Two Results on Spontaneous Anonymous Group Signatures

CHAN Kwok Leong

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Information Engineering

©The Chinese University of Hong Kong
Aug, 2005

摘要

這裡我們設計了二個自發性匿名群簽章 ： 瞎自發性匿名群簽章 和 可分開的連結自發性匿名群簽章。 瞎簽名為用戶提供一個途徑從簽署人獲得一個簽名, 簽署人不能獲得關於所簽署文件內容。連結自發性匿名群簽章提供一個方式確定任何兩個簽章是否由同一人簽署。

在自發性匿名群簽章, 任一個實體可任意與其它成員組成一個群組。在簽名生成過程中 成員對包括是未察覺的。它不需要設置階段或成員連接的程式, 也不需要被信任的第三方, 所以沒有人可以知道誰製造密文或簽章。它能夠提供無條件的私隱保護，即不論群組內的成員或群組外的人也不能分辨出誰是加密者或署名者。

我們介紹二個添加了盲簽名和連結屬性的自發性匿名群簽章。這些自發性匿名群簽章提供不同的匿名屬性，以提供不同的匿名需求:

1. (盲簽名) 發送者身分是無條件地安全, 並且簽署人也不能從盲自發性匿名群簽章中找到哪些人等曾參與製造這個簽名

2. (連結屬性) 發送者身分是保密, 它能對同一簽署人簽發的電子文件作出聯繫 。

除盲自發性匿名群簽章和連結自發性匿名群簽章外，我們會簡介專門術語、自發性匿名群簽章和盲簽章的綜合概論及保安模型。 最後，我們會總覽一些自發性匿名群簽章的應用例子。

Abstract

Here we present two results on the Spontaneous Anonymous Group (SAG) Signature - 1. *Blind SAG Signature* and 2. *Separable Linkable SAG signature.* Blind signature provides a mean for users to obtain a signature from a signer, in which the signer can obtain no information about the message. Linkable signature provides a way to determine wherever two messages are generated by the same signer.

In spontaneous anonymous group (SAG) signature schemes, any entity can from arbitrary groups with other members and generate a signature on behave of the group. The members are unaware of the inclusion in the signature generation process. No setup stage or member joining procedure is required. No trusted third party is present so that the anonymity of the actual signer is irrevocable. Unconditional protection on anonymity is provided so that neither insider nor outsider is able to find out the actual signer.

We introduce two SAG signature variants by adding the blindness and linkable properties. These signature variants provide different anonymity properties than the ordinary SAG signatures. Our work provides different anonymity requirements: *(blindness)* the blind signer's identity is unconditionally secure, and the blind signer cannot point out whether a given message-signature pair is generated by himself or not. *(linkability)* the sender identity is computationally secure, where it'll be able to link the identity on different messages and if someone misbehaves.

Chapter 1 we give introduction to cryptography. Chapter 2 we provide the

preliminaries in cryptography and the mathematics. Then in chapter 3 we go through the basics and the security models of digital signatures. Chapter 4 we give introduction to spontaneous group signatures. In chapter 5 we present our results on blind SAG signature, our constructions, and the security proofs. Finally, we present our results on separable linkable SAG signature with a thresholding option in chapter 6.

Acknowledgments

I would like to take this opportunity to express my gratitude to some people. First of all, this thesis would not be completed without the help from my supervisor, Prof. Victor Wei, for his guidance and support during my graduate studies. I would also like to thank my colleague: Mr. Siu-Ting Ho and Mr. Adrian Sai-Wah Tam, who enriched me with knowledge and provide valuable help on every aspect in my study. Next I would like to express my sincere gratitude to laboratory technician Mr. Hung-Kwong Yip who deserves thanks for giving pleasurable moments during the laboratory sessions.

I have been happy to affiliate with the Department of Information-Engineering, and would like to thank all professors, staffs and friends for their encouragement. Special thanks will be given to my colleagues in the Information Security Laboratory: Mr. Patrick Pak-Kwong Tsang, Mr. Allen Man-Ho. Au, Mr. Tsz-Hon Yuen, Mr. Patrick Pak-To Chan, Mr. Robert Siu-Kuen Leung, Mr. Sebastian Fleissner, Miss Rosanna Yuen-Yan Chan, Prof. Joseph Kai-Sui Liu for fruitful discussions and useful advises.

Finally I would like to mention one people who mean a lot to me. With heartfelt thanks to Miss Kar-Yin Fung who always there for me, for her patience, care and love.

