L_{W_1}.
- **Signature-Query**: $\mathcal{A}_I$ queries a signature on $(m_j, \bigcup_{i=0}^{n-1}\{ID_i\})$ at most q_s times and $1 \leq j \leq q_s$. $\mathcal{B}$ responds the query and maintains the list L_{W_2} as follows:
  - If ID_i is not in the list $L_{W_1} \cup L_{W_2}$, where $i \in \{0, n-1\}$. $\mathcal{B}$ runs H_1 **Query**.
  - If $con_i = 0$, sets $SK_i = s_i$, $W_i = V^{\frac{u_i + s_i}{y}}$, $s_i \in_R \mathbb{Z}_q^*$ and adds (ID_k, SK_k, W_k) to the list L_{W_2}.
  - If $\forall ID \in \bigcup_{i=0}^{n-1}\{ID_i\}$, $con = 1$, $\mathcal{B}$ aborts the simulation and returns FAIL.
  - Chooses an index $k \in_R \{0, \ldots, n-1\} \backslash \{i : con_i = 1\}$.
  - If ID_k is in the list L_{W_2}, $\mathcal{B}$ responds the as in **Sign** algorithm.
  - Otherwise, selects $\alpha \in_R \mathbb{Z}_q^*$, and computes

    $$c_{k+1} = H_2(L_j||m_j||e(g,g)^{\alpha}),$$

where $L_j = ID_0||ID_1||\ldots||ID_{n-1}$. If c_{k+1} is in the list L_{H_2}, $\mathcal{B}$ repeats this step.

– Selects $r_i \in_R \mathbb{Z}_q^*$ and computes

$$c_{i+1} = H_2(L_j||m_j||e(g^{r_i}H_1(ID_i)^{-c_i}, g)e(W_i^{c_i}, U)),$$

where $i = k+1, \ldots, n-1, 0, \ldots, k-1$. $\mathcal{B}$ sets $R_{i+1} = c_{i+1}$, $\theta_{i+1} = e(g^{r_i}H_1(ID_i)^{-c_i}, g)e(W_i^{c_i}, U)$ and adds $(L_j, m_j, \theta_{i+1}, R_{i+1})$ into the list L_{H_2}.

– $\mathcal{B}$ selects $r_k \in_R \mathbb{Z}_q^*$ that (L_j, m_j, θ_k) is not in the list L_{H_2}, where $\theta_k = e(g^{r_k}H_1(ID_k)^{-c_k}, g)e(W_k^{c_k}, U)$. Sets $R = c_{k+1}$ and adds (L_j, m_j, θ_k, R) into the list L_{H_2}.

– Returns the signature $\sigma = (c_0, r_0, r_1, \ldots, r_{n-1})$.

Forgery: $\mathcal{A}_I$ forges a signature of a message m^* after queries. Let $\sigma^* = (c_0^*, r_0^*, r_1^*, \ldots, r_{n-1}^*)$ be a forgery signature generated by fake witnesses. According to the forking lemma [PS96], if the Verify$(m^*, \sigma^*, \bigcup_{i=0}^{n-1}\{ID_i^*\}, \bigcup_{i=0}^{n-1}\{W_i^*\})$ outputs *true* and $(\bigcup_{i=0}^{n-1}\{ID_i^*\}, m^*)$ is never used in queries, $\mathcal{B}$ can get another valid forgery. $\mathcal{B}$ replays $\mathcal{A}_I$ with the same random tape but the different H_2. Suppose the two H_2 outputs h and h' respectively, which $h \neq h'$. Let the second forgery be $\sigma^{*\prime} = (c_0^{*\prime}, r_0^{*\prime}, r_1^{*\prime}, \ldots, r_{n-1}^{*\prime})$. $\mathcal{B}$ thus gets

$$\begin{cases} r_k^* = \alpha_k^* - s_k^* c_k^*, \\ \\ r_k^{*\prime} = \alpha_k^* - s_k^* c_k^{*\prime}. \end{cases}$$

