Contributions to privacy preserving with ring signatures

YiQun Chen
University of Wollongong

Contributions to Privacy Preserving with Ring Signatures

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

Master of Computer Science by Research

from

UNIVERSITY OF WOLLONGONG

by

YiQun Chen

School of Information Technology and Computer Science
August 2006

Dedicated to
My Dad & Mum

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.

YiQun Chen
August 23, 2006

Abstract

A ring signature is a cryptographic primitive that enables a signer to produce a signature without revealing his or her identity. In this thesis, we propose two ring signature schemes for privacy-preserving applications over the Internet.

First we design a protocol that enables a ring signer to receive an acknowledgement from the verifier. We propose two constructions. With the basic construction, the verifier can send a message back to the original signer while keeping the latter's identity hidden. Additionally, the verifier is assured that the signer is indeed the user in a group. We then extend our basic construction to a multi-party scenario. In the second construction, the verifier can discern a certain number of signers involved in the specific signature while their identities remain anonymous. We also investigate the possible applications of our schemes, such as in E-Commerce and Pay-TV.

Then, we introduce the concept of *identity-based anonymous designated ring signatures*, which has not been studied before. This concept extends the existing notion of ring signatures in two ways: firstly, it allows a member of the ring to sign a message directed to a designated verifier. Secondly, we enable the concept of anonymous designated verifier. We show that it has useful applications in Peer-to-Peer networks and provide a construction based on bilinear pairings. Furthermore, we formulate a security model and prove the security of our proposed construction against a chosen message attack. We extend the scheme to construct a convertible version of the previous scheme, which enables a designated verifier to prove its participation in a particular session in case of dispute.

Acknowledgements

I would like to thank Associate Professor Yi Mu and Associate Professor Willy Susilo, my supervisors, for their patient guidance and constant support during my study. Many thanks for choosing me as their student. I admire their wealth of knowledge in security and cryptography, and appreciate them taking me into the area of cryptography. I am also grateful to Dr. Kathleen Weekley for her kind help in the English expression of this thesis. Finally, I would like to thank my family and friends for their enduring love and support.

Publications

YiQun Chen, Willy Susilo and Yi Mu. "Identity-based Anonymous Designated Ring Signatures", Computer and Network Security Symposium, IWCMC, 2006.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We start this thesis with the question:"What is privacy?" Different people may have different answers to this question. The contemporary idea of privacy is provided by Warren and Brandeis [102] who explicitly separate thoughts, sentiments, and emotions from other properties and identify them as private. Since then, privacy has been regarded as a fundamental right of human beings. Generally speaking, what is private is what people do not wish to share with others. More explicitly, privacy is the ability of an individual or group to stop information about themselves becoming known to people other than those to whom they choose to give the information [53]. Privacy is often associated with anonymity, which protects privacy by making one user indistinguishable from the others. It is not absolute though; the degree of anonymity can vary from *super-identification* to *anonymity* [38].

As we move into the Information Age, violation of privacy on the Internet has become a serious issue. According to a report published by Goldberg, Wagner and Brewer [43], the increased use of the Internet is bringing new threats to personal privacy. One scenario of such privacy violation is personal conversations. In the physical world, Alice and Bob can talk to each other in a place where no one is nearby, or they can talk on the telephone at home. All their conversations are going through a secure channel. However, the situation is different in the web environment. The adversary can use some network tools such as sniffer to eavesdrop on their conversations. Another scenario of privacy violation is shopping. When Alice purchases something over the Internet, the adversary is able to get as much information about her as he wants. He can find Alice's location by tracing her IP address; he can get Alice's birth dates by viewing her registration info or he might even discover Alice's hobbies by tracking her purchase activities.

The violation of privacy may have severe consequences. That is because private information such as names, addresses or birth dates may be misused against us. The

adversary can easily collect information about us and sell it to interested parties, and these parties then may send us junk mail, email advertisements, or promotions on the Internet. In fact, there are many companies that make a lot of money solely from the sale of personal information. Due to a report published by Federal Trade Commission, in 1998, the gross annual revenue of companies's selling information was $1.5 billion [57]. A more severe consequence of the misuse of personal information is fraud, such as the identity theft. That is adversary can use stolen information to open a home telephone or cellular account or he could even obtain a credit card under another person's name since many credit card companies and banks require only such personal information as a form of identification or proof of residency [59].

1.1 Challenges

As mentioned in the previous section, invasion of privacy has become a serious problem over the Internet. There are companies who collect and sell our personal information and there are individuals who use this information for illegal purposes. Therefore, countermeasures must be taken to protect our privacy. However, preserving privacy on the Internet is more difficult than in simplistic theoretical network models. Generally speaking, it faces the following challenges:

1. There is a conflict between privacy and security. Since security is far more important than privacy on the Internet, in order to maintain it, privacy is often compromised. For example, in Pay-TV systems, subscribers are required to authenticate themselves to the broadcaster. This reveals subscribers' identities and enables the broadcaster to collect information about them. However, without authentication, anyone may access the TV program. Therefore, in this thesis we try to deal with the following conflict: A user A would like to hide its identity when accessing the service while a service provider B would like to identify A for authentication purpose.

2. There is a conflict between privacy and other system requirements. For example, an anonymous user may want to designate the verification ability to group members. However, the designated party may not wish to reveal his membership unless he can be sure that the signer is also a member. In this thesis, we consider the following challenge issue: An offer $\mathcal{O}$ wish to share his file collection to a receiver $\mathcal{B}$ who belongs to a specific group, so they need to identify each other.

However neither the offer nor the receiver want to reveal his identities and they can identify each other only if they belong to the same group. Therefore, in this thesis we try to deal with the following conflict: An anonymous user A would like to designate the verification ability to specific member B while the member B does not wish to reveal its identity.

1.2 Existing Solutions

Many studies have shown that a majority of online users are concerned about the collection and use of their personal information [53]. For that reason, various privacy protection technologies have been proposed over the past decade.

One of the most basic privacy protection technologies is the proxy. With such technology, all communication between the user and the web owner is routed through a trusted server in such a way that it is infeasible to determine the IP address of the user. Therefore, without revealing any personal information about the user, a trusted server could conduct online activities on behalf of the user. Many companies provide such services over the Internet. For example, Anonymizer.com is a website which provides anonymous web surfing, FTP and other anonymous online activities [51]. However, such services alone may not be sufficient for privacy protection. That is because it only hides the IP address, the identity of the user can still be discovered. The web owner could collect enough personal information about the user if a certain kind of authentication is required. Besides, a trusted server may not be available for some web applications.

A more efficient way of providing privacy protection in the digital world is through cryptography. Cryptography provides various tools for privacy protection, such as encryption, digital signature, zero-knowledge proof. One commonly used tool is the blind signature which allows the signer to sign a message without knowing the content [24]. Since its appearance, many blind signature schemes have been proposed [1, 8, 2, 4]. This kind of signature scheme is widely used in anonymous payment systems such as electronic cash [29, 39, 42, 87, 77, 37]. A basic anonymous payment system involves three parties: the bank, the merchant and the customer. First, the customer makes some electronic coins and asks bank to sign it. The bank deducts the money from the customer's account and blindly signs the coin. After that, the customer purchases the goods and transfers the coins to the merchant. Then the merchant shows the coins to the bank and the bank deposits the money into merchant's account. Although

electronic cash has many advantages, it also has several drawbacks. For example, it only provides anonymity for the electronic cash, not the identity of the user. Another disadvantage of electronic cash is money laundering [14].

Mix-net is another widely used cryptographic tool which combines and forwards messages from several senders to several recipients so that the relation between any particular sender and any recipient can not be found [23]. It usually consists of a number of mix-centers which permute the list of inputs from the senders by re-encrypting the message in such way that the output list from a Mix-net has a different order from the input. A similar method is mentioned in [84] which groups users into a geographically diverse group called "Crowd" and directs a connecting request through it. The request passes a randomised number of crowd members and finally is submitted to the recipient who cannot identify which member in the crowd is the originator of the request. The Mix-net and crowd methods can provide anonymity to the user's identity. However, they are only suitable for a situation where authentication is not required.

Privacy protection can also be achieved by anonymous credential systems [25]. The anonymous credential systems allow users to interact with other parties using distinct and unlinkable credentials. That is to say, a user can obtain an credential from one party and then show it to a third party in such a way that the third party cannot link this credential to the one user registered at his place. A credential system also needs to be consistent. Each credential should belong to a particular user. It should be impossible for different users to collude together and produce a valid credential such that it can not be produced by any one of them on his own. This kind of system could achieve anonymity in authentication. However, it does not provide perfect anonymity. The identity of user can be revealed if several parties collude [31].

1.3 Aims and Objectives

Although there are many technologies available to protect user's privacy on the Internet, they all have disadvantages. Some of them can not be used when security is required, while others only protect a user's privacy from outsiders. Therefore, considering the challenging issues we described in the previous section, many of these approaches appear unsuitable for our requirements. For example, the proxy and Mix-net do not provide authentication ability while the credential systems does not provide perfect anonymity. In this thesis, we focus only on ring signatures, with the following objectives:

1. To design a protocol with ring signature which enables a user in the ring to communicate with an outsider anonymously. Meanwhile, the outsider should be able to communicate with the user.

2. To develop a new ring signature scheme which enables one anonymous party to authenticate himself to another designated party, whose identity is not revealed.

1.4 Contributions

In order to achieve our goals, we make use of *ring signatures*. A ring signature is a simplified group signature without any manager. It protects the anonymity of a signer. Due to its perfect anonymity property, ring signature is suitable for solving our problems. The detail introduction of ring signature will be given in the next chapter.

The contribution of this thesis is as follows:

- We design two protocols which allows user to protect its privacy while using authenticated services.

- We introduce the concept of *Anonymous Designated Ring Signatures* which allows both communication parties remain anonymous without compromising the designation property.

1.5 Structure of Thesis

This thesis is organised as follows. In Chapter 2, we review some background knowledge necessary for the subsequent chapters. First, we review some mathematical primitives used in our research. We then review related signature schemes, for example, the ring signature schemes on which we focus and the designated signature schemes which we use. We also illustrate the idea of zero knowledge proofs and commitment schemes. The security related background such as security modeling, security requirements and security proofs are covered in this chapter. Finally, we briefly review some useful cryptographic tools, such as hash functions, bilinear pairings, random oracle model.

In Chapter 3, we propose two proven secure and practical privacy protection protocols based on ring signatures. In our first construction, a user who initiates the communication is able to authenticate himself to another party anonymously. At the same time, that party can send back an acknowledgement in such way that only the

user can receive it. After that, we extend the first protocol to support a multi-party scenario. In our second construction, two or more users could collaborate to communicate with another party. That party is equipped with the ability to discern the number of users involved. As well, we investigate possible applications of our protocols, such as *E-Commerce* and *Pay-TV*.

In Chapter 4, we introduce the notion of *identity-based anonymous designated ring signature*. It extends the existing notion of ring signatures in two ways: Firstly, it allows a member from a ring to sign a message directed to a designated verifier. Secondly, we require an anonymous designated verifier. We show that this concept has some applications in Peer-to-Peer networks and provide a construction based on bilinear pairings. We formulate a security model and prove the security of our proposed construction against chosen message attacks. We then extend the scheme to construct a *convertible identity-based anonymous designated ring signatures*. With such a scheme, the anonymous designated verifier is able to prove his participation in a particular session.

In Chapter 5, we summarize the results of this thesis.

1.6 Notation and Abbreviations

In this section, we provide the general notations used in our thesis.

We use $\mathbb{Z}$ to denote the set of integers.

We use $[a, b]$ to denote an integer x satisfying $a \leqslant x \leqslant b$.

$|S|$ means the number of elements in S if S is a finite set, or the length of S if S is a string, or the bit-length of S if S is an integer.

By $x \in_R S$ we mean x is chosen from the finite set S uniformly at random.

Let $\mathcal{A}$ be an algorithm. By $\mathcal{A}(\cdot)$ we denote that $\mathcal{A}$ has one input, by $\mathcal{A}(\cdot, \cdot)$ two inputs and so on.

By $y \leftarrow \mathcal{A}(x)$ we mean that y is obtained by running algorithm $\mathcal{A}$ on input x. If $\mathcal{A}$ is deterministic, then y is unique.

Let S be a finite set, by $y \leftarrow S$ we denote that y is chosen from S uniformly at random.

Let $b(\cdot)$ be a boolean function (boolean means the results is either true or false), by $y \leftarrow A(x) : b(y)$ we denote the event that $b(y)$ is true after y was generated by A

on input of x. Hence the statement

$$Pr[\{y_i \leftarrow A_i(x_i)\}_{i\in[1,n]} : b(y_n)] = \alpha$$

means the probability of $b(y_n) = 1$ is α, where $b(y_n)$ is generated after algorithm A outputs y_n. Note two quantities are said to be equal if they are, in some well defined sense, equivalent. An equivalence test is used to assess whether one proportion is equivalent to another.

Chapter 2

Background

The goal of this chapter is to provide some background on the foundations of cryptography. First, we briefly review some mathematical primitives. We then proceed to related cryptographic knowledge, such as signature schemes, zero knowledge proofs, security proofs. Finally, we introduce some useful cryptographic tools, such as hash functions, bilinear pairings, random oracle model.

2.1 Mathematical Background

In this section, we review some mathematical primitives that will be used in the following chapters. Further background can be found in [72].

2.1.1 Number Theory

Definition 1 *Two integers a and b are said to be relatively prime or coprime if $gcd(a, b) = 1$ [72].*

gcd denotes the greatest common divisor, which is the largest positive integer that divides both a and b.

Definition 2 *If a and b are integers, then a is said to be congruent with b modulo n, written $a \equiv b \pmod{n}$, if n divides $(a - b)$. The integer n is called the modulus of the congruence [72].*

Definition 3 *The integers modulo n, denoted $\mathbb{Z}_n$, is the set of integers $\{0, 1, 2, ..., n - 1\}$. Addition, subtraction, and multiplication over $\mathbb{Z}_n$ are performed modulo n [72].*

Definition 4 *$\mathbb{Z}_q^*$ is the set of integers $\{1, 2, ..., q - 1\}$. Addition, subtraction, and multiplication over $\mathbb{Z}_q^*$ are performed modulo q [72].*

2.1.2 Group Theory

Definition 5 *A binary operation $*$ on a set S is a mapping from $S \times S$ to S. That is, $*$ is a rule which assigns to each ordered pair of elements from S an element of S [72].*

Definition 6 *A group is a set $\mathbb{G}$ together with an associative binary operation $*$ on elements of $\mathbb{G}$ such that $\mathbb{G}$ contains an identity element for $*$ and every element has an inverse under $*$. If $*$ is commutative, then the group is called Abelian or commutative. Often, a group is denoted by $\langle \mathbb{G}, * \rangle$ or simply by $\mathbb{G}$. A group $\mathbb{G}$ is called finite if $|\mathbb{G}|$ is finite. The number of elements of a finite group is called its order [72].*

Definition 7 *A group $\mathbb{G}$ is cyclic if there is $g \in \mathbb{G}$ such that every element $a \in \mathbb{G}$ can be written in the form of g^k for some $k \in Z$. That is $\mathbb{G} = \{g^i | i \geqslant 0\}$. We call such g a generator of $\mathbb{G}$ and write $\langle g \rangle = \mathbb{G}$ to indicate that g generates $\mathbb{G}$ [72].*

Definition 8 *Let $\mathbb{G}$ be a group and $a \in \mathbb{G}$. The order of a, denoted by $ord(a)$, is the smallest positive integer n such that $a^n = 1$, provided that such an integer exists. If such an n does not exist, then the order of a is defined as ∞ [72].*

Definition 9 *Let $(\mathbb{G}, *)$ be a group. We say that $(H, *)$ is a subgroup of $\mathbb{G}$ if $H \subseteq \mathbb{G}$ and $(H, *)$ is a group [72].*

2.2 Public Key Infrastructure

PKI (Public Key Infrastructure) is a system that provides authentic public keys (public keys signed by certification authority) to the applications. According to X.509 which is a Telecommunication Standardization Sector standard for public key infrastructure [55], the functions of a Certification Authority include:

– `Certification Authority`: The Certification Authority (CA) is the core of a PKI, which issues digital certificates for other parties. It authenticates the public key of an entity by signing it with its private key. The other parties can verify the signature and confirm the binding between the public key and its owner. A certificate revocation mechanism is employed to revoke a certificate in case of accidental events, such as key compromise or loss. The functions of a Certification Authority include:

  1) Issuing certificate for users.

2) Issuing cross certificates for other CAs.

3) Revoking a certificate in case of key compromise or loss.

4) Updating the Key Revocation List (CRL).

– `Registration Authority`: The Registration Authority (RA) is a component of a PKI which verifies an entity's request for a digital certificate and tells the Certificate Authorities to issue it. The functions of Registration Authority include:

1) Verifying the identity of an entity.

2) Verifying the possession of the private key.

3) Requesting certificate revocation for a certificate issued by CA.

– `Directory Server`: The Directory Server is a component of PKI which stores the certificates and the CRL list. It could be a system or a collection of distributed systems. Normally it does not need to be trusted. A Directory Server has the following functions:

1) Holding certificates or CRL.

2) Allowing an entity to download a certificate or CRL.

In large-scale deployments, there could be more than one CA. Each CA has one or more RAs and can publish data in one or more Directory Servers. In such a situation, a hierarchy model is usually employed, in which the upper level CA delegates trust to subordinate CAs. For example, suppose Alice and Bob belong to different CAs. A trust can be established between Alice and Bob if Bob's certificate also includes his CA's public key signed by Alice's CA. However, this kind of cross-certification may cause quite long certificate chains.

2.3 Identity-Based Cryptosystems

Current Public Key Infrastructures involve complex constructions of Certificate Authorities and consequently involves expensive communication and computation costs for certificate verification. To solve this problem, Shamir introduced an innovative concept called Identity-Based Cryptosystems [93]. An Identity-Based system allows any party to generate a public key from a well-known identity, such as an email address or phone number. The corresponding private key is generated by a trusted third party

called the Private Key Generator (PKG). To authorise the use of identity, the PKG uses a master private key to generate the private key for the chosen identity. Identity-Based Cryptosystems can be used for encryption schemes as well as signature schemes. In Identity-Based encryption schemes, when Alice wants to send an encrypted message to Bob, she simply encrypts the message with Bob's email address. Since Alice knows that this mail address belongs to Bob, there is no need to obtain a certificate for the public key. On receiving the message, Bob can authenticate himself to the PKG and decrypt the message with the corresponding private key [12]. In Identity-Based signature schemes, Alice and Bob can verify each other's signatures without exchanging private or public keys. When Alice wants to send a message to Bob, she signs it with her secret key, encrypts the result with Bob's name and adds her name to the message. When Bob receives the message, he can decrypt with his secret key and verifies the signature by using Alice's name as a verification key [93]. The most efficient identity-based signature schemes are currently based on bilinear pairings, such as the Weil or Tate pairings [12]. A standard Identity-Based Cryptosystem for signature consists of the following algorithms:

- `Setup`: It is a probabilistic algorithm which takes a system parameter ℓ as input, and outputs common parameters cp and a master secret key MSK.
- `Extract`: It is a probabilistic algorithm which takes a master secret key MSK, an arbitrary string ID and common parameters cp as input, and outputs a secret key SK.
- `Sign`: It is a probabilistic algorithm which takes a message $m \in \mathcal{M}$, a signing secret key SK and common parameters cp as input, and outputs a signature σ.
- `Verify`: It is a deterministic algorithm which takes a message m, a signature σ, an arbitrary string ID and common parameters cp as input, and outputs either `Accept` or `Reject`.