Contents

Chapter 1

Introduction

Since the bloom of digital communication in the 90s, the way of information manipulation has changed. Internet connects people around the world and provides an efficient communication channel, which has never been before. Sending transatlantic letter once need days or months by postal services; now can be done in just seconds by electronic means. Taking the advantage of efficiency and convenience, global corporations and medium size enterprise make good use of Internet to conduct business. Meanwhile, the network also provides means for the attackers to acquire valuable and private information. Security concerns have been playing a crucial part in protecting our information today.

Cryptography is the science to make a message of record incomprehensible to unauthorized persons and also methods to achieve security properties such as confidentiality, authentication, integrity and non-repudiation. Confidentiality is the service to keep message secret from all but authorized receiver. Authentication provides as-certainty the origin of the information. Integrity ensures a message is not altered during transmission. For non-repudiation, a person cannot falsely deny the action that one has performed. Anonymity is another important properties, which hides the identity of an entity to a certain degree. The importances of these properties are growing rapidly as the demand for privacy in the society is rising during these years.

Communication privacy is to hide the identity of the sender/receiver. Nowadays message confidentiality can be easily achieved by using cryptographic techniques such as public key encryption (RSA, ElGamal). However little progresses have been made to protect the identity of the communication parties until the last 10 years.

Group signature[4] and ring signature[5] are some of the most promising solution in providing identity protection, while preserving the authentication property. Both ring signature and group signature scheme allows a member of group to sign a message on behave of the group which preserving their anonymity, while group signature offers a option to revoke the identity of the signer in case of dispute.

Traditional group signature schemes require a setup stage for a special key distribution and in addition a member joining protocol. The requirement of the setup stage incurs processing, and limits the usefulness of those schemes in case where the group membership is highly dynamic. Some scheme may even require a powerful trusted third party to perform the key distribution process, whom may have too much power over the group. Due to the inefficiency of those schemes, *Spontaneous Anonymous Group (SAG) signature* is proposed. For the spontaneous properties, there is no setup stage, no trusted third party required. Also the group formation is dynamic, users can form a group without any collaboration from other group members or any other entities. The *anonymous* properties of SAG cryptography protects privacy and achieves 1-out-of-n anonymity, i.e. no person can figure out the actual signer. SAG signature provides exculpable anonymity as the anonymity is unconditionally irrevocable.

The authentication and anonymity properties of Spontaneous Anonymous Group signature are perfect ingredient to authenticate a message in any ad-hoc anonymous systems such as mix-net and onion routing. By introducing properties such blindness and linkability into SAG signature, we achieved to construct two SAG signature variants with different anonymity levels:

- *Blind SAG Signature*, which provides perfect identity protections and message untraceability. All parties except the signature requester, are unable to figure out who signer the Blind SAG signature. Message-signature relationships and signature-signer relationships are unlinkable during and after the protocol.
- *Linkable SAG Signature*, which provides partial identity protections. Given two linkable SAG signature, one can tell whether these two signatures are generated by the same actual signer.

Here we compare the anonymity level provided by these SAG variants in figure 1.1. Blind SAG signature provides the best anonymity protection among all with its unconditional signer anonymity and message untraceability; While the linkable SAG signature provides a anonymity protection level which lies between the group signature and the ordinary SAG signature due to the partial information provided by the linkable property.

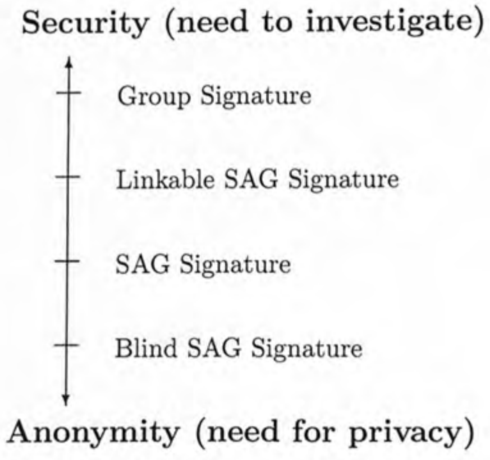

Figure 1.1: Security vs Anonymity for Digital Signature Schemes

Details of our blind SAG signature will be discussed in Chapter 5 and Linkable SAG signature in Chapter 6.