Then $\mathcal{B}$ can solve the k+1 exponent problem (k=3) as the following equations hold:

$$s_k^* = \frac{r_k^* - r_k^{*\prime}}{c_k^{*\prime} - c_k^*},$$

$$W^* = g^{au_k^* a^2} V^{\frac{s_k^*}{y}},$$

$$g^{a^3} = \frac{W^* V^{\frac{-s_k^*}{y}}}{u_k^*}.$$

Probability: Let the event that a *Type I attacker* outputs a valid forgery be forge. Let event that $\mathcal{B}$ successfully completes a simulation be sim. The probability ϵ' of $\mathcal{B}$ to solve the k+1 exponent problem is bounded as

$$\epsilon' \geq \Pr[\mathsf{forge} \wedge \mathsf{sim}] = \Pr[\mathsf{forge}] \cdot \Pr[\mathsf{sim}].$$

In this game, there are two cases that $\mathcal{B}$ aborts a simulation. The probability of each case is as follows:

- The probability that $\mathcal{B}$ does not abort during witness queries is $(1-p)^{q_w}$.
- The probability that $\mathcal{B}$ does not abort during signature queries is $(1-p^n)^{q_s}$.

Since we have $\Pr[\mathsf{forge}] = \epsilon$, the probability ϵ' is

$$\begin{aligned}\epsilon' &\geq (1-p)^{q_w} \cdot (1-p^n)^{q_s} \cdot p\epsilon \\ &\geq (1-p)^{q_w+q_s} p\epsilon.\end{aligned}$$

Let t_exp be the running time of one exponentiation on a group and t_pa be the running time of one bilinear pairing operation. We have the running time of $\mathcal{B}$ is $t' \leq t + (q_1 + (3n-1)q_s + 3q_w)t_{pa} + 2(n-1)q_s t_{exp}$.

4.4.2 Game 2 Security

Theorem 4.2 *Our SCR signature scheme is $(t, q_1, q_p, q_s, \epsilon)$-secure against Type II attack if the DL problem is (t', ϵ')-hard, where*

$$\epsilon' = \epsilon, \quad t' \leq t + (q_1 + 2q_p + (3n-2)q_s)t_{exp} + (q_p + 2(n-1)q_s)t_{pa}.$$

Here, q_1 is number of queries on H_1 hash function, t_{exp} is the running time of one exponentiation on a group and t_{pa} is the running time of one bilinear paring operation.

Proof: Suppose there exists a Type II attacker who can (t, q_p, q_s, ϵ)-break our self-certified ring signature scheme. We construct an algorithm $\mathcal{B}$ run by the challenger to solve the DL problem. The algorithm $\mathcal{B}$ receives the instance (g, g^a) of DL problem and aims to outputs a using the algorithm $\mathcal{A}_{II}$. The interaction between $\mathcal{B}$ and $\mathcal{A}_{II}$ is as follows.

Setup: In Game 2, $\mathcal{B}$ randomly chooses $x, y \in_R \mathbb{Z}_q^*$ and sets $(U, V) = (g^x, g^{\frac{y}{x}})$, where (x, y) are master secret keys. Then it gives (x, y) and Params to $\mathcal{A}_{II}$. $\mathcal{B}$ creates and maintains three lists $L_{H_1} = \{(ID, u)\}$, $L_{H_2} = \{(L, m, \theta, R)\}$ and $L_K = \{(ID, PK, Q, s)\}$, which are initially empty.

H_1 **Query**: $\mathcal{A}_{II}$ can query the result of H_1 on ID_i at most q_1 times, where $1 \leq i \leq q_1$. If ID_i is in L_{H_1}, $H_1(ID_i)$ is returned. Otherwise, $\mathcal{B}$ returns $H_1(ID_i) = g^{u_i}$, where $u_i \in_R \mathbb{Z}_q^*$ and adds (ID_i, u_i) to L_{H_1}.

H_2 **Query**: $\mathcal{A}_{II}$ can query the result of H_2 on input (L_j, m_j, θ_j) at most q_2 times, where $\theta_j \in \mathbb{G}_2$ and $1 \leq j \leq q_2$. If (L_j, m_j, θ_j) is in L_{H_2}, $H_2(L_j||m_j||\theta_j)$ will be returned as the answer. Otherwise, $\mathcal{B}$ sets $H_2(L_j||m_j||\theta_j) = R_j$, where $R_j \in_R \mathbb{Z}_q^*$. Then, $\mathcal{B}$ responds R_j and adds $(L_j, m_j, \theta_j, R_j)$ into L_{H_2}.