2.4 Digital Signature Schemes

2.4.1 Basics of Digital Signature Schemes

A *Digital signature* is a cryptographic primitive which provides authentication. It allows someone to sign a message so that everyone can verify the authenticity of the

signature but no one can forge the signature on a new message. The concept of digital signatures was first introduced by Diffie and Hellman [33]. Since then many signature schemes have been proposed, for example, Rivest, Shamir and Adleman proposed a signature scheme based on RSA [88]. The other digital signature schemes include [92, 68, 82, 74]. A standard digital signature scheme consists of the following algorithms:

- `Key Generation`: It is a probabilistic algorithm that takes a system parameter ℓ as input, and outputs a public key PK and the corresponding secret key SK for a signer.
- `Sign`: It is a probabilistic algorithm that takes a message $m \in \mathcal{M}$ and a secret key SK as input, and outputs a signature σ.
- `Verify`: It is a deterministic algorithm that takes a message m, a signature σ and the public key PK as input, and outputs either `Accept` or `Reject`.

A digital signature scheme should meet the following requirements:

1. σ is a valid signature if and only if for all messages m, for all key pairs (SK, PK) output by `Key Generation`, it holds that `Verify`(m, `Sign`(m, SK), PK) = `Accept`.
2. It is computationally infeasible for any entity other than the signer $\mathcal{S}$ to find, for any message m, a signature σ such that `Verify`(m, σ, PK) = `Accept`.

Here we use an RSA signature scheme as an example; the algorithm is as follows [88]:

- `Key Generation`: On input of a security parameter ℓ, the signer generates $n = pq$ and $\varphi(n) = (p-1)(q-1)$, where $p = 2p' + 1$, $q = 2q' + 1$ and p, q, p', q' are all primes. It also chooses an integer e such that $1 < e < \varphi(n)$ and gcd(e, $\varphi(n)$) = 1. Finally it computes the secret key d such that $1 < d < \varphi(n)$ and $ed \equiv 1 \pmod{\varphi(n)}$. It publishes (e, n) and keeps d secret.
- `Sign`: To sign a message m, the signer computes $\sigma = h(m)^d \pmod{n}$, where h is a hash function. It sends σ and m to the verifier.
- `Verify`: To verify a signature σ, the verifier checks $h(m) \stackrel{?}{=} \sigma^e$. It outputs `Accept` if the above equation holds.

2.4.2 Group Signatures

A variation of basic signature schemes, known as *group signature*, was proposed by Chaum and van Heijst [27]. A group signature is a signature scheme that allows any member of a group to digitally sign a document which a verifier can confirm came from the group, but does not know which individual in the group signed the document. A group signature schemes usually consists of many signers and a single group manager. In case of dispute, the group manager is able to find out who actually produced the signature. Since its appearance, various group signature schemes have been proposed, such as [78, 18, 32]. However, one disadvantage of such a signature is that the length of the signatures is related to the size of the group. This restricts the application of such schemes to small groups. Camenisch and Stadler [20] presented a new group signature scheme which remains practical even for large groups. The first efficient and provably secure group signature scheme was proposed in [6]. An extension of the group signature, called group blind signature, was first introduced by Lysyanskaya and Ramzan [69]. Group blind signatures incorporate the properties of both blind signatures and group signatures. They can be used in many of the settings where blind signatures are used. Particularly, they can be used to design privacy-protecting electronic payment systems. The other application areas of the group signature include anonymous credentials, voting and bidding. The group signature usually consists of the following algorithms [9]:

- `Key Generation`: It is a probabilistic algorithm that takes a system parameter ℓ as input, and outputs a tuple $(GPK, GMSK, GSK)$, where GPK is the group public key, $GMSK$ is the group manager's secret key, and GSK is an n-vector of keys with $GSK[i]$ being a secret key for user $i \in [n]$.

- `Sign`: It is a probabilistic algorithm that takes a message $m \in \mathcal{M}$ and a secret key $GSK[i]$ as input, and outputs a signature σ.

- `Verify`: It is a deterministic algorithm that takes a message m, a signature σ and the group public key GPK as input, and outputs either `Accept` or `Reject`.

- `Open`: It is a deterministic algorithm that takes the group manager's secret key $GMSK$, a message m and a signature σ as input, and outputs an identity i or the symbol $\perp$ to indicate failure.

A group signature should also meet the following requirements:

1. σ is a valid signature if and only if for all messages m, for all key pairs (GSK, GPK) output by `Key Generation`, it holds that `Verify`(m, `Sign`(m, $GSK[i]$), GPK) = `Accept`.

2. It is computationally infeasible for any entity other than the group member $U_i \in \mathcal{U}$ to find, for any message $m \in \mathcal{M}$, a group signature σ such that `Verify`(m, `Sign`(m, $GSK[i]$), GPK) = `Accept`.

3. It is computationally infeasible for any entity to find, for any group signature σ, the member $U_i \in \mathcal{U}$ who produce the signature.

4. The identity of the signer $\mathcal{S}$ can be revealed if and only if the group manager runs `Open`(m, σ, $GMSK$).

2.4.3 Ring Signature

A variation of group signature schemes, known as *ring signature*, was first introduced by Rivest, Shamir and Tauman [94]. A ring signature can be regarded as a group signature without a group manager. This signature can be used to convince any verifier that one of the participants in the group has signed the message on behalf of the group. However, the verifier cannot identify who has actually signed the message. It is different from a group signature in that there is no group manager who can reveal the identity of signer even in case of a dispute. Therefore, this kind of signature scheme provides perfect signer ambiguity. Since then, many ring signature schemes have been proposed, such as [100, 21, 61, 89]. In [3], a method to construct a ring signature from different types of public keys, such as those for integer factoring based schemes and discrete log based schemes, was proposed. It is more efficient than that in [94]. A ring signature usually consists of the following algorithms [94]:

– `Key Generation`: It is a probabilistic algorithm that takes a system parameter ℓ as input, and outputs a secret key SK_i and the corresponding public key PK_i for a member i.

– `Sign`: It is a probabilistic algorithm that takes a message $m \in \mathcal{M}$, the public keys PK_1, PK_2, ..., PK_n of the n ring members and the secret key SK_s of the signer as input, and outputs a ring signature σ.

– `Verify`: It is a deterministic algorithm that takes a message m, a ring signature σ and the public keys PK_1, ..., PK_n as input, and outputs either `Accept` or

`Reject`.

A ring signature should meet the following requirements:

1. σ is a valid signature if and only if for all messages m, for all key pairs (SK_i, PK_i) output by `Key Generation`, it holds that `Verify`(m, `Sign`(m, SK_s, PK_1, ..., PK_n), PK_1, ..., PK_n) = `Accept`.

2. It is computationally infeasible for any entity other than the group member $U_i \in \mathcal{U}$ to find, for any message $m \in \mathcal{M}$, a ring signature σ such that `Verify`(m, `Sign`(m, SK_s, PK_1, ..., PK_n), PK_1, ..., PK_n) = `Accept`.

3. It is computationally infeasible for any entity to find, for any ring signature σ, a member $U_s \in \mathcal{U}$ who produce the signature.

4. A ring signature σ is setup free. It can be produced without the participation of other ring members $U_i \neq U_s$, where U_s is the actual signer of the ring.

An extension of the ring signature, called a threshold ring signature, was proposed by Bresson, Stern and Szydlo [15]. Their scheme is based on threshold cryptosystems [36] which allow n parties to share the ability to perform a cryptographic operation. In a threshold ring signature, the generation of signatures for a group of n members requires the involvement of at least d members and the signature reveals nothing about the signers. So this kind of scheme could effectively prove that a certain minimum number of members in group have collaborated to produce the signature while the identities of members remain hidden. Other threshold ring signature schemes were proposed in [101, 65].

While ring signatures protect the identity of the signer, the blind signatures protect the content of the message. Chan, Fung, Liu and Wei [22] combined the notion of ring signature and blind signature to construct the first blind ring signature. Their construction can be based on any well known blind signature scheme. The blindness of the blind ring signature depends on the blindness in the underlying blind signature schemes.

Another extension of the ring signature is linkable ring signature, introduced by Liu [66]. A linkable ring signature allows anyone to determine if two or more ring signatures are signed by the same group member. However, if the group member only signed once with his private key, he can still have anonymity, as in conventional ring

signature schemes. Addressing the problem that early linkable ring signatures have large size, Tsang and Wei gave a short linkable ring signature scheme in [99].

The notion of deniable ring authentication was introduced by Naor [75]. In a deniable ring signature scheme, it is possible to convince a verifier that a member of an ad-hoc collection of participants is authenticating a message, without revealing which member it is. Moreover, the verifier cannot convince anyone else that the message is authenticated. This is done by showing that the verifier could have produced such a signature by himself, without any interaction with the signers. Later, Susilo and Mu proposed a non-interactive version which avoids inefficient implementation of an anonymous channel [90]. They also extended their scheme to provide a non-interactive deniable ring for a threshold ring authentication scheme. In this scheme, the signer in the ring can sign a message and convince a group of verifiers of this fact, but the verifiers cannot convince any third party of the authenticity of this message. That is because a collusion of t verifiers can always create a valid message-signature pair that will also pass the verification stage.

In a traditional PKI, the public key is usually generated in a random way such that it is unrelated to the identity of the user. Therefore, a trusted authority is usually employed to ensure the relationship between the cryptographic keys and the user. As a result, the verifier needs to obtain a certificate of the signer's public key and checks the validity of the certificate before it verifies the signature. However, in a ring signature scheme, this mechanism is rather inefficient. To authenticate the signer, a verifier would need to verify all the public keys of the group. Furthermore, the anonymity of signer may be compromised if the all other certificates used are invalid. To solve the problem, Zhang and Kim [104] proposed the concept of *identity-based ring signatures*. In an identity-based ring signature scheme, the public key of each user can be easily obtained from a string corresponding to this user's identity, such as email address. Therefore, the certificate of public key is not required. A more efficient construction was proposed by Lin and Wu in 2003 [67] followed by several other schemes such as [50, 5].

2.4.4 Designated Verifier Signature

A special digital signature called an *undeniable signature* was proposed by Chaum and Van Antwerpen in 1989 [28]. In this scheme, a certificate is only verifiable with the signer's consent by an interactive proof. However, it was known that this type

of signature scheme has some drawbacks due to blackmail and mafia attack [34, 35]. To overcome this problem, another special digital signature called *designated verifier signature* was proposed in [62], where the proof is non-interactive. This scheme is known to be the first non-interactive version of [26]. With a designated verifier signature, only a designated person can verify the signature. When a designated verifier signature is presented to a chosen verifier, he can come to the following two conclusions: Either the signature is produced by the signer, or the signer knows his private key. Since the verifier knows that he did not produce the signature and the signer does not know his private key, he is convinced of the validity of the signature. However, if the verifier releases the signature to a third party, he can not tell whether the signer or the verifier signed the message. Thus he will not be convinced even if the verifier agrees to reveal his private key. The applications of a designated verifier signature include copyright protection and document notarisation.

An extension of the designated verifier signature called *Universal Designated-Verifier Signatures* (UDVS) was recently proposed in [96]. A UDVS scheme can be used as a standard publicly-verifiable digital signature but with an additional functionality that allows any signature holder to designate the signature to any desired designated verifier, using the verifier's public key. The construction proposed in [96] is based on bilinear pairings, which is an extension of the Boneh, Lynn and Shacham's short signature scheme [12]. Recently, an interactive version of the UDVS scheme was proposed in [16], where the public key setup of the designated verifier is not required. A designated verifier signature consists of the following algorithms:

- `Setup`: It is a probabilistic algorithm that takes a system parameter ℓ as input, and outputs common parameters cp.
- `Signer Key Generation`: It is a probabilistic algorithm which takes public parameters as input and outputs a pair of signing keys (SK_s, PK_s).
- `Verifier Key Generation`: It is a probabilistic algorithm which takes common parameters cp as input and outputs a pair of verifying keys (SK_v, PK_v).
- `Sign`: This algorithm can be either probabilistic or deterministic. It takes a message $m \in \mathcal{M}$, a signing secret key SK_s, a verifying public key PK_v and common parameters cp as input, and outputs a designated verifier signature σ.
- `Verify`: It is a deterministic algorithm which takes a designated verifier signature σ, a message m, a signing public key PK_s, a verifying secret key SK_v and common

parameters *cp* as input, and outputs either `Accept` or `Reject`.

A designated verifier signature should meet the following requirements [95]:

1. σ is a valid signature if and only if for all messages m, for all key pairs (SK_s, PK_s) and (SK_v, PK_v) output by `Key Generation`, it holds that `Verify`(m, `Sign`(m, SK_s, PK_v), PK_s, SK_v) = `Accept`.

2. It is computationally infeasible for any entity, without the knowledge of the secret key SK_s or SK_v, to find a designated verifier signature σ such that `Verify`(m, `Sign`(m, SK_s, PK_v), PK_s, SK_v) = `Accept`.

3. It is computationally infeasible for any entity to find, for any message m and any designated verifier signature σ, who of the original signer or the designated verifier performed the `Sign` algorithm, even if one knows SK_s and SK_v.

2.5 Encryption Schemes

2.5.1 Basics of Encryption Scheme

An *encryption scheme* is a cryptographic primitive that allows someone to send data to someone else in a confidential way. There are two main kinds of encryption schemes: symmetric-key based and public-key based.

In a symmetric-key encryption scheme, the encryption and decryption keys are identical or simply related. This type can be further divided into stream ciphers and block ciphers. The advantages of such encryption schemes are short key size and less intensive computation. However, the key to be kept secret at both communication ends and a secure symmetric-key encryption may require the key be changed frequently. This kind of encryption scheme is suitable for high rates of data throughput. Examples of popular symmetric-key algorithms include Twofish, Serpent, AES, 3DES and IDEA.

A public-key encryption scheme allows two parties to communicate securely without sharing a common secret key. The encryption and decryption keys are mathematically related. When Alice wishes to send a secret message to Bob, she looks up Bob's public key, uses it to encrypt the message and sends it off. Bob then uses his private key to decrypt the message. Only the private key need be kept secret and the private/public key pair could remain unchanged for a considerable period of time. However, the key

size is relatively larger and the encryption scheme is much slower than the previous one. Examples of popular public-key algorithms include RSA and Elgamal.

In this thesis, we mainly focus on the public-key encryption schemes. They usually consist of the following polynomial-time algorithms:

- `Key Generation`: It is a probabilistic algorithm that takes a system parameter ℓ as input, and outputs a pair of encryption keys (SK, PK).
- `Encrypt`: It is a probabilistic algorithm that takes a message $m \in \mathcal{M}$, a public key PK as input, and outputs a ciphertext $\mathcal{Z}$.
- `Decrypt`: It is a deterministic algorithm which takes ciphertext $\mathcal{Z}$, a secret key SK as input, and outputs a message m.

Here we use an RSA encryption scheme as an example; the algorithm is as follows [88]:

- `Key Generation`: On input of a security parameter ℓ, the key owner generates $n = pq$ and $\varphi(n) = (p-1)(q-1)$, where $p = 2p' + 1$, $q = 2q' + 1$ and p, q, p', q' are all primes. It also chooses an integer e such that $1 < e < \varphi(n)$ and gcd(e, $\varphi(n)$) = 1. Finally it computes the secret key d such that $1 < d < \varphi(n)$ and $ed \equiv 1 \pmod{\varphi(n)}$. It publishes ($e$, n) and keeps d secret.
- `Encrypt`: To encrypt a message m, the sender obtains the receiver's public key (e, n). It then computes $\mathcal{Z} = m^e \pmod{n}$ and sends it to the receiver.
- `Decrypt`: To decrypt a ciphertext $\mathcal{Z}$, the receiver computes $m = \mathcal{Z}^d \pmod{n}$ with his private key (d, n).

2.5.2 Broadcast Encryption

The concept of *broadcast encryption* was first introduced by Fiat and Naor [41]. In a broadcast encryption scheme, there is one sender and many receivers. The sender provides the receivers with prearranged keys and distributes data to a selected set of users. Applications of broadcast encryption include to PayTV and to file access control systems. A broadcast encryption scheme should meet the following requirements:

1. The subset of receivers $\mathcal{U}$ can be dynamically changed.
2. It is computationally infeasible for any entity other than the receivers $\mathcal{U}$ to find the message m.

3. It is computationally infeasible for any subset of k users to collude in such way that a non-member $U_i \notin \mathcal{U}$ can read the message without knowing one of the secret key SK_i. Here k is the number of members that must collude in order to read the message.

A broadcast encryption scheme consists of the following polynomial-time algorithms:

- `Key Generation`: It is a probabilistic algorithm that takes a system parameter ℓ as input, and outputs n private keys SK_1, SK_2, ..., SK_n, a public key PK and common parameters cp.
- `Encrypt`: It is a probabilistic algorithm that takes a subset of receivers $\mathcal{U}$, a message $m \in \mathcal{M}$ and a public key PK as input, and outputs a ciphertext $\mathcal{Z}$.
- `Decrypt`: It is a deterministic algorithm that takes a ciphetext $\mathcal{Z}$, the private key SK_i of receiver i and common parameters cp as input, and outputs a message m.

Here we use the broadcast encryption scheme proposed by Mu and Varadharajan [70] as an example. Their scheme uses the polynomial function:

$$f(x) = \prod_{i=1}^{n}(x - x_i) \equiv \sum_{i=0}^{n} a_i x^i \ (mod\ q)$$

where a_i denotes the coefficient of x_i after the expansion of $f(x)$:

$$\begin{aligned} a_0 &= \prod_{i=1}^{n} x_i \ (mod\ q) \\ a_1 &= \sum_{i=1}^{n}\prod_{j\neq i}^{n}(-x_j) \ (mod\ q) \\ &\vdots \\ a_{n-1} &= \sum_{i=1}^{n}(-x_i) \ (mod\ q) \\ a_n &= 1 \ (mod\ q) \end{aligned}$$

Note that $\sum_{i=0}^{n} a_i x^i \equiv 0 \pmod{q}$. We could use this property to construct an encryption scheme:

- `Setup`: On input of a security parameter ℓ, the broadcaster randomly selects a generator $g \in \mathbb{Z}_q^*$. It also generates n distinct random numbers $x_i \in \mathbb{Z}_q$ and sets $g_i = g^{x_i} \pmod{q}$ for $i = 1, ..., n$.

– `Encrypt`: The broadcaster picks a random number $r \in \mathbb{Z}_q^*$ and encrypts the message $m \cdot g_0^r$. The ciphertext is $\mathcal{Z} = \{m \cdot g_0^r, g_1^r, ..., g_n^r\}$.

– `Decrypt`: On receiving the ciphertext, a receiver i decrypts the message with its private key x_i.

$$\begin{aligned} m \cdot g_0^r \cdot \prod_{j=1}^{n} g_j^{r x_i^j} &= m \cdot \prod_{j=0}^{n} g_j^{r x_i^j} \ (mod\ q) \\ &= m \cdot g^{\sum_{j=0}^{n} a_j x_i^j \cdot r} \ (mod\ q) \\ &= m \cdot g^{f(x_i) \cdot r} \ (mod\ q) \\ &= m \cdot 1^r \ (mod\ q) \\ &= m \end{aligned}$$

2.6 Commitment Schemes

A *commitment scheme* is a two party protocol between a committer and a receiver. It is an important building blocks for most cryptographic protocols, such as zero-knowledge proofs. In general, a commitment scheme does not need to be interactive. To commit to a value x, the committer generates a random r, computes the commitment R and sends it to the receiver. In the `Open` phase, the receiver verifies the commitment value R with r. A commitment scheme has the following requirements [58]:

– `Hiding`: Even if the adversary $\mathcal{A}$ chooses the input distribution for the value x to which the Committer commits, he learns no more information about this value than what is computable from the input distribution itself.