Chapter 2

Preliminaries

In this chapter, we describe the mathematical background and theories on cryptography and provable security. For this purpose, we first state the notation used in this thesis in chapter 2.1. Next we discuss the basic primitives of cryptography in section 2.2. We are then ready to deal with provable security and security model in Section 2.3. For more details about provable security, the reader is referred to [6, 7];

2.1 Notation

The following notation will be used throughout the article:

- $\{0,1\}^*$ denotes the space of finite binary strings, strings are finite unless we say otherwise.
- $a||b$ denotes the concatenation of strings a and b
- "PPT" stands for "probabilistic, polynomial time"
- If $\mathcal{A}$ is a probabilistic algorithm then, for any inputs $x, y, \cdots$ the notation $\mathcal{A}(x,y)$ refers to the probability space which to the string σ, assigns the probability that $\mathcal{A}$, on input $x, y, \cdots$, output σ.

- $< P, V > (x)$ denotes the output of the interactive machine V after interacting with the interactive machine P on common input x. If the interactive machine V is probabilistic, then it denotes the random variable of the output V.

2.2 Cryptographic Primitives

2.2.1 Symmetric Key Cryptography

Symmetric key cryptography is the earliest form of cryptography which can be aged to several thousands years ago. People have been attempting to conceal certain information that they wanted to keep secret by substituting parts of the information with symbols, numbers and pictures.

Definition 2.1. *Symmetric Key Cryptosystem*

Let $\mathcal{M}$ be the message space, $\mathcal{C}$ be the ciphertext space and $\mathcal{K}$ be the key space. A symmetric key cryptosystem consists the followings:

1. *an efficient probabilistic generation algorithm*

$$\mathcal{G} : 1^\lambda \rightarrow \mathcal{K}$$

2. *an efficient encryption algorithm*

$$\mathcal{E} : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$$

3. *an efficient decryption algorithm*

$$\mathcal{D} : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$$

where 1^k denotes arbitrary string with length k.

For $k \in \mathcal{K}$ and $m \in \mathcal{M}$, we denote $\mathcal{E}(m,k)$ by $\mathcal{E}_k(m)$ and $\mathcal{D}(c,k)$ by $\mathcal{E}_k(c)$. It is required that for all $m \in \mathcal{M}$ and $k \in \mathcal{K}$, $\mathcal{D}_k(\mathcal{E}_k(m)) = m$. Symmetric key cryptosystem can be classified into two categories: block cipher and stream cipher. For block ciphers messages are divided into blocks with predetermined block size, and the encryption algorithm handles each block independently; Stream ciphers handle the message 1 bit at a time.At the time of writing , the standard block ciphers are AES[8] and 3DES , and the standard stream cipher is RC4 [9].

2.2.2 Asymmetric Key Cryptosystem

Unlike symmetric key cryptosystem, asymmetric key cryptosystem allows the encryption key and the decryption key to be different. Syntactically, an asymmetric key cryptosystem can be defined as follow:

Definition 2.2. *Asymmetric Key Cryptosystem*

Let $\mathcal{M}$ be the message space, $\mathcal{C}$ be the ciphertext space and $\mathcal{K}$ be the key space. An asymmetric key cryptosystem consists of the followings:

1. *an efficient key generation algorithm*

$$\mathcal{G} : 1^\lambda \mapsto (\mathcal{K}, \mathcal{K}')$$

2. *an efficient encryption algorithm*

$$\mathcal{E} : \mathcal{M} \times \mathcal{K} \mapsto \mathcal{C}$$

3. *an efficient decryption algorithm*

$$\mathcal{D} : \mathcal{C} \times \mathcal{K}' \mapsto \mathcal{M}$$

where 1^k denotes arbitrary string with length k.

For $k, k' \in \mathcal{K}$ and $m \in \mathcal{M}$, we denote $\mathcal{E}(m, k)$ by $\mathcal{E}_k(m)$ and $\mathcal{D}(c, k')$ by $\mathcal{E}_{k'}(c)$. It is required that for all $m \in \mathcal{M}$ and $k \in \mathcal{K}$, there exist a $k' \in \mathcal{K}$ such that $\mathcal{D}_{k'}(\mathcal{E}_k(m)) = m$.

Throughout the thesis we denote the publicly know encryption key k be PK, and the keep secret decryption key be SK. At the time of writing the standard asymmetric key cryptosystems includes RSA [10], and Elliptic Curve Cryptosystem (ECC).