Queries:

- **Public-key-Query**: $\mathcal{A}_{II}$ can query the public key PK_i and the corresponding proof of knowledge of private key Q_i on input ID_i at most q_p times that $1 \leq i \leq q_p$. $\mathcal{B}$ responds the query and maintains the list L_K as follows:
  - If ID_i is in the list L_K, $\mathcal{B}$ returns the corresponding PK_i and Q_i.
  - Otherwise, $\mathcal{B}$ randomly chooses a number $s_i \in_R \mathbb{Z}_q^*$ and sets $(PK_i, Q_i) = (e(g^{s_i xa}, g), g^{s_i ya})$. Then, $\mathcal{B}$ returns (PK_i, Q_i) to $\mathcal{A}_{II}$ and adds (ID_i, PK_i, Q_i, s_i) into the list L_K.
- **Signature-Query**: $\mathcal{A}_{II}$ can query a signature on inputs $(m_j, \bigcup_{j=0}^{n-1} ID_j, \bigcup_{j=0}^{n-1} W_j)$ at most q_s times where $1 \leq j \leq q_s$. Since the $\mathcal{B}$ responds a signature query as follows:
  - Chooses an index $k \in_R \{0, \ldots, n-1\}$.
  - Selects $\alpha \in_R \mathbb{Z}_q^*$ and computes $c_{k+1} = H_2(L_j||m_j||e(g,g)^\alpha)$, where $L_j = ID_0||ID_1||\ldots||ID_{n-1}$. If c_{k+1} is in the list L_{H_2}, $\mathcal{B}$ repeats this step.
  - Selects $r_i \in_R \mathbb{Z}_q^*$ and computes

    $$c_{i+1} = H_2(L_j||m_j||e(g^{r_i} H_1(ID_i)^{-c_i}, g)e(W_i^{c_i}, U)),$$

    where $i = k+1, \ldots, n-1, 0, \ldots, k-1$. $\mathcal{B}$ sets $R_{i+1} = c_{i+1}$, $\theta_{i+1} = e(g^{r_i} H_1(ID_i)^{-c_i}, g)e(W_i^{c_i}, U)$ and adds $(L_j, m_j, \theta_{i+1}, R_{i+1})$ into the list L_{H_2}.
  - $\mathcal{B}$ selects $r_k \in_R \mathbb{Z}_q^*$ that (L_j, m_j, θ_k) is not in the list L_{H_2}, where $\theta_k = e(g^{r_k} H_1(ID_k)^{-c_k}, P)e(W_k^{c_k}, U)$. Sets $R = c_{k+1}$ and adds (L_j, m_j, θ_k, R) into the list L_{H_2}.
  - Returns the signature $\sigma = (c_0, r_0, r_1, \ldots, r_{n-1})$.

Forgery: $\mathcal{A}_{II}$ forges a signature of a message m^* after queries. Let $\sigma^* = (c_0^*, r_0^*, r_1^*, \ldots, r_{n-1}^*)$ be a forgery signature generated by $(\bigcup_{i=0}^{n-1}\{ID_i^*\}, \bigcup_{i=0}^{n-1}\{W_i^*\})$. If Verify$(\sigma^*, m^*,$

$\bigcup_{i=0}^{n-1}\{ID_i^*\}, \bigcup_{i=0}^{n-1}\{W_i^*\})$ outputs *true* and the pair that $(\bigcup_{i=0}^{n-1}\{ID_i^*\}, m^*)$ is never queried to $\mathcal{B}$, then $\mathcal{B}$ can apply the forking lemma. $\mathcal{B}$ replays the same random tape but different H_2. Suppose the two different H_2 outputs h and h' respectively. $\mathcal{B}$ can use the algorithm $\mathcal{A}_{II}$ to get another valid forgery $\sigma^{*\prime} = (c_0^{*\prime}, r_0^{*\prime}, r_1^{*\prime}, \ldots, r_{n-1}^{*\prime})$. Hence, $\mathcal{B}$ can solve the DL problem as follows:

$$s_k^* = \frac{r_k^* - r_k^{*\prime}}{c_k^{*\prime} - c_k^*},$$

$$a = (s_k x)^{-1} \frac{r_k^* - r_k^{*\prime}}{c_k^{*\prime} - c_k^*}.$$

Probability: Since there is no case that $\mathcal{B}$ aborts the simulation, the probability of B is $\epsilon' = \epsilon$. We denote as t_exp the running time of one exponentiation and t_pa the running time of one paring operation. The total running time of $\mathcal{B}$ is $t' \leq t + (q_1 + 2q_p + (3n-2)q_s)t_{exp} + (q_p + 2(n-1)q_s)t_{pa}$.