– `Binding`: No probabilistic polynomial-time adversary $\mathcal{A}$ can open a commitment in two different ways.

A commitment scheme usually consists of the following algorithms [58]:

– `Setup`: It is a probabilistic algorithm that on input of a system parameter ℓ, outputs common parameters *cp*.

– `Commit`: It is a probabilistic algorithm that on input of a secret value x and the user's public key PK, outputs a commitment R.

– `Open`: It is a deterministic algorithm that on input of a commitment value R, outputs the value x.

Various commitment schemes have been proposed since the appearance of the first. One is the unconditional hiding commitment for which the hiding property described above holds even against a computationally unbounded recipient. Another is the non-malleable and mutually independent commitments in which multiple players are committers and recipients at the same time. Among them the most widely used is the trapdoor commitment.

A trapdoor commitment scheme, also called a chameleon commitment, is a commitment scheme with an additional trapdoor key [13]. Knowledge of the trapdoor allows the sender to open the commitment in more than one way, which is often referred as the equivocality property. On the other hand, without knowledge of the trapdoor, equivocality remains computationally infeasible. When the commitments computed by means of a trapdoor are distributed exactly as real commitments then the trapdoor commitment scheme is unconditionally hiding. Instead, the equivocality property allows only computationally binding trapdoor commitment schemes.

2.7 Zero Knowledge Proofs

A Zero Knowledge Proof is an interactive method with which one party can prove the truth of some statement to another without revealing anything other than the truth of the statement [48]. The idea of the zero-knowledge proof was motivated by the authentication system where one party wants to prove his identity to another via some secret information, but does not want the other party to learn anything about this secret. A zero knowledge proof should satisfy the following requirements [48]:

1. Completeness: If the statement is true, the honest verifier will be convinced of this fact by an honest prover.

2. Soundness: If the statement is false, no cheating prover can convince the verifier that it is true.

3. Zero-Knowledge: If the statement is true, no cheating verifier learns anything other than this fact.

2.7.1 Proof of Knowledge

A Zero Knowledge Proof of Knowledge is a protocol which allows one party to prove to another his knowledge of some secret information without leaking any information

about the secret. It was first defined by Feige, Fiat and Shamir [40] and further refined by Bellare and Boldreich [11]. Here is an example: Suppose Alice wants to convince Bob of her knowledge of a RSA secret key d corresponding to a public key e. However, she does not want to tell Bob the key or to decrypt any message for Bob. So she proves her knowledge of the corresponding secret key as follows [11]:

1. Alice and Bob agree on a random k and m such that $km \equiv e \pmod{n}$ Note. Both k and m are greater than three. k is generated by a flip-coin protocol.
2. Alice and Bob generate a random ciphertext $\mathcal{Z}$.
3. Alice uses the private key d to decrypt $M = \mathcal{Z}^d \pmod{n}$ and $X = M^k \pmod{n}$. She then sends X to Bob.
4. Bob verifies $X^m \pmod{n} \stackrel{?}{=} \mathcal{Z}$, if it does, then he believes Alice.

2.8 Signature of Knowledge

The first signature based on proof of knowledge (SPK) was proposed in [19, **?**]. We will use the following definition of SPK from [19].

Let q be a large prime and $p = 2q+1$ also be a prime. Let $\mathbb{G}$ be a finite cyclic group of prime order p. Let g be a generator of $\mathbb{Z}_p^*$ such that computing discrete logarithms of any group elements (apart from the identity element) with respect to one of the generators is infeasible. Let $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$ denote a strong collision-resistant hash function.

Definition 10 *A pair* $(c, s) \in \{0,1\}^\ell \times \mathbb{Z}_q$ *satisfying* $c = H(g||y||g^s y^c||m)$ *is a signature based on proof of knowledge of the discrete logarithm of a group element* y *to the base* g *of the message* $m \in \{0,1\}^*$ *and is denoted by* $SPK\{\alpha : y = g^\alpha\}(m)$.

An $SPK\{\alpha : y = g^\alpha\}(m)$ can be computed if and only if the value (secret key) $\alpha = \log_g(y)$ is known. This is also known as a non-interactive proof of the knowledge α.

Definition 11 *A pair* (c, s) *satisfying* $c = H(U||\ P||S||Q||\ sU + cS||sP + cQ||m)$ *is a signature of equality of the elliptic curve discrete logarithm problem of the group element* S *with respect to the base* U *and the discrete logarithm of the group element* Q *with respect to the base* P *for the message* m. *It is denoted by* $ECSPKEQ\{\alpha : Q = \alpha P \wedge S = \alpha U\}(m)$.

2.9 Security Requirements for Digital Signatures

Attacks

Usually, a digital signature scheme is said to be secure if it is against certain types of attack. There are two types of attacks in general [49]: *Key-Only Attacks* and *Message Attack*. The *Key-Only Attack* is also referred to as a *No-Message Attack* in which the attacker only knows the public key of the signer. The *Message Attack* is more severe, because it allows the attacker to access a list of message-signature pairs. It can be further divided into four subclasses [80]:

- `Known Message Attack`: Attacker $\mathcal{A}$ has access to the signatures $\{\sigma_i\}_{i\in[1,n]}$ for a set of messages $\{m_i\}_{i\in[1,n]}$ with the restriction that he has not chosen them.
- `Generic Chosen Message Attack`: Attacker $\mathcal{A}$ has access to the signatures $\{\sigma_i\}_{i\in[1,n]}$ for a set of messages $\{m_i\}_{i\in[1,n]}$ chosen by him. However, this choice must be made before accessing the public key of the signer $\mathcal{S}$. Since this attack is independent of $\mathcal{S}$'s public key, we refer to it as "generic".
- `Directed Chosen Message Attack`: Attacker $\mathcal{A}$ has access to the signatures $\{\sigma_i\}_{i\in[1,n]}$ for a set of messages $\{m_i\}_{i\in[1,n]}$ chosen by him. However, this choice must be made before observing any signatures of the signer $\mathcal{S}$. Since this attack is against a particular signer $\mathcal{S}$, we refer to it as "direct chosen message attack".
- `Adaptive Chosen Message Attack`: Attacker $\mathcal{A}$ can choose arbitrary message m and request the signature σ from the signer $\mathcal{S}$. We call it "adaptive" because attacker $\mathcal{A}$ can adapt his queries according to the previously obtained signatures.

Forgeries

An attack is said to be successful if the attacker $\mathcal{A}$ can forge a valid signature. The attacks can be classified [81] according to the severity of attack:

- `Total Break`: This is the most serious attack, in which attacker $\mathcal{A}$ can compute the signer $\mathcal{S}$'s secret key.
- `Universal Forgery`: Attacker $\mathcal{A}$ can construct an efficient signing algorithm functionally equivalent to $\mathcal{S}$'s signing algorithm.

- `Existential Forgery`: Attacker $\mathcal{A}$ can forge a valid signature-message pair. This kind of forgery does not allow the attacker has control over the message to be signed. This kind of attack is usually considered to be least severe.

Security Requirements

The notion of *Existential Unforgeability Against Chosen Message Attack* for digital signatures was first introduced by Goldwasser-Micali-Rivest in 1984 [47]. Since then, many researchers have used this as a standard for measuring security for their signature schemes. Generally speaking, a standard digital signature scheme should meet the following requirements [79]:

- `Correctness`: A digital signature scheme is correct if, given a signing key SK and corresponding verification key PK, for any message $m \in \mathcal{M}$, the output of signing algorithm `Sign` will always be accepted by the verification algorithm `Verify`. Formally, for any m:

$$Pr[(SK, PK) \leftarrow \mathbf{KeyGen}(\ell); \sigma \leftarrow \mathbf{Sign}(m, SK) \wedge \mathbf{Verify}(m, \sigma, PK)] = 1$$

- `Unforgeability`: A digital signature scheme is *existentially unforgeable against chosen message attack* if no adversary $\mathcal{A}$, who has access to a signing oracle $\mathcal{SO}$ (which outputs the valid signature for the message m), can produce a new message-signature pair (m, σ) with non-negligible probability. Formally, for any polynomial-time oracle $\mathcal{A}^{SO_{SK}}(\cdot)$,

$$\begin{aligned} Adv_{\mathcal{A}}(\ell) \;\; = \;\; & Pr[(SK, PK) \leftarrow KeyGen(\ell); \sigma \leftarrow \mathcal{A}^{SO_{SK}(\cdot)}(\ell) : \\ & Verify(m, \sigma, PK) = 1] \leqslant \epsilon \end{aligned}$$

  where ϵ is a negligible function.

Definition 12 *A signature scheme is secure if an existential forgery is computationally infeasible under an adaptive chosen message attack.*

2.10 Security Proofs

One important task in designing a cryptographic scheme is to provide a security proof. That is because without formal proof, the scheme may be vulnerable to some unseen weakness. The idea of provable security was introduced by Goldwasser and Micali [46] and formalised by Mihir Bellare [17]. It consists of the following steps:

1. Define the security goals, that is, what security requirements we want to achieve.
2. Define a security model and set up an experiment between the attacker and simulated environment under that model.
3. Select a hard mathematical problem as an atomic primitive.
4. Reduce the experiment to that atomic primitive and show that the only way to defeat the scheme is to solve the underlying atomic primitive.
5. Conclude that since the hard problem is unsolvable, the success of the attacker would lead to a contradiction, and therefore the scheme is secure.

2.11 Cryptographic Tools

2.11.1 Hash Functions

A hash function is an efficient computable function which takes a variable length string m as input, and outputs a fixed length string h (that is , $h = H(m)$). The resulting string is called the hash value, which is relatively short. The hash functions are used for data integrity in conjunction with digital signature schemes. There are various hash functions, such as MD5 [86] and SHA-1 [76]. In general they should meet the following requirements:

- `One-wayness`: A hash function should be one way. That is, given any value m, it is easy to compute $H(m)$. But given a hash value h, it is computationally infeasible to find some input m such that $H(m) = h$.
- `Collision-free`: Given a message m, it is computationally infeasible to find a message m' not equal to m such that $H(m) = H(m')$.
- `Strong Collision-free`: It is computationally infeasible to find any two messages m and m' such that $H(m) = H(m')$.

2.11.2 Bilinear Pairing

Bilinear pairing is a cryptography tool that has been used frequently in recent cryptography applications [12, 104, 105, 103]. Here we define the basic concept of bilinear pairing.

Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic additive groups generated by P_1, P_2, respectively, whose order are prime q. Let $\mathbb{G}_M$ be a cyclic multiplicative group with the same order q. We assume there is an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(P_2) = P_1$. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_M$ be a bilinear mapping with the following properties:

1. *Bilinearity*: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b, \in \mathbb{Z}_q$.
2. *Non-degeneracy*: There exists $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ such that $\hat{e}(P, Q) \neq 1$.
3. *Computability*: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$.

For simplicity, hereafter, we set $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$. And aP notation implies P^a in G_1, bQ notation implies Q^b in G_1.

A bilinear pairing instance generator is defined as a probabilistic polynomial time algorithm $\mathcal{IG}$ that takes as input a security parameter ℓ and returns a uniformly random tuple $param = (p, \mathbb{G}_1, \mathbb{G}_M, \hat{e}, P)$ of bilinear parameters, including a prime number p of size ℓ, a cyclic additive group $\mathbb{G}_1$ of order q, a multiplicative group $\mathbb{G}_M$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_M$ and a generator P of $\mathbb{G}_1$. For a group $\mathbb{G}$ of prime order, we denote the set $\mathbb{G}^* = \mathbb{G} \setminus \{\mathcal{O}\}$ where $\mathcal{O}$ is the identity element of the group.

2.11.3 Time Stamp

A timestamp is a digital proof which provides timeliness and uniqueness guarantees. It is usually used to prevent message replay attacks. Let Alice be the sender and Bob be the receiver. A timestamp works as follows [73]:

1. Alice obtains the current time code from her local clock and binds it to the message. Such binding could be encrypted in the message together with the time code. She then sends the message to Bob.
2. Upon receiving the message, Bob also obtains a time code from his local clock and substracts the time code received. Bob is convinced that the received message is valid if no message with the same timestamp has been received before from Alice.

However, the use of a timestamp requires that the local clocks at both ends be synchronised and secure. This greatly restricts the applicability of such a cryptography

tool. For example, in a distributed environment, it is hard to prevent the adversary from modifying local clocks. Another drawback of such a protocol is the potential need for large storage spaces. That is because the receiver needs to compare the current timestamp with the old ones, therefore a list of used timestamps must be maintained.

2.11.4 Intractable Problems

In this section, we introduce some intractable problems that will be used in the following chapters. A problem is intractable if there is no deterministic algorithm that can solve the problem within a polynomial time. Our schemes base their security on the intractability of the following mathematical problems:

Definition 13 Discrete Logarithm (DL) Problem.
Given a finite cyclic group $\mathbb{G} = g^1, g^2, ..., g^n$ *and a randomly picked value* $y \in \mathbb{G}$*, find the integer* x *such that* $y = g^x$*.*

For the DL problem to be hard, the security parameter n must be large enough so that there is no known algorithm for efficiently computing the integer x.

Definition 14 Bilinear Diffie-Hellman (BDH) Problem.
Given a randomly chosen $P \in \mathbb{G}_1$*, as well as* aP, bP *and* cP *(for unknown randomly chosen* $a, b, c \in \mathbb{Z}_q$*), compute* $\hat{e}(P, P)^{abc}$*.*

For the BDH problem to be hard, $\mathbb{G}_1$ and $\mathbb{G}_M$ must be chosen so that there is no known algorithm for efficiently solving the Diffie-Hellman problem in either $\mathbb{G}_1$ or $\mathbb{G}_M$. We note that if the BDH problem is hard for a pairing $\hat{e}$, then it follows that $\hat{e}$ is non-degenerate. That is if $\hat{e}(x, y) = 0$ for all $x \in \mathbb{G}_1$, then $y = 0$ and if $\hat{e}(x, y) = 0$ for all $y \in \mathbb{G}_1$, then $x = 0$.

Definition 15 Bilinear Diffie-Hellman Assumption.
If $\mathcal{IG}$ *is a BDH parameter generator, the advantage* $\mathtt{Adv}_{\mathcal{IG}}(\mathcal{A})$ *that an algorithm* $\mathcal{A}$ *has in solving the BDH problem is the probability that the algorithm* $\mathcal{A}$ *outputs* $\hat{e}(P, P)^{abc}$ *on inputs* $\mathbb{G}_1, \mathbb{G}_M, \hat{e}, P, aP, bP, cP$*, where* $(\mathbb{G}_1, \mathbb{G}_M, \hat{e})$ *is the output of* $\mathcal{IG}$ *for sufficiently large security parameter* ℓ*,* P *is a random generator of* $\mathbb{G}_1$ *and* a, b, c *are random elements of* $\mathbb{Z}_q$*. The BDH assumption is that* $\mathtt{Adv}_{\mathcal{IG}}(\mathcal{A})$ *is negligible for all efficient algorithms* $\mathcal{A}$*.*

Definition 16 Computational Diffie-Hellman (CDH) Problem.
Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP, for some $a, b \in \mathbb{Z}_q^*$, compute abP.

The CDH problem is a variant of the DL problem. For the CDH problem to be hard, there must be no known efficient algorithm such that given aP or bP, it can calculate the value of a or b, hence to calculate $(aP)^b$ or $(bP)^a$.

Definition 17 Computational Diffie-Hellman (CDH) Assumption.
If $\mathcal{IG}$ is a BDH parameter generator, the advantage $\mathtt{Adv}_{\mathcal{IG}}(\mathcal{A})$ that an algorithm $\mathcal{A}$ has in solving the CDH problem is the probability that the algorithm $\mathcal{A}$ outputs abP on inputs $\mathbb{G}_1, P, aP, bP$, where $(\mathbb{G}_1, \mathbb{G}_M, \hat{e})$ is the output of $\mathcal{IG}$ for sufficiently large security parameter ℓ, P is a random generator of $\mathbb{G}_1$ and a, b are random elements of $\mathbb{Z}_q$. The CDH assumption is that $\mathtt{Adv}_{\mathcal{IG}}(\mathcal{A})$ is negligible for all efficient algorithms $\mathcal{A}$.

2.11.5 Random Oracle Model

A random oracle model is a popular methodology used in security proofs when there is no real function that provides the mathematical properties necessary to satisfy the proof of security. The concept of the random oracle was built on the work of Goldreich, Goldwasser and Micali [44, 45] and later formalised by Bellare and Rogaway [10]. Thereafter, many researchers have used the random oracle model in their security proof as opposed to a generic model. In the random oracle model, we assume hash functions are random functions and are publicly accessible by all parties. A random oracle, $\mathcal{O}$, is an object to instantiate all hash functions in the model and reply to all queries from the parties. A polynomial time algorithm cannot distinguish between the query reply from a real world and the random oracle simulated by a function. A random oracle $\mathcal{O}$ should meet the following requirements:

- $\mathcal{O}$ is assumed to be a random function such that given an input, no party can guess the output with non-negligible probability.
- $\mathcal{O}$ is a one-way function. Given an output, it is difficult to determine its preimage.
- $\mathcal{O} : \{0,1\}^* \rightarrow \{0,1\}^\infty$. Note $\{0,1\}^*$ denotes the space of finite binary strings and $\{0,1\}^\infty$ denotes the space of infinite ones.
- $\mathcal{O}$ is collision resistant so that given x,y where $x \neq y$, $\mathcal{O}(x) = \mathcal{O}(y)$ with negligible probability.

– Given the same inputs x and y, where $x = y$, $\mathcal{O}(x) = \mathcal{O}(y)$.

– All parties in the random oracle model must query $\mathcal{O}$ to obtain random values. They cannot distinguish the values generated by $\mathcal{O}$ from the real hash function.

However, in reality there exists no hash function which behaves purely randomly. Therefore a scheme that is secure in the random oracle model does not mean that it is also secure in real life. Certain very artificial protocols are proven secure in the random oracle model, but trivially insecure when any real hash function is substituted for the random oracle. So the meaning of a security proof in the random oracle model is still unclear. Despite its impracticalities, the random oracle model is useful for yielding an efficient solution to prove the security of a scheme. It is better than no proof at all.

2.12 Summary

In this chapter, we reviewed some background knowledge that will be used in the following chapters. First, we introduced some mathematical background, such as number theory and group theory. Then we discussed with the concept of ID-based Cryptosystems which are used to construct our ID-based ring signature schemes. After that, we reviewed several related signature schemes and related encryption schemes. We also illustrated the idea of commitment schemes and zero knowledge proof. Furthermore, we introduced security related issues such as security modeling, security requirement and security proof. Finally, we briefly reviewed some cryptographic tools, such as hash function, bilinear pairings, and random oracle.

Chapter 3

User Privacy Protection with Ring Signatures

3.1 Introduction

In this chapter, we propose two privacy protection protocols with ring signature. First, we construct a simple and basic protocol between a user and a service provider. We then extend the protocol to allow two or more users to communicate with a single party. In both protocols, the service provider is able to send an acknowledgement back to the users. We also investigate possible applications of our protocols.