2.2.3 Secure Hash Function

A secure hash function is a deterministic function which maps a bit string of an arbitary length to a hashed value which is a bit string of a fixed length ℓ.

$$H : (0,1)^* \mapsto (0,1)^\ell$$

A **secure** hash function needs to satisfy the following properties:

- *[Pre-Image Resistance]*

  Given $y \in (0,1)^\ell$, is it computationally infeasible to find x such that $H(x) = y$;

- *[Weak Collision Resistance]*

  Given x, It is computationally infeasible to find distinct x' with $x' \neq x$ such that $H(x) = h(x')$;

- *[Strong Collision Resistance]*

  It is computationally infeasible to find distinct x and x' such that $h(x) = H(x')$.

In the conference of Crypto04, Biham and Chen[11] proposed a new hash attack targeting SHA-0, where MD5 and SHA-1 is depended on. Therefore these hash functions are at risk and should not be used. Stronger hash functions such as SHA-256 and SHA-512 are proposed and their resistance to this new attack is still unknown.

2.2.4 Digital Signature

Digital signature, likes it handwritten counterpart, provides a mean to bind the signer's identity together with a message, and moreover it protects the integrity of the message from being tamper. There's many types of signature with different structures and properties (e.g. Group Signature, Ring Signature), and in general they are two parties involved:

1. **Signer** generates a signature from a message with his secret key. Then he pass the message and the signature to a verifier.

2. **Verifier** check the signature to determine whatever it is legitimate.

Also a digital signature scheme must possess the following properties:

1. **Authentic** The signature can convince the verifier that the signer deliberately signed the document.

2. **Unforgeable** Only the signer, but no one else is able to produce the signature.

3. **non-Reusable** The signature only bind the signer's identity to the particular document. No person can associate the signature to a different document.

4. **non-repudiatable** The signer cannot deny having sign on the document once he did it.

2.2.5 Digital Certificate and Public Key Infrastructure

Digital certificate is a digital signature issued by a trusted certificate authority(CA), which contains the information of the owner,his public key and certificate authority information. They do two things:

1. They authenticate their owner are who they claimed to be.

2. They protects confidentiality by preventing Man-in-the-middle attack.

Certificate Authority is usually a public trustworthy organizations such as Verisign, VISA.

Public Key Infrastructure (PKI) is a integration of certificate authority, digital certificate and public key cryptography to provide confidentiality and integrity. It forms a hierarchical of trust by delegating trust along a chain. For example a company can apply a company certificate from a certificate authority. Then the company can issue certificates for their employee. Since there is a chain of trust along the certificates, verifiers can assure the identity of the employee even the verifier only knows about the certificate in use.

2.3 Provable Security and Security Model

Since the early 80's the concept of provable security have evolved. By assuming the existence of **one-way** functions (Section. 2.3.2), security of a system can be proved by reduction. The Rabin encryption [12] is an early example that had a reductionist security property in which he claims if someone who can find even a small percentage of the messages from the ciphertext must also be able to factor n, which is considered as an hard problem.

In this section we will first go through the mathematics required, following some important cryptographic theories will be introduced. At the end of this section we'll go through the security model we are going to used in our security proofs.

2.3.1 Mathematics Background

Here we define the probability ensembles that are commonly used in security proofs.

Definition 2.3. *A function $f : \mathbf{N} \rightarrow \mathbf{R}^+$ is **negligible** if for all $c \in \mathbf{N}$ there* *exists $k_c \in \mathbf{N}$ such that for all $k > k_c$, $f(k) < k^{-c}$. A function is non-negligible if it is not negligible.* □

For two random variable X and Y over a finite domain D we let the **statistical difference** be

$$SD(X,Y) = 1/2 \sum_{d \in D} |Pr[X = d] - Pr[Y = d]|$$

A **distribution ensemble** is a sequence $\{X_n\}_{n \in \mathbf{N}}$ where each X_n is a random variable with finite domain D.