4.4.3 SCR Anonymity

Theorem 4.3 *Our self-certified ring signature scheme is* (t, q_s, ϵ)*-anonymous.*

Proof: To prove the anonymity of our SCR signature scheme, we are aiming to show that there is no PPT algorithm that has non-negligible advantage to find the actual signer of the given signature. Suppose there is a powerful adversary $\mathcal{A}_p$ who has a set of private keys $\bigcup_{i=0}^{n-1} s_i$ and the corresponding witnesses of identities, we construct an algorithm $\mathcal{B}$ run by $\mathcal{C}$ to interact with $\mathcal{A}_p$ as follows:

Setup: In Game 3, taking as input a security parameter λ, $\mathcal{B}$ returns Params and master secret key (x, y). $\mathcal{B}$ gives Params to $\mathcal{A}_p$.

Query: $\mathcal{A}_p$ queries a signature on inputs $(m_i, \bigcup_{i=0}^{n-1}\{ID_i\}, \bigcup_{i=0}^{n-1}\{W_i\}, \bigcup_{i=0}^{n-1}\{s_i\})$ at most q_s times that $1 \leq i \leq q_s$, $\mathcal{B}$ randomly chooses a user j to be the actual signer, where $j \in_R \{0, \ldots, n-1\}$, and returns to $\mathcal{A}_p$ a signature using SCR signature scheme.

Guess: $\mathcal{A}_p$ chooses a message m^* and a set of identities $\bigcup_{i=0}^{n-1}\{ID_i\}$ which is used before. It gets a signature $\sigma^* = (c_0^*, r_0^*, \ldots, r_j^*, \ldots, r_{n-1}^*)$ from $\mathcal{B}$. Hence, we have $r_j^* = \alpha_j^* - s_j^* c_j^*$, $\alpha_j^* \in_R \mathbb{Z}_q^*$. Then $\mathcal{A}_p$ tries to output the index of actual signer j.

Because of that r_i^*, $i \in \{0, \ldots, n-1\}$ is randomly chosen from $\mathbb{Z}_q^*$, any user in T^* has the same probability to get the same random tape. It seems that the signer's r_j^* is not randomly selected, but it is generated by a random value $\alpha_j^* \in_R \mathbb{Z}_q^*$. Therefore,

they are all random to the adversary. Moreover, for any user $k \in \{0, \ldots, n-1\} \backslash \{j\}$, there is a number $\alpha_k^* \in \mathbb{Z}_q^*$ that lets $r_k^* = \alpha_k^* - s_k^* c_k^*$ and makes the same signature. Therefore, the advantage of $\mathcal{A}_p$ is

$$\epsilon \leq 2^{-\lambda},$$

which is negligible. We have proved Theorem 3.

4.5 Conclusion

In this chapter, we proposed a new notion, *Self-Certified Ring Signature* (SCRS). It solved the private key escrow and certificate management problems. Since our scheme embedded the public key into the witness, it reduces the cost of storage, communication and computation. We compared it with two related schemes: certificateless ring signatures and certificate-based ring signatures. Our SCRS is better due to shorter key size and lower setup cost. We proposed a precise definition of self-certified ring signatures and provided a concrete scheme. Our scheme has been proven secure.

Chapter 5

Conclusion

In this thesis, we studied the notion of self-certified digital signatures. It aims to resolve the certificate management problem and key escrow problem in traditional PKI-based digital signatures and identity-based signatures respectively. In a self-certified signature scheme, a trusted third party who generates witnesses for users has been employed. Equipped with a valid witness, anyone who accepts public parameters of the witness issuer and an identity of the witness holder can explicitly recover the holder's genuine public key. Actually, a self-certified signature scheme embeds the public key verification process into a signature verification. That is, once a signature is valid, it implies that the relationship between a user's identity and a public key is successfully authenticated.

In Chapter 1, we reviewed the concept of digital signatures. The validity of a user's identity and his/her public key is provided by checking the public key certificate. Since traditional PKI needs certificates, certificate management becomes an inherent problem. We presented several existing solutions of this issue and compared their advantages and disadvantages. Some related work of this thesis is briefly reviewed in this chapter, including identity-based signatures, certificateless signatures, certificate-based signatures, online/offline signatures and ring signatures, etc.