3.1.1 Motivation

With the user and number of online applications continuously increasing, there is an increasing concern about preserving privacy. This is due to existing technologies that allow service providers to easily acquire users' personal information, such as names, birth dates and addresses. Internet users are worried about the collection and misuse of such information. For this reason, there is an increasing demand that countermeasures be taken to enable users to gain control over the extent to which their personal information is available to others. However, because of security concerns on the Internet, authentication is an indispensable part of many Internet services. This makes most privacy protection technologies inapplicable.

In this chapter, we consider a privacy protection issue where certain kinds of authentication are required. Formally, we would like to have a protocol that satisfies the following. A user, A, can authenticate himself to a service provider B in such way that B is convinced of A's membership, but B can not tell which specific member A is. Also, B should be able to interact with A. We achieve our aim using ring signatures. A ring signature allows a signer in a ring to construct a signature such that the receiver

is assured that the signer is indeed a user in the ring, while the identification of the signer is indistinguishable to him.

3.1.2 Contributions of This Chapter

Considering the drawback of previous solutions reviewed in 1.2, we propose a new privacy protection protocol that makes it possible for users to authenticate themselves as a member while remaining anonymous. Our first protocol is derived from Susilo and Mu's scheme [71] and Liu's scheme [66]. We then extend it to a multi-user scenario. The merits of our protocol are as follows:

1. Users are able to control the amount of personal information they reveal on the Internet.

2. It does not compromise any security concerns. The service provider is still able to authenticate the user as a member. But he can not find out which specific member the user is.

3. The service provider is still able to interact with the user.

3.2 Definition of Our Scheme

3.2.1 System Model

In this section, we consider a situation where t users $\{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_t\} \in \mathcal{C}$ would like to communicate with a service provider $\mathcal{SP}$. Our protocol relies on a key generation centre $\mathcal{KGC}$. The $\mathcal{KGC}$'s role is to set up the necessary system parameters and distribute private keys, hence it is not necessary to stay online. Before the protocol takes place, both the users $\mathcal{C}$ and the service provider $\mathcal{SP}$ need to register with the $\mathcal{KGC}$. $\mathcal{KGC}$ issues a different private/public key pair for each of them. We assume this can be done in a secure way, such as in person. Our protocol also relies on anonymous networks [83, 23, 60, 52]. We assume that there exists an anonymous channel (such as Mix-net in Chapter One) between $\mathcal{C}$ and $\mathcal{SP}$ which provides protection against various attacks. Before users $\mathcal{C}$ interact with any specific service provider, they also need to register with that $\mathcal{SP}$. After registration, $\mathcal{SP}$ puts $\mathcal{C}_i$'s public key pk_i into a member list and updates the list on his public directory.

3.2.2 Definition of Our Ring Signature Scheme

The notion of ring signature was spelled out in Section 2.4.3. In this section, we present the definition of our ring signature scheme. Our ring signature incorporates a temporary public key in such way that the verifier is able to send an acknowledgement back to the signer. Moreover, in our scheme, the signer actually proves his knowledge of the private key that corresponds to this public key. Therefore, the originality of the receiver is guaranteed. Our ring signature scheme is defined as follows:

Definition 18 *R_n^t is a t-out-of-n ring signature scheme which consists of the following algorithms:*

- `R-Setup`*: It is a probabilistic algorithm that takes as input a security parameter ℓ, and outputs the common parameters cp. That is cp* $\leftarrow$**R-Setup**(ℓ).
- `R-KeyGen`*: It is a probabilistic algorithm that takes as input common parameters cp, and outputs a private/public key pair (sk_i, pk_i). That is (sk_i, pk_i)* $\leftarrow$**R-KeyGen**(cp). *We denote* **SK** *and* **PK** *the domains of the possible private keys and public keys respectively. When we say that a private key corresponds to a public key or vice versa, we mean that the private/public key pair is an output of* **KeyGen**.
- `R-Sign`*: It is a probabilistic algorithm that takes as input common parameters cp, a list of n public keys Ł, a subgroup of t signers' private key $\{sk_i\}_{i\in S_t}$ and a message $m \in \mathcal{M}$, and outputs a ring signature σ. That is $\sigma \leftarrow$* **R-Sign** $(cp, Ł, \{sk_i\}_{i\in S_t}, m)$. *Note, the list Ł contains the public keys of t signers and $n-t$ public keys of other members.*
- `R-Verify`*: It is a deterministic algorithm that takes as input common parameters cp, a ring signature σ, a list of public keys Ł and a message m, outputs 1 or 0 for* Accept *or* Reject *respectively. That is (1/0)* $\leftarrow$ **R-Verify** $(cp, \sigma, Ł, m)$. *If the algorithm outputs* Accept*, the message-signature pair (m, σ) is said to be valid.*

3.2.3 Definition of the Broadcast Encryption Scheme

Upon receiving the ring signature from the signer via unicast, the verifier needs to send an acknowledgement back to the signer. However, since the identity of signer is unknown, the verifier can not send the acknowledgement directly to the signer. Thus,

he needs to use a broadcast encryption scheme in such a way that only the signer can read the message. The concept of broadcast encryption was described in 2.5.2. In this section, we present the definition of the broadcast encryption scheme we use.

Definition 19 *ε is a broadcast encryption scheme that is composed of the following algorithms:*

- *ε-*`Setup`*: It is a probabilistic algorithm that takes as input a security parameter ℓ, and outputs definitions of the message space $\mathcal{M}_e$ and the ciphertext space $\mathcal{M}_c$. That is $(\mathcal{M}_e, \mathcal{M}_c) \leftarrow \varepsilon$-**Setup**$(\ell)$.*
- *ε-*`Encrypt`*: It is a probabilistic algorithm that takes as input the common parameters cp, a message $m \in \mathcal{M}_e$ and public keys $\{A_i\}_{i \in S_t}$, and outputs a ciphertext $\mathcal{Z}$ and an associated parameter b. That is $(\mathcal{Z}, b) \leftarrow \varepsilon$-**Encrypt** $(cp, m, \{A_i\}_{i \in S_t})$. Note that the common parameters cp here are the same as the ones generated by R-Setup algorithm and the public keys $\{A_i\}_{i \in S_t}$ are a list of t elements generated from R-Sign algorithm.*
- *ε-*`Decrypt`*: It is a deterministic algorithm that takes as input a common parameters cp, a ciphertext $\mathcal{Z}$, an associated parameter b and a private key sk_i corresponding to one of the public key A_i, and outputs a plaintext $m \in M_e$. That is $m \leftarrow \varepsilon$-**Decrypt** $(cp, \mathcal{Z}, b, sk_i)$.*

3.2.4 Cryptographic Requirement

Correctness

We require our R_n^t ring signature scheme to satisfy the following probability equation:

$$Pr[y \leftarrow \texttt{RSign}(cp, m, L, \{sk_i\}_{i \in S_t}) : \texttt{RVerify}(cp, L, m, y)] = 1$$

where

$$\begin{aligned} L &= \{pk_1, ..., pk_n\} \\ cp &\leftarrow \texttt{RSetup}(\ell) \\ (sk_i, pk_i) &\leftarrow \texttt{RKeyGen}(cp) \\ \sigma &\leftarrow \texttt{RSign}(cp, m, L, \{sk_i\}_{i \in S_t}) \\ 1 &\leftarrow \texttt{RVerify}(cp, L, m, \sigma) \end{aligned}$$

This equation means for a signature on any message m any subset of signers L and any secret key sk_i, the probability of **Accept** is always one.

Proof of Knowledge

In our scheme, the signer actually proves his knowledge of the private key corresponding to the public key A_i that he embedded in the signature. This is achieved by a zero knowledge proof of the equality of two keys: the signing key sk_s and A_i's corresponding private key. With Chaum and Pedersen's proof of equality of discrete logarithms [30], we can prove the equality of discrete logarithm of y_1 and y_2 to the bases of $g_1 \in \mathbb{Z}_q^*$ and $g_2 \in \mathbb{Z}_q^*$. The proof consists of the following steps:

1. The prover selects a random number $w \in \mathbb{Z}_q$ and computes $W_1 = g_1^w \pmod q$, $W_2 = g_2^w \pmod q$. It sends W_1 and W_2 to the verifier.

2. On receiving W_1 and W_2, the verifier selects a random $c \in \{0,1\}^*$ and sends it to the prover.

3. The prover computes $s = w - cx \pmod q$ and sends it to verifier. The verifier checks $W_1 \stackrel{?}{\equiv} g_1^s y_1^c \pmod q$ and $W_2 \stackrel{?}{\equiv} g_2^s y_2^c \pmod q$. If both equations hold, then the verifier is convinced that y_1 and y_2 share the same private key x.

Signer Ambiguity

For any unbounded adversary $\mathcal{A}$, any message $m \in \mathcal{M}$, any set of public keys L and any signature σ. Our ring signature scheme should satisfy the following probability equation:

$$Pr[A(\sigma) = sk_s] = 1/|L|$$

where

$$\begin{aligned} |L| &= n \\ L &= \{pk_1, ..., pk_n\} \\ cp &\leftarrow \texttt{RSetup}(\ell) \\ (sk_i, pk_i) &\leftarrow \texttt{RKeyGen}(cp) \\ \sigma &\leftarrow \texttt{RSign}(cp, m, L, sk_s) \end{aligned}$$

Unforgeability

In this section, we provide the formal definition of existential unforgeability of our ring signature scheme under adaptive chosen message attack (EU-ACMA). It is defined

using the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$(the holder of oracles who answers given that $\mathcal{A}$ queries HO or SO):

- Let $\mathcal{A}$ be the EU-ACMA adversary. At the start of the game, $\mathcal{C}$ provides the common parameter cp to $\mathcal{A}$, where $cp \leftarrow$ R-Setup(ℓ) and ℓ is the security parameter.
- Because $\mathcal{A}$ has access to all the public keys on the public directory, $\mathcal{C}$ also needs to provide all the public keys $\{pk_i\}_{i=1,\ldots,n}$ to $\mathcal{A}$.
- At any time, $\mathcal{A}$ can query the hash oracle $\mathcal{HO}$ for the hash value on any message of his choice up to q_H times. $\mathcal{C}$ will answer $\mathcal{A}$'s queries by providing the hash value h.
- At any time, $\mathcal{A}$ can query the signing oracle $\mathcal{SO}$ for the signature on any message m, any subset of public keys L of his choice up to q_S times. $\mathcal{C}$ will answer $\mathcal{A}$'s queries by providing the value σ where R-Verify$(cp, m, L, \sigma) = 1$.
- $\mathcal{C}$ will not answer any verification request because $\mathcal{A}$ can verify the signature himself.
- Eventually, $\mathcal{A}$ will output a valid signature σ^* for a message m^* and a subset of public keys L^* that has never been queried to the $\mathcal{SO}$ before.

The success probability of an adversary is defined by $Succ_A^{EU-ACMA}(\ell)$, where ℓ is negligible.

We say that our ring signature scheme has existential unforgeability against adaptive chosen message attack if the probability of success of any polynomially bounded adversary $\mathcal{A}$ in the above game is negligible. In other words,

$$Succ_A^{EU-ACPA}(\ell) \leq \epsilon$$

where ϵ is negligible. Therefore the advantage of $\mathcal{A}$ is always negligible.

3.3 Our Scheme

In this section, we present the implementation of our privacy protection protocols. First, we illustrate the preclusions and assumptions for our protocol. Then we construct a basic protocol that consists of only one user and one service provider. Finally, we

extend the ring signature scheme used in the first protocol to propose a protocol for multiple users.

3.3.1 Preclusion and Assumption

The following activities are assumed to have been done before the actual protocol starts:

1. $\mathcal{KGC}$ runs the **R-Setup** algorithm to initialise. On input of a security parameter ℓ, $\mathcal{KGC}$ selects a prime p and a generator $g \in Z_p^*$ of order $q|p-1$. It defines the message space $\mathcal{M} = \{0,1\}^*$ and signature space $\mathcal{S} = \mathbb{Z}_q$. Two cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \rightarrow Z_p^*$ also selected. Here we assume that $\mathcal{KGC}$ is honest and that it will not reveal any information about the registered users.

2. The user $\mathcal{C}_i$ registers with $\mathcal{KGC}$. $\mathcal{KGC}$ runs the **R-KeyGen** algorithm to generate a random number $x_i \in \mathbb{Z}_q$ as the private key and sets $y_i = g^{x_i} \pmod p$ as the public key. The private/public key pair is sent to C_i via a secure channel, which is a method or technique by which data can be transferred from one place or user to another without risk of interception or tampering.

3. The service provider $\mathcal{SP}$ registers with $\mathcal{KGC}$. $\mathcal{KGC}$ runs the **R-KeyGen** algorithm to generate a random number $x_{\mathcal{SP}} \in \mathbb{Z}_q$ as $\mathcal{SP}$'s private key and sets $y_{\mathcal{SP}} = g^{x_{\mathcal{SP}}} \pmod p$ as his public key. The private key is sent to $\mathcal{SP}$ via a secure channel and the public key is published on $\mathcal{KGC}$'s public directory.

4. $\mathcal{C}_i$ gets $\mathcal{SP}$'s public key $y_{\mathcal{SP}}$ from $\mathcal{KGC}$'s public directory. $\mathcal{C}_i$ believes $y_{\mathcal{SP}}$ to be authenticated since it is published by $\mathcal{KGC}$.

5. The user C_i registers with a specific service provider $\mathcal{SP}$ if necessary. C_i provides his public key to $\mathcal{SP}$. $\mathcal{SP}$ adds $\mathcal{C}_i$'s public key to a member list L and publish L on his public directory.

3.3.2 Basic Construction

The basic construction is between a single user $\mathcal{C}_i$ and a service provider $\mathcal{SP}$. We discuss the detail in the following:

- `Step1`: $\mathcal{C}_i$ forms a ring signature and forwards it to $\mathcal{SP}$. We assume the message is then transmitted through an anonymous channel. The ring signature σ is formed as follows:
  - $\mathcal{C}_i$ computes $\tilde{g} = H_2(L, m, T_1) \cdot g \pmod{p}$, where T_1 is a timestamp.
  - $\mathcal{C}_i$ selects a random number $a \in \mathbb{Z}_q$ and computes
    $$c_{k+1} = H_1(L||m||g||\tilde{g}||g^a||\tilde{g}^a) \ (mod\ q)$$
    where $\|$ denotes a bitwise concatenation.
  - For $i = k+1, ..., n-1, 0, 1, ..., k-1$, C_i randomly selects $s_i \in \mathbb{Z}_q$ and computes:
    $$c_{i+1} = H_1(L||m||g||\tilde{g}||e_i||\tilde{e}_i) \ (mod\ q)$$
    where $e_i = g^{s_i} y_i^{c_i} \pmod{p}$ and $\tilde{e}_i = g^{s_i} A^{c_i} \pmod{p}$, $A = \tilde{g}^{x_k} \pmod{p}$.
  - $\mathcal{C}_i$ computes $s_k = a - x_k c_k \pmod{q}$.

  The ring signature is $\sigma = (c_0, s_0, ..., s_{n-1}, A)$.
- `Step2`: Upon receiving the signature, the service provider $\mathcal{SP}$ verifies σ as follows:
  - $\mathcal{SP}$ collects the subset of public keys L and computes
    $$c_{i+1} = H_1(L||m||g||\tilde{g}||e_i||\tilde{e}_i) \ (mod\ q)$$
    for $i = 0, 1, ..., n-1$. It accepts the signature if
    $$c_0 = H_1(L||m||g||\tilde{g}||e_{n-1}||\tilde{e}_{n-1}) \ (mod\ q)$$
- `Step3`: If the signature is valid, the service provider $\mathcal{SP}$ proceeds as follows:
  - $\mathcal{SP}$ randomly picks a number t and computes $b = \tilde{g}^t \pmod{p}$.
  - $\mathcal{SP}$ encrypts the acknowledgement K by $\mathcal{Z} = K \cdot A^t \pmod{p}$ where $A = \tilde{g}^{x_k} \pmod{p}$. Note here K is used as an unlocking key which can be computed by each user independently with their private key. It contains the hash value of $\mathcal{SP}$'s identity, timestamp and other necessary information.
  - $\mathcal{SP}$ broadcasts ($\mathcal{Z}$, b).
- `Step4`: On receiving the above ciphertext, $\mathcal{C}_i$ decrypts the acknowledgement K with his private key x_k as follows:
  - $\mathcal{C}_i$ computes $K = \mathcal{Z}/b^{x_k}$.

3.3.3 Multi-User Construction

In practice, a number of users may communicate with a single service provider at the same time. Suppose there are t users, using the previous protocol. The service provider $\mathcal{SP}$ needs to verify t ring signatures and broadcast t messages back. This adds to the computation expense and increases communication overheads. In this section, we extend the ring signature scheme to support a multi-user scenario. Suppose there are t users $\mathcal{C}_1, ..., \mathcal{C}_t$ and a single service provider $\mathcal{SP}$. The protocol is as follows:

- `Step1`: t users $\mathcal{C}_1, ..., \mathcal{C}_t$ collaborate to form a ring signature and forward it to $\mathcal{SP}$. They first agree on a subset of public key L which includes their public keys and $n-t$ other members' public keys $pk_1, ..., pk_{n-t}$. They form the ring as follows:
  - Initialisation: The t users $\mathcal{C}_1, ..., \mathcal{C}_t$ randomly select $a_j \in \mathbb{Z}_q$, $j \in S_t$ (S_t is a subset of user who actually sign the message, a_j is selected by each user independently) and compute:
    $$c_{k+1} = H_1(L||m||g||\tilde{g}||\{g^{a_j}||\tilde{g}^{a_j}\}_{j\in S_t}) \ (mod\ q)$$
    where $\|$ denotes a bitwise concatenation and $\{A(j)||B(j)\}_{j\in S_t}$ denotes t bitwise concatenations with respect to $A(j)||B(j)$ and $j \in S_t$.
  - Forward sequence: For $i = k+1, ..., n-1, 0, 1, ..., k-1$ and $j \in S_t$, users randomly select $s_{i,j} \in \mathbb{Z}_q$ and $\mathcal{TP}$ compute:
    $$c_{i+1} = H_1(L||m||g||\tilde{g}||\{e_i^{(j)}||\tilde{e}_i^{(j)}\}_{j\in S_t}) \ (mod\ q)$$
    where $e_i^{(j)} = g^{s_{i,j}} y_{i,j}^{c_i} \pmod p$ and $\tilde{e}_i^{(j)} = \tilde{g}^{s_{i,j}} A_j^{c_i} \pmod p$, A_j is computed as $A_j = \tilde{g}^{x_j} \pmod p$, $j \in S_t$. Note here A_j is a temporary public key for C_j.
  - Forming the ring: Users compute $s_{k,j} = a_j - x_{k,j}c_k \pmod q$, $j \in S_t$.

  The signature is
  $$\sigma = (c_0, \{s_{0,J}, ..., s_{n-1,J}\}_{J=1,...,t})$$
  where J is a new suffix derived from $j \in S_t$ by renumbering in numerical order.

- `Step2`: Upon receiving the signature, the service provider $\mathcal{SP}$ verifies σ as follows:
  $$c_{i+1} = H_1(L||m||g||\tilde{g}||\{e_i^{(J)}||\tilde{e}_i^{(J)}\}_{J=1,...,t}) \ (mod\ q)$$
  where $e_i^{(J)} = g^{s_{i,J}} y_{i,J}^{c_i} \pmod p$ and $\tilde{e}_i^{(J)} = \tilde{g}^{s_{i,J}} A_J^{c_i} \pmod p$, $J = 1, ..., t$. The service provider accepts the signature if
  $$c_0 = H_1(L||m||g||\tilde{g}||\{e_{n-1}^J||\tilde{e}_{n-1}^J\}_{J=1,..,t}) \ (mod\ q)$$

- `Step3`: If the signature is valid, the service provider $\mathcal{SP}$ proceeds as follows:

  1. $\mathcal{SP}$ randomly picks a number t and computes $b = \tilde{g}^t$.
  2. $\mathcal{SP}$ computes $K_i = A_i^t$.
  3. $\mathcal{SP}$ computes the polynomial function $f(x) = \prod_{i=1}^{t}(x - K_i) \pmod p$. From $\prod_{i=1}^{t}(x - K_i) = \sum_{i=0}^{t} a_i x^i \pmod p$, we obtain

$$\begin{aligned} a_0 &= \prod_{j=1}^{t}(-K_i) \ (mod\ p) \\ a_1 &= \sum_{i=1}^{t}\prod_{j\neq i}^{n}(-K_j) \ (mod\ p) \\ &= \cdots \\ a_{t-1} &= \sum_{j=1}^{t}(-K_i) \ (mod\ p) \\ a_t &= 1 \ (mod\ p) \end{aligned}$$

     The $\{a_i\}$ satisfy $\sum_{j=0}^{t} a_j K_i^j = 0 \pmod p$, $i = 1, ..., t$.
  4. $\mathcal{SP}$ sends the ciphertext $\mathcal{Z} = (b,\ K_s \cdot \tilde{g}^{ra_0},\ \tilde{g}^{ra_1},\ ...,\ \tilde{g}^{ra_t})$ to the receivers, where r is a random number selected by the $\mathcal{SP}$.

- `Step4`: On receiving the above ciphertext, users proceed as follows:

  1. Each user C_i computes $K_i = b^{x_{k,i}} \pmod p$ with his private key $x_{k,i}$.
  2. Each $\mathcal{C}_i$ computes:

$$\begin{aligned} & K_s \cdot \tilde{g}^{ra_0} \cdot (\tilde{g}^{ra_1})^{K_i} \cdot (\tilde{g}^{ra_2})^{K_i^2} \cdots (\tilde{g}^{ra_t})^{K_i^t} \ (mod\ p) \\ = \ & K_s \cdot \tilde{g}^{r(a_0 + a_1 K_i + \cdots + a_t K_i^t)} \ (mod\ p) \\ = \ & K_s \cdot \tilde{g}^{rf(K_i)} \ (mod\ p) \\ = \ & K_s \cdot 1 \ (mod\ p) \\ = \ & K_s \end{aligned}$$

Note, K_s is the hash value of the identity of service provider, timestamp and other necessary information.

3.4 Security Analysis

In this section, we provide the security analysis of our scheme. First, we show the correctness of our scheme. Then we illustrate how the signer proves his knowledge of the

respective private key x_i that corresponds to his temporary public key A_i. Finally, we provide proof of signer ambiguity and the unforgeability of our ring signature scheme.

Correctness

Theorem 1 *Our R_n^t ring signature scheme is correct.*

Proof. We show that our ring signature is perfectly closed. That is to say, an outsider has no idea where the ring signature starts and ends.

$$\begin{aligned}
c_{k+1} &= H_1(L||m||g||\tilde{g}||\{g^{a_j}||\tilde{g}^{a_j}\}_{j\in S_t}) \ (mod\ q) \\
c_{k+2} &= H_1(L||m||g||\tilde{g}||\{g^{s_{k+1,j}} y_{k+1,j}^{c_{k+1}}||\tilde{g}^{s_{k+1,j}} A_j^{c_{k+1}}\}_{j\in S_t}) \ (mod\ q) \\
&= \cdots \\
c_n &= H_1(L||m||g||\tilde{g}||\{g^{s_{n-1,j}} y_{n-1,j}^{c_{n-1}}||\tilde{g}^{s_{n-1,j}} A_j^{c_{n-1}}\}_{j\in S_t}) = c_0 \ (mod\ q) \\
&= \cdots \\
c_{k+1} &= H_1(L||m||g||\tilde{g}||\{g^{s_{k,j}} y_{k,j}^{c_k}||\tilde{g}^{s_{k,j}} A_j^{c_k}\}_{j\in S_t}) \ (mod\ q) \\
&= H_1(L||m||g||\tilde{g}||\{g^{a_j}||\tilde{g}^{a_j}\}_{j\in S_t}) \ (mod\ q)
\end{aligned}$$

With a correct ring signature, the user $\mathcal{C}_i$ proves his membership to the service provider. That is because all the public keys in the ring belong to the members. In order to close the ring, $\mathcal{C}_i$ should possess at least one of the member's private keys, otherwise he can not produce a valid signature. Therefore, $\mathcal{SP}$ is assured that $\mathcal{C}$ is a member but he can not tell which one. □

Knowledge Proof

When signing a message, the user $\mathcal{C}_i$ actually proves the following discrete logarithm equalities:

$$\log_g y_{k,j} = \log_{\tilde{g}} A_j$$

where $y_{k,j} = g^{x_k} \pmod p$ and $A_j = \tilde{g}^{x_k} \pmod p$. x_k is the private key of the signer. This proof is based on Chaum and Pedersen's proof of equality of discrete logs. We reduce the proof in our scheme as follows. Given $y = g^x \pmod p$, $A = \tilde{g}^x \pmod p$ prove $\log_g y = \log_{\tilde{g}} A$. We can see that in our scheme, the customer chooses a random $a \in \mathbb{Z}_q$ and computes $s = a - cx \pmod q$ at the closing point where $c = H_1(L||m||g||\tilde{g}||g^s y^c||\tilde{g}^s A^c) \pmod q$. The proof output is (c, s). The verification is done by verifying $c \stackrel{?}{=} H_1(L||m||g||\tilde{g}||g^s y^c||\tilde{g}^s A^c) \pmod q$.

Signer Ambiguity

Theorem 2 *Our R_n^t ring signature scheme satisfies the property of unconditional signer ambiguity.*

Proof. In our scheme, c_0 is independent of L, m and $s_{i,j}$. All $s_{i,j}$ are randomly selected except $s_{k,j}$ at the closing point. That is to say all the $s_{i,j}$ except $s_{k,j}$ are false signer identities. And $s_{k,j}$ is also distributed uniformly over $\mathbb{Z}_q^*$, since a_j is chosen randomly from $\mathbb{Z}_q^*$. Therefore, for fixed (L, m), $\{s_{i,j}\}_{j \in S_t}$ $(i = 0, ..., n-1)$ has q^n solutions, all of which can be chosen with equality probability, regardless of the signers.

Unforgeability

Theorem 3 *If there exists (t, ϵ, q_h)-adversary $\mathcal{A}$ for public key set L of size n, then there exists a simulator (τ, μ)-***SIM** *that takes advantage of $\mathcal{A}$ to compute the discrete logarithm x_k of $(p, q, y_k, g) \in L$ and $\mathcal{A}_k$ with a probability at least μ within running time τ, for $\tau < 2/\epsilon$ and $\mu > \frac{9}{25}$ under the condition that $\epsilon > \frac{q_h}{q_p}$.*

Note (t, ϵ, q_h)-adversary denotes an adversary that runs in time t, makes q_h queries to the hash oracle and has an advantage $Adv(\mathcal{A}) < \epsilon$. q_p is the number of times the signing oracle is accessed. Let $\mathcal{A}$ be an UF-ACMA adversary in the unforgeability game. We will build a simulator $\mathcal{B}$ that will use $\mathcal{A}$ to solve the discrete logarithm problem. The purpose of $\mathcal{B}$ is to compute x_k from y_k which is given at the beginning of the game. The simulation is as follows:

1. $\mathcal{B}$ provides $\mathcal{A}$ with the common parameters cp.
2. Each time $\mathcal{A}$ issues a hash query on $Q_j = (j, L_j, m_j, e_j, \tilde{e}_j)$, $\mathcal{B}$ will answer the query as follows:
   - $\mathcal{B}$ maintains a hash record. If an entry for the query is found, the same answer will be given to $\mathcal{A}$. Otherwise, a random value will be used as an answer to $\mathcal{A}$, and the query and the answer will then be stored.
3. Each time $\mathcal{A}$ issues a sign query $R_j = (L_j, m_j)$, $\mathcal{B}$ will answer the query as follows:
   - If a query on $R_j = (L_j, m_j)$ has been asked before, the same answer will be given to $\mathcal{A}$.

– If a query on $R_j = (L_j, m_j)$ has not been asked before, $\mathcal{B}$ will answer the query as follows:

* Select $a \in \mathbb{Z}_q^*$, $A \in Z_p^*$
* Compute $\tilde{g} = H_2(L, m, T_1) \cdot g$
* Compute $c_0 = H_1(L||m||g||\tilde{g}||g^a||\tilde{g}^a)$
* For $i = 0, ..., |L_j|-1$, select $s_i \in \mathbb{Z}_q$, compute $e_i = g_i^{s_i} y_i^{c_i}$ and $\tilde{e_i} = \tilde{g}_i^{s_i} A^{c_i}$.
* Compute $c_{i+1} = H_1(L||m||g||\tilde{g}||e_i||\tilde{e}_i)$ for $i \neq |L_j| - 1$
* Assign c_0 to the value of $H_1(L||m||g||\tilde{g}||e_{|L_j|-1}||\tilde{e}_{|L_j|-1})$

Here $|L_j| = n$, $\mathcal{B}$ outputs $s_{n-1} = a - x_{n-1}c_{n-1}$ as the answer. The simulation fails if the last step causes inconsistency in H_1. It happens with probability at most q_h/q_p. For simplicity, we assume that q_h is the number of times the hash oracle is accessed in the last step and omit other queries.

4. Eventually, $\mathcal{A}$ outputs a forged ring signature $\sigma = \langle c_0, s_0, ..., s_{|L_j|-1}, A\rangle$ with a successful probability of ϵ.

The possibility of success for a (t, ϵ, q_h) adversary $\mathcal{A}$ running an experiment is at least $\epsilon = \frac{q_h}{q_p}$. By invoking $\mathcal{A}$ $t_1 = 1/\epsilon$ times, $\mathcal{B}$ finds a valid $s_{|L_j|-1}$ and $\mathcal{A}$ with a probability at least:

$$1 - (1 - \epsilon)^{1/\epsilon} \approx 1 - e^{-1} > 3/5$$

Now $\mathcal{B}$ runs $\mathcal{A}$ again for $t_2 = t_1$ times with the same public key set L_j but a different message m'_j. And let $g' = g$, $\tilde{g}' = \tilde{g}$, $a' = a$ and $A' = A$, but $c'_0 \neq c_0$. From the signing algorithm, $\mathcal{B}$ can determine a $s'_{|L_j|-1}$ such that $s'_{|L_j|-1} = a - x_{|L_j|-1}c'_{|L_j|-1}$. Since $c'_0 \neq c_0$, $c'_{|L_j|-1} \neq c_{|L_j|-1}$. $x_{|L_j|-1}$ can be computed by

$$\begin{aligned}
& \frac{s'_{|L_j|-1} - s_{|L_j|-1}}{c_{|L_j|-1} - c'_{|L_j|-1}} \\
= & \frac{(a - x_{|L_j|-1}c'_{|L_j|-1}) - (a - x_{|L_j|-1}c_{|L_j|-1})}{c_{|L_j|-1} - c'_{|L_j|-1}} \\
= & \frac{x_{|L_j|-1}(c_{|L_j|-1} - c'_{|L_j|-1})}{c_{|L_j|-1} - c'_{|L_j|-1}} \\
= & \; x_{|L_j|-1}
\end{aligned}$$

The total time used for the simulation is $t_1 + t_2 = \frac{2q_p}{q_h}$ with probability $\mu > (3/5)^2 = 9/25$.

3.5 Efficiency

Let n be the number of potential signers and t be the number of real signers of ring signature. Let $T(DL)$, T(RSA^{-1}), T(RSA) be the computational costs of modular exponentiation, inverse RSA function and RSA function. Typically $T(DL) = T((1024)^{160})$, T(RSA^{-1}) = T(1024^{1024}) and T(RSA) = T(1024^{16}). The table below shows a summary of the computation costs of our proposed schemes compared with DL and RSA signatures.

	Costs of Generation	Costs of Verification
WI Signatures	$T(DL) \times 5/4 \times n$	$T(DL) \times 5/4 \times n$
RSA Ring Signatures	$T(RSA^{-1}) + T(RSA) \times n$	$T(RSA) \times n$
Our Proposed Signatures I	$2 \times T(DL) \times 5/4 \times n$	$2 \times T(DL) \times 5/4 \times n$
Our Proposed Signatures II	$2t \times T(DL) \times 5/4 \times n$	$2t \times T(DL) \times 5/4 \times n$

Let L(DL) be the length of exponent of DL signature and L(RSA) be the length of modular of RSA signature. Typically L(DL) = 160-bit and L(RSA) = 1024-bit. The table below shows a summary of signature size of our proposed schemes, together with DL and RSA signatures.

	Signature Size
WI Signatures	$2 \times L(DL) \times n$
RSA Ring Signatures	$(L(RSA) + 160) + (L(RSA) + 160) \times n$
Our Proposed Signatures I	$L(DL) \times n + L(DL)$
Our Proposed Signatures II	$L(DL) + L(DL) \times n \times t$

3.6 Applications

3.6.1 E-Commerce

Introduction

Electronic Commerce (E-Commerce) usually refers to the distributing, buying and selling of products or services over electronic systems such as the Internet and other computer networks. Over the past few years, E-Commerce has been growing with surprising speed. According to a statistics report [56] the value of total E-Commerce sales in the year 2004 was approximately $69 billion, while in 2005 it was $86.3 billion, indicating an increase of 24.6 percent.

With E-Commerce services continually increasing, there is also an increasing concern about preserving privacy during E-Commerce transactions. Many solutions have been given, such as [63, 85, 98]. Among them, one efficient and practical scheme was proposed by Bao, Deng and Feng [7]. Their scheme makes use of a cryptographic primitive called a commutative symmetric key cryptosystem. They assume merchants have n digital goods for sale, each good is encrypted with a separate secret key K_i and all the secret keys are encrypted again with a single master key S using a commutative encryption algorithm. The customers can download goods anonymously. However, they need the merchant's help to decrypt. To retain anonymity, the customers first blind the goods with a random secret key R and then ask the merchant to decrypt. Thus, the merchant does not know which goods the customer actually purchased. However, their approach has the following drawbacks:

1. The price of all goods should be the same if the merchant want to charge the price per good. Otherwise, the merchant could find out which goods the customers have purchased by the amount of money they paid. This puts many restrictions on the application of the scheme since in practice it is unreasonable to set all goods at the same price.

2. The interaction between customers and merchant is not anonymous if payment is made by membership. That is because the authentication phase will inevitably reveal some private information about the customer.

3. The merchant can only collect statistics on the downloaded time of goods. This does not reflect the number of goods actually sold, because a customer may download digital goods without payment.

To overcome these disadvantages, we consider a privacy protection issue where customers want to maintain control over their personal information. We restrict our scenario to anonymous online sales of digital goods. That is, the customers want to purchase goods from a merchant over the Internet anonymously. At the same time, they also want to use their membership privileges, such as discounts, so they need to authenticate themselves to the shop owner. Our proposed protocol should enable the customers to achieve both these aims.

Construction

The protocol we proposed in 3.3.2 can be used to construct such an E-Commerce scheme. Here, the merchant takes the role of the service provider and the customer takes the role of user. Besides the customer $\mathcal{C}$ and the merchant $\mathcal{M}$, a bank $\mathcal{B}$ is also involved in the transaction. $\mathcal{B}$ is in charge of issuing anonymous electronic cash to $\mathcal{C}$ and transferring money to $\mathcal{M}$. We assume the customers $\mathcal{C}$ and the merchant $\mathcal{M}$ have registered accounts with $\mathcal{B}$ before the transaction. $\mathcal{M}$ advertises goods over its website.

- `Order`: $\mathcal{C}$ starts the transaction by sending $\mathcal{M}$ the following items:
  - The purchase *Order* which includes the ID of items $\mathcal{C}$ is interested in.
  - $\mathcal{C}$'s signature on the *Order*, which is formed as in *Step1*.
  - The electronic cash $\mathcal{C}$ obtained from $\mathcal{B}$. It consists of t coins $\{Coin_i\}_{i=1,\ldots,t}$, the total of which is equal to the price of the item.
  - A timestamp T_1.

  All of these are encrypted with $\mathcal{M}$'s public key $y_{\mathcal{M}}$ so that only the merchant can decrypt and learn the content.

- `Check`: Upon receiving the *Order* message, merchant $\mathcal{M}$ decrypts it and verifies the signature σ as in *Step2*. If the signature on the *Order* message is valid, merchant $\mathcal{M}$ gets the electronic coin $\{Coin_i\}_{i=1,\ldots,t}$ and checks the validity of the coin using the bank's public key $y_{\mathcal{B}}$. After that, $\mathcal{M}$ contacts $\mathcal{B}$ to get the money.

- `Deposit`: On receiving $\mathcal{M}$'s request, the bank checks the $\{Coin_i\}_{i=1,\ldots,t}$ against double-spending. If none have been spent yet, $\mathcal{B}$ deposits the appropriate amount to $\mathcal{M}$'s account and sends back an acknowledgement.

- `Delivery`: On receiving $\mathcal{B}$'s acknowledgement, $\mathcal{M}$ encrypts the authorisation key as in *Step3* and broadcasts it to $\mathcal{C}$.

- `Decrypt`: On receiving the above ciphertext, $\mathcal{C}$ decrypts the authorisation key K with his private key x_i as in *Step 4*.

3.6.2 Pay-TV

Introduction

Pay television, or pay TV, refers to subscription-based television services, usually provided by either analogue and digital cable, or by satellite, but also increasingly by digital terrestrial methods[54]. A Pay TV system consists of a broadcaster and a number of subscribers. Each subscriber is entitled to some subset of services. The broadcaster encrypts the content and sends it through the network so that only those users who have subscribed to the service can decrypt and watch the programs.

Recently, as the technologies for collecting and analyzing personal information advances, the privacy protection issue in Pay-TV has become an important concern. Customers are worried about the easy acquisition and abuse of their personal data such as TV-watching habits. Therefore, various ways of protection privacy have been proposed. For example, in 2000 Lee [64] proposed his *Privacy and Non-repudiation on Pay-TV System*. In his scheme, a conditional access system is employed so that only designated subscribers can watch the TV programs while unauthorised viewers can see nothing. A preferred conditional access system (CAS) could make sure that no one can find out to which channel a particular subscriber has registered except the system administrator. And even if the system administrator knows to which channels the particular subscriber has registered, he has no evidence to prove it to other people. However, this system only protects the subscribers' privacy from outsiders. In 2003, Rong, Song and Korba proposed a *Pay-TV System with Strong Privacy and Non-Repudiation Protection*[97]. In his paper, a new CAS mechanism was developed. It not only protects a customer's privacy against malicious outsiders, but also prevents service providers from collecting subscribers' personal information. This is realised by using an "e-ticket". A customer must first buy the ticket from the providers and when the customer wants to subscribe to some program, he just shows his ticket to the providers. However, the use of an "e-ticket" requires additional storage space. In practice, this kind of subscribe could always be done through an anonymous channel such as Mix-net so that user do not need to lose any of their privacy.

To overcome the above disadvantages, we apply our privacy protection protocol to Pay-TV. Our new pay-TV protocol provides perfect subscriber anonymity, not only from the malicious outsider, but also from the Service Provider, who learns nothing about the subscriber. Moreover, the subscriber can subscribe to the TV program in real time; there is no requirement for pre-establishment and storage of any "e-ticket".