Different types of digital signature schemes were formally defined in Chapter 2, such as, self-certified signature schemes, certificate-based signature schemes and ring signature schemes. In addition, some mathematical definitions and underlying hard problems were also given.

In Chapter 3, we described an efficient construction of self-certified signature with batch verification. A formal definition of self-certified digital signature schemes is presented along with a security model. According to the security levels for self-certified signatures defined in [Gir91], we specified two types of adversaries. Our proposed scheme is proved to be secure against both attacks. Comparing with

existing certificateless and certificate-based signature schemes which have the same purpose as self-certified signatures, our construction is more efficient since it captures two features. The signing process requires only one multiplicative calculation with pre-computation. The scheme naturally provides the property of batch verification. Especially, bilinear pairing computations in multi-signer setting are independent of the size of the set. Along with the features of self-certified signatures, our scheme is suited for restricted computing power and low bandwidth communications in multi-user environment.

In Chapter 4, we introduced the first construction of self-certified ring signature scheme. The notion of self-certified ring signatures is extended from the conception of self-certified signatures and ring signatures. Meanwhile, a formal definition of self-certified signature schemes has been presented along with its security requirements. Our proposed scheme is based on a generic construction of ring signature schemes in [AOS02] and secure against two types of attacks in the random oracle model. The security proof depends on the hardness of k+1EP and Discrete Logarithm problem. Additionally, a property of perfect anonymity is also provided. Since public key certificates are unnecessary in our scheme, it can be efficiently used in large scale applications.

Both of our proposed schemes capture all features of original self-certified signatures and the strongest security notion. They prevent both certificate management problem and key escrow problem. The cost for communications has also been reduced.

Future work: Many digital signature schemes based on self-certified public keys have been proposed. Unfortunately, all were proved secure under the random oracle model based on strong assumptions. In my future research, I will construct a provably secure self-certified digital signature scheme without random oracles under weak complexity assumptions.

Bibliography

[AdM04] Giuseppe Ateniese and Breno de Medeiros. Identity-based chameleon hash and applications. In Ari Juels, editor, *Financial Cryptography*, volume 3110 of *LNCS*, pages 164–180. Springer, 2004.

[ALSY06] Man Ho Au, Joseph K. Liu, Willy Susilo, and Tsz Hon Yuen. Constant-size id-based linkable and revocable-iff-linked ring signature. In Rana Barua and Tanja Lange, editors, *INDOCRYPT*, volume 4329 of *LNCS*, pages 364–378. Springer, 2006.

[ALSY07] Man Ho Au, Joseph K. Liu, Willy Susilo, and Tsz Hon Yuen. Certificate based (linkable) ring signature. In Ed Dawson and Duncan S. Wong, editors, *ISPEC*, volume 4464 of *LNCS*, pages 79–92. Springer, 2007.

[ALYW06] Man Ho Au, Joseph K. Liu, Y. H. Yuen, and Duncan S. Wong. Id-based ring signature scheme secure in the standard model. In Hiroshi Yoshiura, Kouichi Sakurai, Kai Rannenberg, Yuko Murayama, and Shin ichi Kawamura, editors, *IWSEC '06*, volume 4266 of *LNCS*, pages 1–16. Springer, 2006.

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In Yuliang Zheng, editor, *Advances in Cryptology - Asiacrypt '02*, volume 2501 of *LNCS*, pages 415–432. Springer, 2002.

[ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *Advances in Cryptology-Asiacrypt '03*, volume 2894 of *LNCS*, pages 452–473. Springer, 2003.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[BDZ03] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of diffie-hellman problem. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *ICICS*, volume 2836 of *LNCS*, pages 301–312. Springer, 2003.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98, LNCS*, volume 1403 of *LNCS*, pages 236–250. Springer, 1998.

[BL96] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *LNCS*, pages 283–297. Springer, 1996.

[BLS04a] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Efficient implementation of pairing-based cryptosystems. *J. Cryptology*, 17(4):321–334, 2004.

[BLS04b] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 390–399. ACM, 2006.

[Bon04] Robert J. Boncella. Web services and web services security. *Communications of the Association for Information Systems*, 14:344–363, 2004.

[BP00] Colin Boyd and Chris Pavlovski. Attacking and repairing batch verification schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *LNCS*, pages 58–71. Springer, 2000.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making uowhfs practical. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *LNCS*, pages 470–484. Springer, 1997.