Construction

The protocol proposed in 3.3.3 can be used to construct the *anonymous Pay-TV* protocol. We assume the t subscribers take the role of users in 3.3.3. The Pay-TV broadcaster takes the role of both KGC and service provider.

- `Subscribe`: Several subscribers within a group could collaborate to subscribe for certain TV program. In order to hide their identities, they need to collect the public keys of non-subscribers within their group and form a ring signature as in *Step 1*. In practice, we assume group members know each other and they exchange information. One group member may act as coordinator and all the information are transferred through that member to the service provider.
- `Check`: On receiving the subscription message, the Pay-TV broadcaster verifies the signature as in *Step 2*.
- `Broadcast`: The Pay-TV broadcaster encrypts the session key for the TV program as in *Step 3*.
- `Watch`: The subscribers get the session key as in *Step 4*.

3.7 Summary

In this chapter, we proposed two proven secure and practical privacy protection protocols based on ring signatures. In our first construction, a user in a ring is able to communicate with another party outside anonymously. At the same time, the outsider can authenticate the user and send back an acknowledgement. After that, we extended the basic construction to support a multi-party scenario, where two or more users can collaborate to communicate with a single party. Meanwhile, that party is able to determine the exact number of users involved. We also investigated possible applications of our protocols, such as for *E-Commerce* and *Pay-TV*. We showed how to use our proposed protocols to implement them.

Chapter 4

Identity-Based Anonymous Designated Ring Signature

4.1 Introduction

In this chapter, we propose the concept of *identity-based anonymous designated ring signature*. We then proceed with a construction based on bilinear pairings and provide the security proof for our construction. Finally, we extend the scheme to a *convertible ID-based anonymous designated ring signature*.

4.1.1 Motivation

The user privacy protection protocol we proposed in the previous chapter provides good privacy protection with authentication ability. However, it does not guarantee the authenticity of the verifier. Consider a situation where Alice would like to share something from her music collection to a Peer-to-Peer (P2P) network, such as BitTorrent or Emule. She would like to share her music with several restrictions, as follows. 1) Alice does not want her identity to be known by anyone, 2) She only wants to share her music with a specific group (eg. a specific group that belongs to a particular forum on the web) that she has identified and therefore, she would like to obtain some confirmation from this group, and 3) the member of the group who would like to confirm Alice's offer also does not want his identity revealed. In this situation, in order to ensure the authenticity of the file (or music, in this example), Alice needs to obtain some confirmation from the members of the group after she offers her music file.

Formally, we would like to have a cryptographic primitive that satisfies the following. An offerer, A, can offer her resource to a specific group in the P2P network, such that one of the group confirms that he/she would like to obtain this resource, but both identities (A's identity and the confirmer's identity) must be kept anonymous. This

problem can be solved with a primitive that we call an *anonymous designated ring signature*. Here, a member of the ring can sign on behalf of the group in the ring to a *designated* party, but the identity of the designated party remains unknown to the signer. In this paper, we are interested in the *identity-based* version of *anonymous designated ring signatures* since our application is directly related to P2P networks where each member's identity can be derived from his/her IP address in the network.

4.1.2 Contributions of This Chapter

The contribution of this chapter is twofold. First, we introduce the concept of *Anonymous Designated Ring Signatures*. In such ring signature schemes, a signature holder can designate the signature to any anonymous designated verifier. We present the formal model for such a scheme, and proceed with an efficient construction from pairings. Furthermore, we extend this concept to *convertible anonymous designated ring signatures* that allow the "anonymous designated verifier" to reveal her identity.

4.2 Our Construction

4.2.1 System Model

The *Identity-Based Anonymous Designated Ring Signature* is a ring signature which has the following additional properties:

1. The signature is designated to a particular verifier.
2. The identities of neither signer nor designated verifier are revealed.

There are three distinct parties in an identity-based anonymous designated ring signature (ID-ADRS) scheme, namely a Key Generation Centre (KGC) who generates the secret keys for the participants, an offerer (or a designated verifier), $\mathcal{O}$, and an ad-hoc group of signers, $\mathcal{U} = \{U_1, \cdots, U_n\}$.

An identity-based anonymous designated ring signature (ID-ADRS) scheme consists of the following algorithms.

- `Setup`: On input of a security parameter ℓ, it outputs a master secret key s and common parameters cp. That is, $\{s, cp\} \leftarrow \texttt{Setup}(\ell)$.

- `Signer Key Generation`(SKG): On input of an arbitrary string $\mathsf{ID}_i \in \{0,1\}^*$ and KGC's master secret key s, it outputs private key $\mathcal{S}_{ID_i}$ for signer U_i.
- `Verifier Key Generation`(VKG): On input of an arbitrary string $\mathsf{ID}_{\mathcal{O}} \in \{0,1\}^*$ and KGC's master secret key s, it outputs verifier's private key $\mathcal{S}_{\mathcal{O}}$.
- `Sign`: A signing algorithm is an interactive algorithm between the offerer $\mathcal{O}$ and the ad-hoc group of signers $\mathcal{U}$. On input a signer's secret key, $\mathcal{S}_{\mathsf{ID}_i}$, $U_i \in \mathcal{U}$, and a message $m \in \{0,1\}^*$, it produces a signature σ, where *only* the offerer $\mathcal{O}$ can verify the authenticity of σ (using $\mathcal{O}$'s secret key or secret commitment, respectively).
- `Verify`: On input of $\mathcal{O}$'s secret key (or secret commitment, respectively), a signature σ on a message m, it produces `Accept` or `Reject`.

4.2.2 Cryptographic Requirements

Completeness

We require the IDADS scheme to satisfy the following probability equation:

$$Pr[y \leftarrow \mathbf{Sign}(cp, m, L, S_{ID_k}) : \mathbf{Verify}(cp, L, m, S_{ID_{\mathcal{O}}}, y)] = 1$$

where

$$\begin{aligned} L &= \{ID_1, ..., ID_n\} \\ cp &\leftarrow Setup(\ell) \\ S_{ID_k} &\leftarrow SKG(cp, ID_k) \\ S_{ID_{\mathcal{O}}} &\leftarrow VKG(cp, ID_{\mathcal{O}}) \\ \sigma &\leftarrow Sign(cp, m, L, S_{ID_k}, R) \\ \texttt{True} &\leftarrow Verify(cp, m, L, \sigma) \end{aligned}$$

Note this equation means for any message m, any subset of signers L and any valid private key S_{ID_k}, the probability of a signature accepted by the verifier is always one.

Perfect Anonymity

We require the IDADS scheme to satisfy both signer ambiguity and offerer anonymity. For any message m, any subset of of identities L and any $\sigma \leftarrow \mathbf{Sign}(m, L, S_{ID})$, where

$ID \in L$, we require that an unbound adversary $\mathcal{A}$ outputs i such that $ID = ID_i$ with probability $1/|L|$. We also require that $\mathcal{A}$ could find $ID_{\mathcal{O}}$ with only negligible probability. The perfectness of anonymity can be defined on uniform distributions and the difference between a priori and a posteriori knowledge. For an anonymity set of size N, if the adversary's priori knowledge about the signer before attack equals the posteriori knowledge after the attack, then we have perfect anonymity.

Unforgeability

We require any signature produced by the `Sign` algorithm to be unforgeable. Formally, it is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The existential unforgeability of an ID-ADRS scheme under a chosen message attack (UF-ID-ADRS-CMA) is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Let $\mathcal{A}$ be the UF-ID-ADRS-CMA adversary. In the startup of the game, $\mathcal{C}$ provides the common parameter cp to $\mathcal{A}$, where $cp \leftarrow \mathbf{Setup}(\ell)$ and ℓ is the security parameter.
- $\mathcal{C}$ provides $\mathcal{A}$ the possible signer's identities $(ID_1, ..., ID_n)$ and offerer's public parameter R.
- At any time, $\mathcal{A}$ can query the hash oracle on any identity ID of his choice up to q_{H_1} times and query any message of his choice up to q_{H_2} times. $\mathcal{C}$ will answer $\mathcal{A}$'s queries by providing the hash value.
- At any time, $\mathcal{A}$ can query the signing oracle for the signature on any message m_i, any subset of identities L and any designated offer $\mathcal{O}$ up to q_S times. $\mathcal{C}$ will answer $\mathcal{A}$'s query by providing the value σ.
- Eventually, $\mathcal{A}$ will output a valid ID-ADRS for a message m^*, a subset of identities L^* that has never been queried before for an offerer's parameter R^*.

The success probability of an adversary is defined by $Succ_{\mathcal{A}}^{UF-ID-ADRS-CMA}(\ell)$ where ℓ is negligible.

We say that an ID-ADRS scheme is existentially unforgeable under a chosen message attack if the probability of success of any polynomially bounded adversary in the above game is negligible. In other words,

$$Succ_{\mathcal{A}}^{UF-ID-ADRS-CMA}(\ell) \leq \epsilon$$

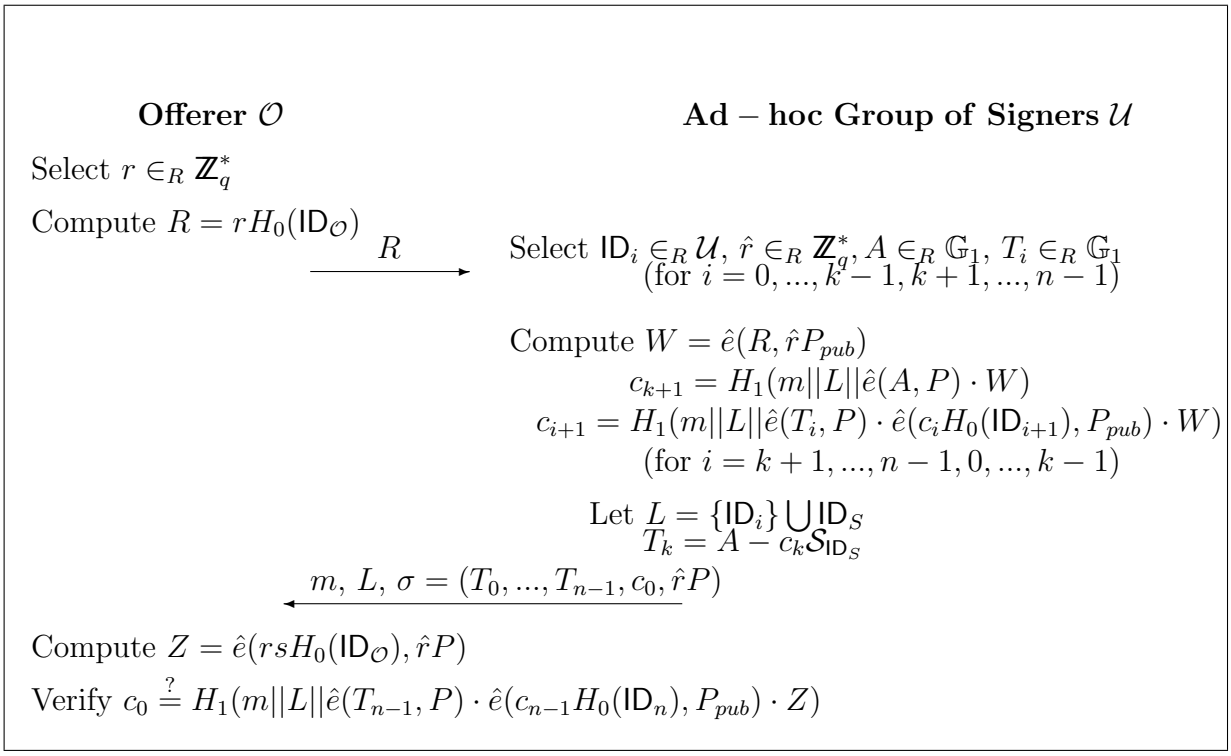

Figure 4.1: ID-ADRS Sign and Verify

where ϵ is negligible.

4.3 Implementation of Our Scheme

In this section, we present an efficient Identity-based anonymous designated ring signature (ID-ADRS) scheme. Our scheme is inspired by Zhang-Kim's ID-based ring signature scheme [104]. Suppose m is the message to be signed. Let $U_i \in \mathcal{U}, i \in L$ be an ad-hoc group of potential signers. $\mathcal{O}$ is the offerer and U_S is the signer. The protocol is shown in 4.1.

– **Setup**. On input of a security parameter ℓ, it generates two groups $(\mathbb{G}_1, +)$ and $(\mathbb{G}_M, \cdot)$ of prime order q and a bilinear pairing map $\hat{e} : (\mathbb{G}_1, +)^2 \to (\mathbb{G}_M, \cdot)$, together with an arbitrary generator $P \in \mathbb{G}_1$. It also selects its master key $s \in \mathbb{Z}_q^*$ and sets $P_{pub} = sP$. Finally, two cryptographically strong hash functions $H_0 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ are chosen.

– **Signer Key Generation**. On input of an arbitrary string $\mathsf{ID}_k \in \{0,1\}^*$ and KGC's master key s, it computes $\mathsf{Q}_{\mathsf{ID}_s} = H_0(\mathsf{ID}_s) \in \mathbb{G}_1$ and the corresponding private key $\mathcal{S}_{\mathsf{ID}_s} = s\mathsf{Q}_{\mathsf{ID}_s}$. Then KGC sends the private key to the signer.

– **Verifier Key Generation**. On input of an arbitrary string $ID_{\mathcal{O}} \in \{0,1\}^*$ and the KGC's master key s, it computes $\mathsf{Q}_{\mathsf{ID}_{\mathcal{O}}} = H_0(ID_{\mathcal{O}}) \in \mathbb{G}_1$ and the corresponding private key $\mathcal{S}_{\mathsf{ID}_{\mathcal{O}}} = s\mathsf{Q}_{\mathsf{ID}_{\mathcal{O}}}$. The KGC sends the private key to the offerer.

– **Sign**. The signing algorithm is an interactive algorithm between the offerer $\mathcal{O}$ and the ad-hoc group of signers $\mathcal{U}$.

  – An offerer $\mathcal{O}$ randomly selects $r \in \mathbb{Z}_q^*$, computes $R = rH_0(\mathsf{ID}_{\mathcal{O}})$ and broadcasts it to $\mathcal{U}$.

  – On receiving R, $\mathcal{U}_S$ randomly selects $\mathsf{ID}_i \in \mathcal{U}$ for $i = 0, 1, \cdots, k, k+1, \cdots, n-1$. Let $L = \cup_{\forall i}\{\mathsf{ID}_i\}$ where the identity ID_S is listed in L. The signature generation is as follows.

1. $\mathcal{U}_S$ randomly selects $A \in \mathbb{G}_1$, $\hat{r} \in \mathbb{Z}_q^*$ and computes

$$c_{k+1} = H_1(m||L||\hat{e}(A,P)\hat{e}(rH_0(\mathsf{ID}_{\mathcal{O}}), \hat{r}P_{pub}))$$

   where $||$ denotes bitwise concatenation.

2. For $i = k+1, ..., n-1, 0, 1, ..., k-1$, $\mathcal{U}_S$ randomly selects $T_i \in \mathbb{G}_1$ and computes

$$\begin{aligned} c_{i+1} &= H_1(m||L||\hat{e}(T_i,P)\hat{e}(c_iH_0(\mathsf{ID}_{i+1}), P_{pub}) \\ &\quad \hat{e}(rH_0(\mathsf{ID}_{\mathcal{O}}), \hat{r}P_{pub})) \end{aligned}$$

3. Finally, $\mathcal{U}_S$ computes $T_k = A - c_k\mathcal{S}_{\mathsf{ID}_S}$.

The signature is $\sigma = (T_0, ..., T_{n-1}, c_0, \hat{r}P)$.

– **Verify**. On receiving (m, L, σ), $\mathcal{O}$ computes $Z = \hat{e}(rsH_0(\mathsf{ID}_{\mathcal{O}}), \hat{r}P)$ and accepts the message if the following equation holds with equality.

$$c_0 \stackrel{?}{=} H_1(m||L||\hat{e}(T_{n-1}, P) \cdot \hat{e}(c_{n-1}H_0(\mathsf{ID}_n), P_{pub}) \cdot Z)$$

4.4 Security Analysis

Correctness

Theorem 4 *All correct signatures will pass the verification test.*

Proof. We show the correctness of the verification test as follows. To verify the signature σ, an offerer $\mathcal{O}$ needs to have a verification secret key $rsH_0(\mathsf{ID}_\mathcal{O})$ that corresponds to the public key $rH_0(\mathsf{ID}_\mathcal{O})$. Then, it verifies the signature $\sigma = (T_0, ..., T_{n-1}, c_0, \hat{r}P)$ as follows.

$$
\begin{aligned}
c_1 &\stackrel{?}{=} H_1(m||L||\hat{e}(T_0, P)\hat{e}(c_0 H_0(\mathsf{ID}_1), P_{pub}) \\
&\quad \hat{e}(rsH_0(\mathsf{ID}_\mathcal{O}), \hat{r}P)) \\
c_2 &\stackrel{?}{=} H_1(m||L||\hat{e}(T_1, P)\hat{e}(c_1 H_0(\mathsf{ID}_2), P_{pub}) \\
&\quad \hat{e}(rsH_0(\mathsf{ID}_\mathcal{O}), \hat{r}P)) \\
&\vdots \\
c_{k+1} &\stackrel{?}{=} H_1(m||L||\hat{e}(T_k, P)\hat{e}(c_k H_0(\mathsf{ID}_S), P_{pub}) \\
&\quad \hat{e}(rsH_0(\mathsf{ID}_\mathcal{O}), \hat{r}P)) \\
&\stackrel{?}{=} H_1(m||L||\hat{e}(A - c_k sH_0(\mathsf{ID}_S), P) \\
&\quad \hat{e}(c_k H_0(\mathsf{ID}_S), P_{pub})\hat{e}(rsH_0(\mathsf{ID}_\mathcal{O}), \hat{r}P)) \\
&\stackrel{?}{=} H_1(m||L||\hat{e}(A, P)\hat{e}(-c_k H_0(\mathsf{ID}_S), sP) \\
&\quad \hat{e}(c_k H_0(\mathsf{ID}_S), sP)\hat{e}(rsH_0(\mathsf{ID}_\mathcal{O}), \hat{r}P)) \\
&\stackrel{?}{=} H_1(m||L||\hat{e}(A, P)\hat{e}(rH_0(\mathsf{ID}_\mathcal{O}), \hat{r}sP)) \\
&\vdots \\
c_n &\stackrel{?}{=} H_1(m||L||\hat{e}(T_{n-1}, P)\hat{e}(c_{n-1} H_0(\mathsf{ID}_n), P_{pub}) \\
&\quad \hat{e}(rH_0(\mathsf{ID}_\mathcal{O}), \hat{r}P_{pub})) \\
&\stackrel{?}{=} c_0
\end{aligned}
$$

Anonymity

Theorem 5 *Our ID-ADRS scheme satisfies the property of unconditional signer ambiguity.*

Proof. In our scheme, c_0 and R are independent of L, m and $s_{i,j}$. All T_i are randomly selected except T_k at the closing point. Furthermore T_k is also distributed uniformly

over $\mathbb{G}_1$, since A is chosen randomly from $\mathbb{G}_1$. Therefore, for fixed (L, m), T_k has q^n solutions, all of which can be chosen with equal probability, regardless of the signer. The probability of an attacker outputting T_k is $1/q^n$.

Theorem 6 *Our ID-ADRS scheme satisfies the property of offerer anonymity.*

Proof. In our scheme, the offerer's temporary public key is associated with a random number $r \in \mathbb{Z}_q^*$. An adversary $\mathcal{A}$ is unable to distinguish $rH_0(\mathsf{ID}_{\mathcal{O}})$ from a random number $R \in \mathbb{G}_1$. Besides, our ring signature does not provide the linkability. That is to say an adversary cannot decide whether two messages are signed by the same signer. Thus, the anonymity of the offerer is guaranteed.

Unforgeability

Theorem 7 *In the random oracle model, if there is an algorithm $\mathcal{A}$ that can win the UF-ID-ADRS-CMA game in a polynomial time, then there exists a challenger $\mathcal{C}$ that can use $\mathcal{A}$ to solve the CDHP problem with a non-negligible probability in polynomial time.*

To prove our scheme is secure against IND-CPA, we first assume that there exists an adversary $\mathcal{A}$ who wins in the indistinguishability experiment described in Section 4.2.2. Then we create a simulator $\mathcal{B}$ that intercepts all the communication between $\mathcal{A}$ and the challenger $\mathcal{C}$. $\mathcal{B}$ is able to modify and forward the communication contents and is transparent to $\mathcal{A}$ and $\mathcal{C}$. $\mathcal{A}$ sees no difference between the simulator $\mathcal{B}$ and the challenger $\mathcal{C}$. The goal of $\mathcal{B}$ is to make use of $\mathcal{A}$ to solve a cryptographic hard problem. Since the hard problem is known to be unsolvable in polynomial time, the assumption that $\mathcal{A}$ exists leads to a contradiction and hence our scheme is secure. We first review the cryptographic hard problem that we will use in the proof:

CDH is hard with an assumption that there does not exist a polynomial time algorithm with which any attacker can solve CDH, such that the probability of success is non-negligible.

We construct the simulator $\mathcal{B}$ as follows (the challenger $\mathcal{C}$ can be omitted here as $\mathcal{B}$ simulated it):

1. $\mathcal{B}$ is given an instance (P, aP, bP) of CDH as described above.
2. $\mathcal{A}$ picks a group of users ID's to be attacked and tells $\mathcal{B}$.

3. $\mathcal{B}$ runs the KGC's `Setup` algorithm to generate the necessary system parameters. The parameters $\{\mathbb{G}_1, \mathbb{G}_M, \hat{e}, q, n, P, P_{pub}, H_0, H_1\}$ are modified by $\mathcal{B}$ by setting P_{pub} to bP before being given to $\mathcal{A}$.

4. Each time $\mathcal{A}$ issues a hash query on any identity ID_i, $i = 1, ..., q_{H_1}$, $\mathcal{B}$ will answer the query as follows:

   – $\mathcal{B}$ maintains a hash record list L_1 to store all the hash results, which grows as each new hash result is obtained.

   – If the query has been asked before, then $\mathcal{B}$ looks in the records to obtain $H_0(ID_i)$ and answers with the stored value.

   – If the query on ID_i has not been asked before, then $\mathcal{B}$ picks a random number $d_i \in \mathbb{Z}_q^*$ and flips a $\{0, 1\}$ coin W that has probability α on outcome 0 and $1 - \alpha$ on outcome 1. If 0 is obtained, $\mathcal{B}$ answers with $H_0(ID_i) = d_iP$. Otherwise, $\mathcal{B}$ answers with $H_0(ID_i) = aP$. $\mathcal{B}$ updates his record on L_1. Note that when $W = 0$, the associated private key is d_iSP which $\mathcal{B}$ knows how to compute. But when $W = 1$, since both a and b are unknown to $\mathcal{B}$, $\mathcal{B}$ can not generate the associated private key.

5. Each time $\mathcal{A}$ issues a sign query on any message m_i, $i = 1, ..., q_S$, any subset of identity L_i, any random tuple R of his choice, $\mathcal{B}$ will answer the query as follows:

   – $\mathcal{B}$ maintains a list L_3 to store all the sign query results, which grows as a new hash result is obtained.

   – If the query (m_j, L_j, R) has been asked before, the same answer will be given to $\mathcal{A}$.

   – If the query (m_j, L_j, R) has never been asked before

     * $\mathcal{B}$ randomly chooses $A \in \mathbb{Z}_q^*$, $\hat{r} \in \mathbb{Z}_q^*$.
     * $\mathcal{B}$ computes $c_0 = H_1(m||L||\hat{e}(A, P)\hat{e}(R, \hat{r}P_{pub}))$
     * For $i = 0, ..., |L_j| - 1$, $\mathcal{B}$ randomly selects $T_i \in \mathbb{G}_1$, computes

       $$c_{i+1} = H_1(m||L||\hat{e}(T_i, P)\hat{e}(c_iH_0(\mathsf{ID}_{i+1}), P_{pub})\hat{e}(R, \hat{r}P_{pub}))$$

       for $i \neq |L_j| - 1$.
     * Assigns c_0 to the value of

       $$H_1(m||L||\hat{e}(T_{|L_j|-1}, P)\hat{e}(c_{|L_j|-1}H_0(\mathsf{ID}_{|L_j|}), P_{pub})\hat{e}(R, \hat{r}P_{pub}))$$

Here $|L_j| = n$. $\mathcal{B}$ outputs $T_{|L_j|-1} = A - c_{|L_j|-1}S_{\mathsf{ID}|L_j|-1}$ as answers and updates the list L_3. The simulation fails if the last step causes inconsistency in H_1.

6. Eventually, $\mathcal{A}$ outputs a forged signature (m^*, L^*, σ^*) that is signed by a member $\mathsf{ID}^* \in L^*$. The only restrictions here are that σ^* has never been queried in the above *Sign* request and $\mathcal{A}$ has never queried any one of the private keys of the members in the group L.

We assume $\mathcal{A}$ wins the above game with the probability of ϵ by running the above experiment for t_1 times. The output of $\mathcal{A}$ is $T_{|L_j|-1}$ where $|L_j| = n$. It follows from the forking lemma [81] that $\mathcal{B}$ can break CDH by running $\mathcal{A}$ again with different message m' but the same set of public keys$|L_j|$. The parameter A, R remains the same. Because $m' \neq m$, the value of $c_{|L_j|} = H_1(m||L||\hat{e}(T_{|L_j|-1}, P)\hat{e}(c_{|L_j|-1}H_0(\mathsf{ID}_{|L_j|}), P_{pub})\hat{e}(R, \hat{r}P_{pub}))$ is different from $c'_{|L_j|} = H_1(m'||L||\hat{e}(T_{|L_j|-1}, P)\hat{e}(c_{|L_j|-1}H_0(\mathsf{ID}_{|L_j|}), P_{pub})\hat{e}(R, \hat{r}P_{pub}))$. With the outputs $T_{|L_j|}$ and $T'_{|L_j|}$, $\mathcal{B}$ could compute:

$$\begin{aligned}
& \frac{T'_{|L_j|} - T_{|L_j|}}{c_{|L_j|} - c'_{|L_j|}} \\
= & \frac{A - c'_{|L_j|}S_{\mathsf{ID}^*} - (A - c_{|L_j|}S_{\mathsf{ID}^*})}{c_{|L_j|} - c'_{|L_j|}} \\
= & \frac{(c_{|L_j|} - c'_{|L_j|})S_{\mathsf{ID}^*}}{c_{|L_j|} - c'_{|L_j|}} \\
= & S_{\mathsf{ID}^*} \\
= & abP
\end{aligned}$$

Hence, $\mathcal{B}$ has successfully solved the CDH problem for the given instance (P, aP, bP). However, since CDH problem is intractable, there is a contradiction so no such $\mathcal{B}$ exists.

4.5 Convertible Identity-Based Anonymous Designated Ring Signature

In the P2P network, sometimes an offerer will be interrogated, especially when there is a case of a malicious file distributed over the network. In this situation, the offerer must have the ability to prove that he/she is innocent, by providing a proof that he/she has a particular file. We resolve this case by providing a primitive called a convertible

ID-based anonymous designated ring signature, where the offerer is equipped with the ability to *prove* the ownership of a resource (a message or a file).

4.5.1 System Model

Formally, Convertible ID-based Anonymous Designated Ring Signatures are ID-ADRS schemes that consist of the following algorithms:

- `Setup`: On input of a security parameter ℓ, it outputs a master secret key s and common parameters cp. That is, $\{s, cp\} \leftarrow \texttt{Setup}(\ell)$.
- `Signer Key Generation` (SKG): On input of an arbitrary string $\mathsf{ID}_i \in \{0,1\}^*$ and KGC's master secret key s, it outputs private key $\mathcal{S}_{ID_i}$ for signer U_i.
- `Verifier Key Generation` (VKG): On input of an arbitrary string $\mathsf{ID}_{\mathcal{O}} \in \{0,1\}^*$ and KGC's master secret key s, it outputs the verifier's private key $\mathcal{S}_{\mathcal{O}}$.
- `Sign`: Signing algorithm is an interactive algorithm between the offerer $\mathcal{O}$ and the ad-hoc group of signers $\mathcal{U}$. On input of a signer's secret key, $\mathcal{S}_{\mathsf{ID}_i}$, $U_i \in \mathcal{U}$, and a message $m \in \{0,1\}^*$, it produces a signature σ, where *only* the offerer $\mathcal{O}$ can verify the authenticity of σ (using $\mathcal{O}$'s secret key or secret commitment, respectively).
- `Verify`: On input of $\mathcal{O}$'s secret key (or secret commitment, respectively), a signature σ on a message $m \in \{0,1\}^*$, it produces `Accept` or `Reject`.
- `Convert`: The convert algorithm is an non-interactive algorithm used by the offerer $\mathcal{O}$. On input of an offerer's secret committed value r, it produces a proof (c, z) which shows that the offerer knows the secret committed value r in the commitment R.

4.5.2 Cryptographic Requirements

In addition to the **Correctness** and the **Signer Ambiguity**, we require our convertible ID-ADRS scheme to satisfy the following cryptographic requirement:

Convertible Unforgeability

We require that only the offerer $\mathcal{O}$ can produce a valid convertible ID-ADRS. Formally, it is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Let $\mathcal{A}$ be the UF-CIDADRS-CMA adversary. In the startup of the game, $\mathcal{C}$ provides the common parameters cp to $\mathcal{A}$, where $cp \leftarrow \text{Setup}(\ell)$ and ℓ is the security parameter.
- At any time, $\mathcal{A}$ can query the hash oracle on any ID_i of his choice up to q_{H_1} times. $\mathcal{C}$ will answer $\mathcal{A}$'s queries by providing the hash value $H_0(ID_{\mathcal{O}})$.
- At any time, $\mathcal{A}$ can query the hash oracle on any commitment R_i of his choice up to q_{H_2} times. $\mathcal{C}$ will answer $\mathcal{A}$'s queries by providing the value of R_i.
- At any time, $\mathcal{A}$ can query the convertible signing oracle for any ID-ADRS signature σ of his choice up to q_S times. $\mathcal{C}$ will answer $\mathcal{A}$'s queries by providing the value (c, z).
- Eventually, $\mathcal{A}$ will output a valid proof (c^*, z^*) for an ID-ADRS signature σ^* that has never been queried before.

The success probability of an adversary is defined by $Succ_{\mathcal{A}}^{UF-CIDADRS-CMA}(\ell)$ where ℓ is negligible.

We say that a convertible ID-ADRS scheme has existential unforgeability under a chosen message attack if the probability of success of any polynomially bounded adversary in the above game is negligible. In other words,

$$Succ_{\mathcal{A}}^{UF-CIDADRS-CMA}(\ell) \leq \epsilon$$

where ϵ is negligible.

4.5.3 A Convertible ID-ADRS scheme based on pairings

We extend our scheme given in Section 4.2 to construct a convertible ID-ADRS scheme. The convertible ID-ADRS scheme is as follows:

- `Setup`: The same as ID-ADRS scheme.
- `Signer Key Generation`: The same as ID-ADRS scheme.
- `Verifier Key Generation`: The same as ID-ADRS scheme.
- `Sign`: The same as ID-ADRS scheme.
- `Verify`: The same as ID-ADRS scheme.

- `Convert`: Given a valid signature σ, the offerer $\mathcal{O}$ could produce a proof that she is the 'owner' of the commitment $R = rH(\mathsf{ID}_{\mathcal{O}})$ constructed at the first stage. The algorithm is as follows:

  1. The offerer $\mathcal{O}$ randomly selects $a \in \mathbb{Z}_q$.
  2. The offerer $\mathcal{O}$ computes $c = H_1(H_0(ID_{\mathcal{O}})||R||aH_0(ID_{\mathcal{O}})||m)$
  3. The offerer $\mathcal{O}$ computes $z = a - cr$.

  Note that *only* $\mathcal{O}$ can generate this proof, and anyone can verify whether this proof is correct by verifying the commitment (c, s). The verifier computes

$$\begin{aligned} & H_1(H_0(ID_{\mathcal{O}})||R||cR + zH_0(ID_{\mathcal{O}})||m) \\ = \; & H_1(H_0(ID_{\mathcal{O}})||R||crH(ID_{\mathcal{O}}) + (a - cr)H_0(ID_{\mathcal{O}})||m) \\ = \; & H_1(H_0(ID_{\mathcal{O}})||R||crH(ID_{\mathcal{O}}) + aH_0(ID_{\mathcal{O}}) - crH_0(ID_{\mathcal{O}})||m) \\ = \; & H_1(H_0(ID_{\mathcal{O}})||R||aH_0(ID_{\mathcal{O}})||m) \end{aligned}$$

  The verifier then checks if the above value is equal to c. If it is, he believes the offerer $\mathcal{O}$ is innocent.

4.5.4 Security Analysis

Correctness and Signer Ambiguity

Proof. The `Correctness` and `Signer Ambiguity` of the convertible ID-ADRS scheme is the same as the previous ID-ADRS scheme. It is therefore omitted.

Convertible Unforgeability

Theorem 8 *In the random oracle model, if there is an algorithm $\mathcal{A}$ that can win the UF-CIDADRS-CMA game in polynomial time, then there exists a challenger $\mathcal{C}$ that can use $\mathcal{A}$ to solve the DL problem with a non-negligible probability in polynomial time.*

We construct the simulator $\mathcal{B}$ as follows:

1. $\mathcal{B}$ is given an instance (P, aP) of DL problem.
2. Each time $\mathcal{A}$ issues a hash query on an identity ID_i of his choice, $\mathcal{B}$ will answer it as follows:

- $\mathcal{B}$ maintains a list L_1 to store all the query results, it grows as new results are generated.
- If the query has been asked before, $\mathcal{B}$ looks up his record to obtain $H_0(ID_i)$ and answers with the stored value.
- If the query has never been asked before, $\mathcal{B}$ picks a random number $d_i \in \mathbb{Z}_q^*$ and answers with $H_0(ID_i) = d_iP$. $\mathcal{B}$ updates his record on L_1 and gives it to $\mathcal{A}$.

3. Each time $\mathcal{A}$ issues a query on (R_i, ID_i) of his choice, $\mathcal{B}$ will answer it as follows:
   - $\mathcal{B}$ maintains a list L_2 to store all the query results, it grows as new result are generated.
   - If the query has been asked before, $\mathcal{B}$ looks up his record to obtain $H_0(ID_i)$ and answers with the stored value.
   - If the query has never been asked before, $\mathcal{B}$ picks a random number $s_i \in \mathbb{Z}_q^*$ and flips a $\{0, 1\}$ coin W that has probability α on outcome 0 and $1 - \alpha$ on outcome 1. If 0 is obtained, $\mathcal{B}$ answers with $R_i = s_i d_i P$. Otherwise, $\mathcal{B}$ answers with $R_i = s_i a d_i P$. $\mathcal{B}$ updates his record on L_2 and gives it to $\mathcal{A}$.
4. Each time $\mathcal{A}$ issues a query on (R_i, σ), $\mathcal{B}$ will answer the query as follows:
   - $\mathcal{B}$ maintains a list L_3 to store all the query results, it grows as new result are generated.
   - If the query (R_i, σ) has never been asked before
     - $\mathcal{B}$ randomly picks $v \in \mathbb{Z}_q$.
     - $\mathcal{B}$ computes $c = H_1(H_0(ID_i)||R_i||vH_0(ID_i)||m)$.
     - $\mathcal{B}$ computes $z = v - cs_i$.

     $\mathcal{B}$ updates his record on L_3 and answers (c, z).
   - If the query (R_i, σ) has been asked before, the same answer will be given to $\mathcal{A}$.

4 Eventually, $\mathcal{A}$ outputs a forged proof (c^*, z^*) on offerer's commitment R^* and the signature σ^*. The only restriction here is (R^*, σ^*) has never been queries in the previous signing queries.

We assume $\mathcal{A}$ wins the above game with probability ϵ by running the above experiment t_1 times. The output of $\mathcal{A}$ is (c, z). It follows from the forking lemma that $\mathcal{B}$ can break DLP by running $\mathcal{A}$ again with different R, but the same v and $H_0(ID)$. Hence, $c' \neq c$ and $s' \neq s$. $\mathcal{B}$ could compute:

$$\begin{aligned} & \frac{z' - z}{cs_i - c's'_i} \\ = & \frac{v - c's'_i a - v + cs_i a}{cs_i - c's'_i} \\ = & \frac{(cs_i - c's'_i)a}{cs_i - c's'_i} \\ = & a \end{aligned}$$

However, since DLP is intractable, there is a contradiction so no such $\mathcal{B}$ exists.

4.6 Summary

In this chapter, we proposed the concept of *identity-based anonymous designated ring signature*. After that we presented a formal model of such a scheme, and proceeded with a construction based on bilinear pairings. We proved that our scheme is provably secure under the random oracle model. We also provided a convertible version of our scheme which enables a verifier to reveal his identity in a particular session.

Chapter 5

Conclusion

In this thesis, we have used cryptography to address aspects of privacy protection problem on the Internet. Many cryptographic tools can be used to provide anonymity. For example, we can use blind signatures to create anonymous electronic cash, we can use Mix-net to hide a user's identity or we can use a credential system to enable interaction between user and service provider. However, these cryptographic tools are not suitable for the problems we intend to solve in this thesis, which are: 1. To solve the contradiction between anonymity and authentication. 2. To solve the contradiction between anonymity and designated property.

In Chapters 4 and 5, we presented our contribution to the privacy protection problem. Our objectives in this thesis are

1. There is a conflict between privacy and security. Since security is far more important than privacy on the Internet, in order to maintain it, privacy is often compromised. For example, in Pay-TV systems, subscribers are required to authenticate themselves to the broadcaster. This reveals subscribers' identities and enables the broadcaster to collect information about the subscribers. However, if we go without authentication, anyone could watch the TV programs.

2. There is a conflict between privacy and other system requirements. For example, an anonymous user may want to designate the verification ability to group members. However, the designated party does not wish to reveal his membership unless he can be sure the signer is also a member.

We have successfully achieved these goals and we summarise our contribution in each chapter below.

In Chapter 4, we proposed two privacy protection protocols. The basic construction allows a user to interact with a service provider without revealing his identity. At the

same time, the service provider is able to authenticate the user as a member. Also, he can send back an acknowledgement. This is realised by utilising ring signatures together with an embedded public key. The ring signature itself provides perfect anonymity so that the service provider can only tell whether a particular communication request is from a member, but cannot tell which member it is. Moreover, in our scheme, the user actually proves his knowledge of the private key corresponding to the public key he embedded. Therefore, the originality of the signer is guaranteed. We have extended the basic construction to multi-user in our second protocol. It is more efficient than the first one when many users would like to interact with a single service provider at the same time. We have also investigated possible applications of our two protocols. The first one is E-Commerce which enables customers to engage in transactions with online merchants anonymously while authenticating the membership. The second one is Pay-TV in which subscribers could interact with broadcasters with minimum personal information revealed. Thus, in this chapter, we achieved our first goal. With our protocol, the user is able to protect his privacy while a security requirement such as authentication can still be achieved.