[CA89] David Chaum and Hans Van Antwerpen. Undeniable signatures. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 212–216. Springer, 1989.

[CC03] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *LNCS*, pages 18–30. Springer, 2003.

[Cho09] Sherman S. M. Chow. Removing escrow from identity-based encryption. In *Public Key Cryptography*, volume 5443 of *LNCS*, pages 256–276. Springer, 2009.

[CHP07] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *LNCS*, pages 246–263. Springer, 2007.

[CLHY05] Sherman S.M. Chow, Richard W.C. Lui, Lucas C.K. Hui, and S.M. Yiu. Identity based ring signature: Why, how and what next. In David W. Chadwick and Gansen Zhao, editors, *EuroPKI 2005*, volume 3545 of *LNCS*, pages 144–161. Springer, 2005.

[CPHL07] Kyu Young Choi, Jong Hwan Park, Jung Yeon Hwang, and Dong Hoon Lee. Efficient certificateless signature schemes. In Jonathan Katz and Moti Yung, editors, *ACNS '07*, volume 4521 of *LNCS*, pages 443–458. Springer, 2007.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David Wagner, editor, *CRYPTO*, volume 5157 of *LNCS*, pages 1–20. Springer, 2008.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *LNCS*, pages 410–424. Springer, 1997.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.

[CWMZ09] Shuang Chang, Duncan S. Wong, Yi Mu, and Zhenfeng Zhang. Certificateless threshold ring signature. *Inf. Sci.*, 179(20):3685–3696, 2009.

[CY07] Sherman S.M. Chow and Wun-She Yap. Certificateless ring signatures. In *Cryptology ePrint Archive, Report 2007/236, 2007.* http://eprint.iacr.org/2007/236/, 2007.

[CYH05] Sherman S.M. Chow, S.M. Yiu, and Lucas C.K. Hui. Efficient identity based ring signature. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS '05*, volume 3531 of *LNCS*, pages 499–512. Springer, 2005.

[Dam87] Ivan Damgård. Collision free hash functions and public key signature schemes. In *EUROCRYPT*, pages 203–216, 1987.

[DBS04] Ratna Dutta, Rana Barua, and Palash Sarkar. Pairing-based cryptographic protocols: A survey. In *In Cryptology ePrint Archive, Report 2004/064*, 2004.

[DGSW02] J. Dankers, T. Garefalakis, R. Schaffelhofer, and T. Wright. Public key infrastructure in mobile systems. *IEEE Electronics and Communication Engineering Journal*, 14(5):180–189, 2002.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[EGM90] Shimon Even, Oded Golreich, and Silvio Micali. On-line/off-line digit signatures. In Gilles Brassard, editor, *CRYPTO '89*, volume 435 of *LNCS*, pages 263–275. Springer, 1990.

[FGHP09] Anna Lisa Ferrara, Matthew Green, Susan Hohenberger, and Michael Østergaard Pedersen. Practical short signature batch verification. In Marc Fischlin, editor, *CT-RSA*, volume 5473 of *LNCS*, pages 309–324. Springer, 2009.

[Fia90] Amos Fiat. Batch RSA. In Gilles Brassard, editor, *CRYPTO '89*, volume 435 of *LNCS*, pages 175–185. Springer, 1990.

[Fia97] Amos Fiat. Batch RSA. *J. Cryptology*, 10(2):75–88, 1997.

[FMR99] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.

[Gen03] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In Eli Biham, editor, *EUROCRYPT '03*, volume 2656 of *LNCS*, pages 272–293. Springer, 2003.

[GH07] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In Kaoru Kurosawa, editor, *ASIACRYPT '07*, volume 4833 of *LNCS*, pages 265–282. Springer, 2007.

[Gir91] Marc Girault. Self-certified public keys. In *EUROCRYPT '91*, volume 547 of *LNCS*, pages 490–497. Springer, 1991.

[GMC08] Fuchun Guo, Yi Mu, and Zhide Chen. Efficient batch verification of short signatures for a single-signer setting without random oracles. In Kanta Matsuura and Eiichiro Fujisaki, editors, *IWSEC*, volume 5312 of *LNCS*, pages 49–63. Springer, 2008.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[GR06] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC '06*, volume 3958 of *LNCS*, pages 257–273. Springer, 2006.