In Chapter 5, we proposed the concept of identity-based anonymous designated ring signature, which has not been considered before. This concept extends the existing notion of ring signatures in two ways: first, it allows a member of the ring to sign a message directed to a designated verifier, but second, we can also have an anonymous designated verifier. Then, we provided a construction of our concept based on bilinear pairings. With such a scheme in P2P networks, one party is able to authenticate himself as a group member to another designated party. At the same time, both parties can remain anonymous. We also formulated a security model for our ring signature scheme and proved its security under the random oracle model. Furthermore, we extended the scheme to *convertible identity-based Anonymous Designated Ring Signatures*. With the convertible property, the anonymous designated verifier is able to prove his knowledge of a commitment he has made. Thus, in this chapter, we have achieved our second goal. We developed a new ring signature scheme that provides perfect anonymity together with designated and convertible properties.

Bibliography

[1] M. Abe. *Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures.* Proceedings of Eurocrypt'01, Lecture Notes in Computer Science 2045, pages. 136 – 151, 2001.

[2] M. Abe and E. Fujisaki. *How to Date Blind Signatures.* Proceedings of Asiacrypt'96, Lecture Notes in Computer Science 1163, pages. 244 – 251, 1996.

[3] M. Abe, M. Ohkubo, and K. Suzuki. *1-out-of-n Signatures from a Variety of Keys.* Advances in Cryptology - Asiacrypt 2002, Lecture Notes in Computer Science 2501, pages 415 – 432, 2002.

[4] M. Abe and T. Okamoto. *Provably Secure Partially Blind Signature.* Proceedings of Crypto'00, Lecture Notes in Computer Science 1880, pages. 271 – 286, 2000.

[5] A. Awasthi and S. Lal. *ID-based Ring Signature and Proxy Ring Signature Schemes from Bilinear Pairings.* Cryptology ePrint Archive, Report 2004/184, 2004.

[6] G. Ateniese, J. Camenisch, M. Joye and G. Tsudik. *A practical and provably secure coalition-resistant group signature scheme.* Proceedings of CRYPO'00, Lecture Notes in Computer Science 1880, pages. 255 – 270, 2000.

[7] F. Bao, R. Deng, P. Feng. *An Efficient and Practical Scheme for Privacy Protection in the E-Commerce of Digital Goods.* Lecture Notes in Computer Science 2015, pages. 162 – 170, 2001.

[8] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. *The power of RSA inversion oracles and the security of Chaum RSA-based blind signature scheme.* Financial Cryptography'01, 2001.

[9] M. Bellare, D. Micciancio, B. Warinschi. *Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions.* Advances in Cryptology - EUROCRPYT 2003, Lecture Notes in Computer Science 2656, pages. 614 – 629, 2003.

[10] M. Bellare and P. Rogaway *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.* ACM Conference on Computer and Communications Security, pages. 62 – 73, 1993.

[11] M. Bellare and O. Goldreich. *On defining proofs of knowledge.* In Advances in Cryptology - CRYPTO'92, Lecture Notes in Computer Science 740, pages. 390 – 420, 1992.

[12] D. Boneh, B. Lynn, and H. Shacham. *Short signatures from the weil pairing.* Advanced in Cryptology - Asiacrypt 2001, Lecture Notes in Computer Science 2248, Springer Verlag, pages 514–532, 2001.

[13] G. Brassard, D. Chaum and C. Crepeau. *Minimum disclosure proofs of knowledge.* Journal of Computer and System Sciences, 37(2), pages. 156 – 189, 1988.

[14] E. Brickell, P. Gemmell, and D. Kravitz. *Trustee-based tracing extensions to anonymous cash and the making of anonymous change.* In symposium on Distributed Algorithms(SODA). Albuquerque, NM., 1995.

[15] E. Bresson, J. Stern and M. Szydlo. *Threshold ring signatures and applications to ad-hoc groups.* In CRYPTO 2002, Lecture Notes in Computer Science 2442, pages. 465 – 480, 2002.

[16] J. Baek, R. Safavi-Naini, and W. Susilo. *Universal Designated Verifier Signature Proof.* Advances in Cryptology - Asiacrypt 2005, Lecture Notes in Computer Science 3788, pages 644 – 661, 2005.

[17] M. Bellare. *Practice-Oriented Provable-Security.* In Proceedings of First International Workshop on Information Security, Lecture Notes in Computer Science Vol. 1396, pages. 1 – 15, 1999.

[18] J. Camenisch. *Efficient and generalized group signatures.* Proceedings of EUROCRYPT'97, Lecture Notes in Computer Science 1233, pages. 465 – 479, 1997.

[19] J. Camenisch. *Group signature schemes and payment systems based on the discrete logarithm problem.* PhD thesis, ETH Zürich, 1998.

[20] J. Camenisch and M. Stadler. *Efficient group signatures for large groups.* Proceedings of CRYPTO'97, Lecture Notes in Computer Science 1296, pages. 410 – 424, 1997.

[21] S. Chow, L. Hui, and S. Yiu. *Identity Based Threshold Ring Signature.* Information Security and Cryptology C ICISC 2004, Lecture Notes in Computer Science 3506, pages. 218 – 232, 2004.

[22] T. Chan, K. Fung, J. Liu and V. Wei. *Blind spontaneous anonymous group signatures for ad-hoc groups.* In ESAS 2004, Lecture Notes in Computer Science 3313, pages. 82 – 94, 2004.

[23] D. Chaum. *Untraceable electronic mail, return address, and digital pseudonyms.* Comm. of the ACM. Vol. 24, No. 2, 1981.

[24] D. Chaum. *Blind signatures for untraceable payments.* Advances in Cryptology - Crypto '82, Springer-Verlag 1983, pages. 199 – 203, 1983.

[25] D. Chaum. *Security without identification: transaction systems to make big brother obsolete.* Communications of the ACM 28(10), pages. 1030 – 1044, 1985.

[26] D. Chaum. *Zero Knowledge Undeniable Signatures.* In Advances in Cryptology - EUROCRYPT' 90, Lecture Notes in Computer Science 473, pages. 458 – 464, 1990.

[27] D. Chaum and E. Heijst. *Group signatures.* Advances in Cryptology - Eurocrypt '91, Springer-Verlag 1991, pages. 257 – 265, 1991.

[28] D. Chaum and H. Antewepen. *Undeniable Signatures.* In Advances in Cryptology - CRYPTO' 89, Lecture Notes in Computer Science 435, pages. 212 – 216, 1990.

[29] D. Chaum, A. Fiat and M. Naor. *Untraceable Electronic Cash.* In Advances in Cryptology - Proceedings of CRYPTO '88, Lecture Notes in Computer Science 403, pages. 319 – 327, 1990.

[30] D. Chaum and T. Pedersen. *Wallet Databases with Observers.* In Advances in Cryptology - CRYPTO'92, Lecture Notes in Computer Science 740, pages. 89 – 105, 1992.

[31] L. Chen *Access with Pseudonyms.* Lecture Notes in Computer Science 1029, pages. 232 – 243, 1995.

[32] L. Chen and T. Pedersen. *New group signature schemes.* Proceedings of EUROCRYPT'94, Lecture Notes in Computer Science, 950, pages. 171 – 181, 1994.

[33] W. Diffie and M. Hellman. *New directions in cryptography.* IEEE Trans. Inform. Theory IT-22, 6, pages. 644 – 654, 1976.

[34] Y. Desmedt, C. Goutier and S. Bengio *Special Uses and Abuses of the Fiat-Shamir Passport Protocol.* In Advances in Cryptology - CRYPTO' 87, Lecture Notes in Computer Science 293, pages. 21 – 39, 1998.

[35] Y. Desmedt and M. Yung *Weakness of Undeniable Signature Schemes.* In Advances in Cryptology - EUROCRYPT' 91, Lecture Notes in Computer Science 547, pages. 205 – 220, 1991.

[36] Y. Desmedt and Y. Frankel. *Threshold cryptosystems.* In CRYPTO 1989, Lecture Notes in Computer Science 435, pages. 307 – 315, 1989.

[37] T. Eng and T. Okamoto. *Single-Term Divisible Electronic Coins.* In Advances in Cryptography - Proceedings of EUROCRYPT '94, Lecture Notes in Computer Science 950, pages. 306 – 319, 1995.

[38] B. Flinn. *Levels of Anonymity.* Journal of Universal Computer Science, vol. 1, pages. 35 – 47, 1995.

[39] M. Franklin and M. Yung. *Secure and Efficient Off-Line Digital Money.* In Proceedings of ICALP '93, Lecture Notes in Computer Science 700, pages. 265 – 276, 1993.

[40] U. Feige, A. Fiat and A. Shamir. *Zero Knowledge proofs of indetity.* Journal of Cryptology 1, pages. 77 – 94, 1988.

[41] A. Fiat and M. Naor. *Broadcast Encryption.* In Advances in Cryptology - Crypto 93, Springer-Verlag. LNCS 773, pages. 480 – 491, 1993.

[42] N. Ferguson. *Single Term Off-Line Coins.* In Advances in Cryptology - Proceedings of EUROCRYPT '93, Lecture Notes in Computer Science 765, pages. 318 –328, 1994.

[43] I. Goldberg, D. Wagner and E. Brewer. *Privacy-enhancing technologies for the Internet.* IEEE COMPCON '97, 1997.

[44] O. Goldreich, S. Goldwasser and S. Micali. *On the cryptographic applications of random functions.* Crypto 84, 1984.

[45] O. Goldreich, S. Goldwasser and S. Micali. *How to construct random functions.* Journal of the ACM, Vol. 33, No. 4 pages. 210 – 217, 1986.

[46] S. Goldwasser and S. Micali. *Probabilistic encryption,.* J. of Computer and System Sciences, Vol. 28, pages. 270 – 299, 1984.

[47] S. Goldwasser, S. Micali, R. Rivest. *A "paradoxical" solution to thesignature problem.* In Proceeding. of FOCS84, pages. 441 – 448, 1984.

[48] S. Goldwasser, S. Micali and C. Rackoff. *The knowledge Complexity of Interactive proof systems.* In Proceeding of 27th Annual Symposium on Foundations of Computer Science, pages. 291 – 304, 1985.

[49] S. Goldwasser, S. Micali, R. Rivest. *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.* SIAM Journal on Computing 17(2), pages. 281 – 308, 1988.

[50] J. Herranz and G. Saez. *New Identity-Based Ring Signature Schemes*. Cryptology ePrint Archive, Report 2003/261, 2003.

[51] https://www.anonymizer.com/.

[52] http://anonymizer.com.

[53] http://en.wikipedia.org/wiki/Privacy.

[54] http://en.wikipedia.org/Pay_TV.

[55] http://www.ietf.org/html.charters/pkix-charter.html.

[56] http://www.census.gov/estats.

[57] http://www.ftc.gov/os/1998/9808/d9255pub.id.pdf.

[58] http://ocw.mit.edu/NR/rdonlyres/

[59] http://www.pickocc.org/publications/telecom_fraud.pdf.

[60] http://www.zeroknowledge.com/.

[61] T. Isshiki, and K. Tanaka. *An (n - t) -out-of-n Threshold Ring Signature Scheme* . Information Security and Privacy: 10th Australasian Conference, ACISP 2005, Brisbane, Australia, July 4-6, 2005. Proceedings , Lecture Notes in Computer Science 3574, pages. 406 – 416, 2005.

[62] M. Jakobsson, K. Sako and R. Impagliazzo. *Designated Verifier Proofs and Their Applications*. In Advances in Cryptology - EUROCRYPT' 96, Lecture Notes in Computer Science 1070, pages. 143 – 154, 1996.

[63] D. Kugler. *Enabling Privacy Protection in E-commerce Applications*. Lecture Notes in Computer Science 2232, pages. 127 – 138, 2001.

[64] N. Lee, C. Chang, C. Lin, and T. Hwang. *Privacy and Nonrepudiation on Pay-TV Systems*. IEEE Transactions on Consumer Electronics, Vol.46, No.1, pages. 20 – 26, 2000.

[65] J. Liu, V. Wei and D. Wong. *A separable threshold ring signature scheme*. In ICISC 2003, Lecture Notes in Computer Science 2981, pages. 12 – 26, 2003.

[66] J. Liu, V. Wei and D. Wong. *Linkable spontaneous anonymous group signature for ad-hoc gourps*. In ACISP 2004, Lecture Notes in Computer Science 3108, pages. 325 – 335, 2004.

[67] C. Lin and T. Wu. *An Identity-based Ring Signature Scheme from Bilinear Pairings.*. Cryptology ePrint Archive, Report 2003/117, 2003.

[68] L. Lamport. *Constructing digital signatures from a one-way function.* Technical Report CSL-98, SRI International, 1979.

[69] A. Lysyanskaya and Z. Ramzan. *Group Blind Digital Signatures: A Scalable Solution to Electronic Cash.* Financial Cryptography: Second International Conference, FC'98, Lecture Notes in Computer Science 1465, pages. 184 –197, 1998.

[70] Y. Mu and V. Varadharajan. *Robust and Secure Broadcasting.* In Progress in Cryptology - Indocrypt'01, Lecture Notes in Computer Science 2247, pages. 223 – 231, 2001.

[71] Y Mu, F. Zhang and W. Susilo. *Secure and Anonymous Mobile Ad-Hoc Networks.* IEEE International Conference on Networks,2005.

[72] A. Menezes and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Chapter 2, 1996.

[73] A. Menezes and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Chapter 10, 1996.

[74] R. Merkle. *A certificated digital signature.* In Advances in Cryptology - CRYPTO'89, Lecture Notes in Computer Science Vol 435, pages. 218 – 238, 1990.

[75] M. Naor. *Deniable Ring Authentication.* Advances in Cryptology - CRYPTO 2002, Lecture Notes in Computer Science 2442, pages. 481 – 498, 2002.

[76] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS).* FIPS Publication 180, 1993.

[77] T. Okamoto and K. Ohta. *Universal Electronic Cash.* In Advances in Cryptology - Proceedings of CRYPTO '91, Lecture Notes in Computer Science 576, pages. 324 – 337, 1992.

[78] H. Petersen. *How to convert any digital signature scheme into a group signature scheme.* In Security Protocols Workshop, Paris, 1997.

[79] D. Pointcheval and J. Stern. *Provably Secure Blind Signature Schemes.* In Asiacrypt'96, Lecture Notes in Computer Science 1163, pages. 252 – 265, 1996.

[80] D. Pointcheval and J. Stern. *Security Proofs for Signature Schemes.* In Eurocrypt'96, Lecture Notes in Computer Science 1070, pages. 387 – 398, 1996.

[81] D. Pointcheval and J. Stern. *Security Arguments for Digital Signatures and Blind Signatures.* Journal of Cryptology: The Journal of the International Association for Cryptologic Research, pages 361 – 396, 2000.

[82] M. Rabin. *Digitalized signatures as intractable as factorization* Technical Reprot MIT/LCS/TR-212, 1979.

[83] M. Reiter and A. Rubin. *Anonymity for web transactions.* ACM Transactions on Information and Systems Security, 1(1), pages. 66 – 92, 1998.

[84] M. Reiter and A. Rubin. *Anonymous web transaction with Crowds.* Comm. of the ACM. Vol. 42, No. 2, 1999.

[85] I. Ray and M. Geisterfer. *Towards a Privacy Preserving e-Commerce Protocol.* Lecture Notes in Computer Science 3128, pages. 154 – 163, 2004.

[86] R. Rivest. *The MD5 message-digest algorithm.* Internet Engineering Task Force: RFC 1321, 1992.

[87] R. Rivest and A. Shamir. *Payword and Micromint: Two simple micropaymnet schemes.* RSA Security Conference, 1996.

[88] R.Rivest, A. Shamir and L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM, volumn 21(2), pages. 120 – 126, 1978.

[89] W. Susilo, and Y. Mu. *Non-interactive Deniable Ring Authentication.* Information Security and Cryptology -ICISC 2003, Lecture Notes in Computer Science 2971, pages. 386 – 401, 2003.

[90] W. Susilo and Y.Mu. *Non-interactive Deniable Ring Authentication.* Information Security and Cryptology -ICISC 2003, Lecture Notes in Computer Science 2971, pages. 386 – 401, 2004.

[91] M. Stadler, J. Piveteau and J. Camenisch *Fair Blind Signatures.* Proceedings of EUROCRYPT'95, Lecture Notes in Computer Science 921, pages. 209 – 219, 1995.

[92] C. Schnorr. *Efficient signature generation for smart cards* Journel of Cryptology 4(3), pages. 239 – 252, 1991.

[93] A. Shamir. *Identity-Based Cryptosystems and Signature Schemes.* Advances in Cryptology: Proceedings of CRYPTO 84, Lecture Notes in Computer Science 196, pages. 47 – 53, 1984.

[94] A. Shamir, R. Rivest, and Y. Tauman. *How to leak a secret.* In Advances in Cryptology-ASIANCRYPT 2001, Lecture Notes in Computer Science 2248, pages. 552 – 565, 2001.

[95] R. Steinfeld, H. Wang, J. Pieprzyk. *Efficient Extension of Standard Schnorr/RSA signatures into Universal Designated-Verifier Signatures.* In Proceeding of PKC'04, Springer Lecture Notes in Computer Science 2947, pages. 86 – 100, 2004.

[96] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. *Universal designated-verifier signatures.* Proceedings of Asiacrypt 2003, Lecture Notes in Computer Science 2894, pages 523 – 543, 2003.

[97] R. Song, L. Korba. *Pay-TV System with Strong Privacy and Non-Repudiation Protection.* IEEE Transactions on Consumer Electronics, Vol. 49, No. 2, 2003.

[98] J. Traore. *Group Signatures and Their Relevance to Privacy-Protecting Offline Electronic Cash Systems.* In Advances in Cryptology, ACISP'99, Lecture Notes in Computer Science 1587, pages. 228 – 243, 1999.

[99] P. Tsang and V. Wei. *Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation.* In ISPEC 2005, Lecture Notes in Computer Science 3439, pages. 48 – 60, 2005.

[100] P. Tsang, V. Wei, T. Chan, M. Au, J. Liu, and D. Wong. *Separable Linkable Threshold Ring Signatures.* Progress in Cryptology - INDOCRYPT 2004, Lecture Notes in Computer Science 3348, pages. 384 – 398, 2004.

[101] D. Wong, K. Fung, J. Liu and V. Wei. *On the RS-code construction of ring signature schemes and a threshold setting of RST.* In ICISC 2003, Lecture Notes in Computer Science 2836, pages. 34 – 46, 2003.

[102] S. Warren and L. Brandeis. *The right to privacy.* Harvard Law Review, IV(5):193 – 220, 1890.

[103] J. Xu, Z. Zhang and D. Feng. *A Ring Signature Scheme Using Bilinear Pairings.* Information Security Applications: 5th International Workshop, Lecture Notes in Computer Science 3325, pages. 160 – 170, 2005.

[104] F. Zhang and K. Kim. *ID-Based Blind Signature and Ring Signature from Pairings.* Advances in Cryptology - ASIACRYPT 2002, Lecture Notes in Computer Science 2501, pages. 533 – 547, 2002.

[105] F. Zhang, R. Naini and W. Susilo. *An Efficient Signature Scheme from Bilinear Pairings and Its Applications.* Public Key Cryptography C PKC 2004, Lecture Notes in Computer Science 2947, pages. 277 – 290, 2004.