[Gut02] Peter Gutmann. Pki: It's not dead, just resting. *IEEE Computer*, 35(8):41–49, 2002.

[GZ09] Manman Geng and Futai Zhang. Batch verification for certificateless signature schemes. In *CIS (2)*, pages 288–292. IEEE Computer Society, 2009.

[Har98] L. Harn. Batch verifying multiple DSA-type digital signatures. *Electronics Letters*, 34(9):870–871, 1998.

[Her07] Javier Herranz. Identity-based ring signatures from RSA. *Theor. Comput. Sci.*, 389(1-2):100–117, 2007.

[HMP95] Patrick Horster, Markus Michels, and Holger Petersen. Hidden signature schemes based on the discrete logarithm problem and related concepts. In *Proceeding of Communications and Multimedia Security '95*, pages 162–177. Chapman & Hall, 1995.

[HS04] Javier Herranz and Germán Sáez. New identity-based ring signature schemes. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *ICICS*, volume 3269 of *LNCS*, pages 27–39. Springer, 2004.

[HSMZ05] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. On the security of certificateless signature schemes from asiacrypt 2003. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *CANS '05*, volume 3810 of *LNCS*, pages 13–25. Springer, 2005.

[HWZD07] Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang, and Xiaotie Deng. Certificateless signature: A new security model and an improved generic construction. *Des. Codes Cryptography*, 42(2):109–126, 2007.

[KPH04] Bo Gyeong Kang, Je Hong Park, and Sang Geun Hahn. A certificate-based signature scheme. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *LNCS*, pages 99–111. Springer, 2004.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS*, 2000.

[LBSZ08] Joseph K. Liu, Joonsang Baek, Willy Susilo, and Jianying Zhou. Certificate-based signature schemes without pairings or random oracles. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *ISC '08*, volume 5222 of *LNCS*, pages 285–297. Springer, 2008.

[LHM+07] Jiguo Li, Xinyi Huang, Yi Mu, Willy Susilo, and Qianhong Wu. Certificate-based signature: Security model and efficient construction. In Javier Lopez, Pierangela Samarati, and Josep L. Ferrer, editors, *EuroPKI '07*, volume 4582 of *LNCS*, pages 110–125. Springer, 2007.

[LW04] Chih-Yin Lin and Tzong-Chen Wu. An identity-based ring signature scheme from bilinear pairings. In *AINA (2)*, pages 182–186. IEEE Computer Society, 2004.

[Mao04] Wenbo Mao. *Modern Cryptography: Theory & Practice*. Prentice Hall PTR, 2004.

[Mau94] Ueli M. Maurer. Towards the equivalence of breaking the diffie-hellman protocol and computing discrete algorithms. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *LNCS*, pages 271–281. Springer, 1994.

[McC90] Kevin S. McCurley. The discrete logarithm problem. In *Cryptology and Computational Number Theory*, volume 42 of *Proceedings of Sympasia in Applied Mathematics*, pages 49–74. American Mathematical Society, 1990.

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[MSK02] S Mitsunari, R Sakai, and M Kasahara. A new traitor tracing. *IEICE Tran.*, E85-A(2):481–484, 2002.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43. ACM, 1989.

[PH78] S.C. Pohig and M.E. Hellman. An improved algorithm for computing logrithms over $gf(p)$ and its cryptographic significance. *IEEE Transaction on Information Theory*, 24(1):106–110, 1978.

[PH97] Holger Petersen and Patrick Horster. Self-certified keys - concepts and applications. In *Proceeding of Communications and Multimedia Security '97*, pages 102–116. Chapman & Hall, 1997.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology - Proceedings of EUROCRYPT '96*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996.

[PS06] Kenneth G. Paterson and Jacob C. N. Schuldt. Efficient identity-based signatures secure in the standard model. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP '06*, volume 4058 of *LNCS*, pages 207–222. Springer, 2006.

[Riv92] R. Rivest. The md5 message-digest algorithm. RFC 1321, 1992.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *LNCS*, pages 371–388. Springer, 2004.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT '01*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.

[RST06] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret: Theory and applications of ring signatures. In Oded Goldreich, Arnold L. Rosenberg, and Alan L. Selman, editors, *Essays in Memory of Shimon Even*, volume 3895 of *LNCS*, pages 164–186. Springer, 2006.

[Sae97] Shahrokh Saeednia. Identity-based and self-certified key-exchange protocols. In Vijay Varadharajan, Josef Pieprzyk, and Yi Mu, editors, *ACISP '97*, volume 1270 of *LNCS*, pages 330–313. Springer, 1997.

[Sae03] Shahrokh Saeednia. A note on girault's self-certified model. *Inf. Process. Lett.*, 86(6):323–327, 2003.

[Sha49] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology Crypto '84*, volume 196 of *LNCS*, pages 47–53. Springer, 1985.

[SS10] Sven Schäge and Jörg Schwenk. A cdh-based ring signature scheme with short signatures and public keys. In Radu Sion, editor, *Financial Cryptography*, volume 6052 of *LNCS*, pages 129–142. Springer, 2010.

[ST01] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Joe Kilian, editor, *CRYPTO '01*, volume 2139 of *LNCS*, pages 355–367. Springer, 2001.

[Sta06] W. Stallings. *Cryptography and Network Security Principles and Practices*. Prentice Hall, 4th edition, 2006.

[SW07] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *LNCS*, pages 166–180. Springer, 2007.

[TS10] Mohsen Toorani and Ali Asghar Beheshti Shirazi. Lpki - a lightweight public key infrastructure for the mobile environments. *CoRR*, abs/1002.3299, 2010.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

[WMSH08] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. Certificate-based signatures: New definitions and a generic construction from certificateless signatures. In Kyo-Il Chung, Kiwook Sohn, and Moti Yung, editors, *WISA*, volume 5379 of *LNCS*, pages 99–114. Springer, 2008.

[WMSH09] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. Certificate-based signatures revisited. *J. UCS*, 15(8):1659–1684, 2009.

[XZF04] Jing Xu, Zhenfeng Zhang, and Dengguo Feng. A ring signature scheme using bilinear pairings. In Chae Hoon Lim and Moti Yung, editors, *WISA*, volume 3325 of *LNCS*, pages 160–169. Springer, 2004.

[YCHG07] Wun-She Yap, Sherman S. M. Chow, Swee-Huay Heng, and Bok-Min Goi. Security mediated certificateless signatures. In Jonathan Katz and Moti Yung, editors, *ACNS*, volume 4521 of *LNCS*, pages 459–477. Springer, 2007.

[YCK04] HyoJin Yoon, Jung Hee Cheon, and Yongdae Kim. Batch verifications with id-based signatures. In Choonsik Park and Seongtaek Chee, editors, *ICISC '04*, volume 3506 of *LNCS*, pages 233–248. Springer, 2004.

[YL95] Sung-Ming Yen and Chi-Sung Laih. Improved digital signature suitable for batch verification. *IEEE Trans. Computers*, 44(7):957–959, 1995.

[YSM10] Tsz Hon Yuen, Willy Susilo, and Yi Mu. How to construct identity-based signatures without the key escrow problem. In Fabio Martinelli and Bart Preneel, editors, *EuroPKI*, volume 6391 of *LNCS*, pages 286–301. Springer, 2010.

[ZCL04] Yuan Zhou, Zhenfu Cao, and Rongxing Lu. An efficient digital signature using self-certified public keys. In *Proceedings of The 3rd International Conference on Information Security*, volume 85, pages 44–47. ACM, 2004.

[ZK02] Fangguo Zhang and Kwangjo Kim. Id-based blind signature and ring signature from pairings. In Yuliang Zheng, editor, *Advances in Cryptology - Asiacrypt' 02*, volume 2501 of *LNCS*, pages 533–547. Springer, 2002.

[ZPS92] Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry. Haval - a one-way hashing algorithm with variable length of output. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT*, volume 718 of *LNCS*, pages 83–104. Springer, 1992.

[ZSNS04] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 277–290. Springer, 2004.

[ZZ08] Lei Zhang and Futai Zhang. A new provably secure certificateless signature scheme. In *ICC*, pages 1685–1689. IEEE, 2008.

[ZZW07a] Lei Zhang, Futai Zhang, and Wei Wu. A provably secure ring signature scheme in certificateless cryptography. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec*, volume 4784 of *LNCS*, pages 103–121. Springer, 2007.

[ZZW07b] Lei Zhang, Futai Zhang, and Wei Wu. A provably secure ring signature scheme in certificateless cryptography. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec '07*, volume 4784 of *LNCS*, pages 103–121. Springer, 2007.