Definition 2.4. *For two distribution ensemble X and Y with finite domain D we say they are* ***computationally indistinguishable*** *(written as $X \overset{c}{\approx} Y$) if for all probabilistic polynomial time algorithm $A\colon D \to \{0,1\}$ such that $|Pr[A(X_k) = 1] - Pr[A(Y_k) = 1]|$ is negligible . We say they are* ***statistically close*** *(written as $X \overset{s}{\approx} Y$) if $k \mapsto SD(X_k, y_k)$ is negligible. We say they are* ***identical*** *(written as $X = Y$) if for all $k \in \mathbf{N}$ and $d \in D$, $Pr[X_k = d] = Pr[Y_k = d]$.* □

Definition 2.5. *An* **NP***-relation R is a relation over bitstrings for which there is an efficient algorithm to decide whether $(x, z) \in R$ in time polynomial in the length of x. The* **NP***-language L_R associated to R is defined as $L_R \doteq \{x | (\exists z)[(x, z) \in R]\}$.* □

It is easy to show that if $X \overset{c}{\approx} Y$ and $Y \overset{c}{\approx} Z$, then $X \overset{c}{\approx} Z$. If $X \overset{s}{\approx} Y$ and $Y \overset{s}{\approx} Z$, then $X \overset{s}{\approx} Z$. If $X = Y$ and $Y = Z$, then $X = Z$. If $X = Y$, then $X \overset{s}{\approx} Y$, which in turn implies $X \overset{c}{\approx} Y$.

2.3.2 One-Way Function

Before introducing the one-way function, we need the following definition regarding time required for computing a problem:

Definition 2.6. ***Efficient Algorithms*** *An algorithm is said to be* ***efficient*** *if its execution time can be expressed by a polynomial in the size of the input.* □

Definition 2.7. ***Easy Problem*** *An problem is said to be* ***easy*** *if there exist a probabilistic polynomial time algorithm which can compute with non-negligible success probability.* □

Definition 2.8. ***Hard Problem*** *An problem is said to be* ***hard*** *if there does not exist any probabilistic polynomial time algorithm which can compute with non-negligible success probability.* □

Most of the public key cryptosystem depends on one-way functions and one-way trapdoor functions. Informally, a function $f\colon X \to Y$ is said to be **one-way** if:

1. Given $x \in X$, there exist efficient algorithm to compute $f(x)$.
2. Given $y \in Y$, it is hard to find an x such that $f(x) = y$.

Definition 2.9. ***One-Way Function*** *A function f is said to be* ***one-way*** *if for any probabilistic polynomial time function g.*

$$x \overset{r}{\leftarrow} D, Pr[f(g(f(x))) = f(x)] \textit{ is negligible in } n$$

. where x is draw uniformly from D and has length n and all numbers of length n are equally likely. □

The existence of one-way function is still unproved. The existence of one-way function implies many other cryptography primitives such as pseudorandom generator[13], even more an instance of one-way function can imply the complexity class $P \neq NP$. Many literature proposed candidate one-way functions.

One-way trapdoor function, which is a variant of one-way function, have the following properties:

Definition 2.10. *A one-way trapdoor function, which we denote by $f_t(x)\colon \mathcal{D} \rightarrow \mathcal{R}$, is a one-way function, i.e. it is easy to easy to evaluate for all $x \in \mathcal{D}$ and difficult to invert for almost all values in $\mathcal{R}$. However, if the trapdoor information t is used, then for all values $y \in \mathcal{R}$ it is easy to compute $x \in \mathcal{D}$ satisfying $y = f_t(x)$.* □

In the following we will go through some candidate one-way functions.

2.3.3 Candidate One-way Functions

Discrete logarithm is one of the major component of many cryptosystem in use today. This problem was originated by Diffie and Hellman.

Discrete Logarithm Problem

Definition 2.11. Discrete Logarithm Problem (in $\mathbf{Z}_p^*$) Given a prime p, a generator g of $\mathbf{Z}_p^*$ and an element $y \in \mathbf{Z}_P^*$, find an integer x such that $g^x \equiv y \bmod p$. □

A group G is said to be finite if and only if there is an element $g \in G$ such that for all element $a \in G$, there exist an integer i satisfying $g^i = a$. g is called the generator of G. DLP is a special case in which the group G is $\mathbf{Z}_p^*$ where p is prime.

There's a generalization of the DLP problem from a single basis to a set of baais. The generalized discrete logarithm problem (GDLP) can be stated as follow:

Definition 2.12. Generalized Discrete Logarithm Problem (in finite abelian group G) Given a finite abelian group G, generated by n linearly independent elements $\{g_1, \cdots, g_n\}$ of prime order ℓ., and a element y in G. find an integers $c_1, \cdots, c_n$ such that $\prod_{i=1}^{n} g_i^{c_i} \equiv g$. □

Assumption 2.13. ***Discrete Logarithm Assumption*** *The discrete logarithm problem (DLP) is hard.*

Assumption 2.14. ***Generalized Discrete Logarithm Assumption*** *The generalized discrete logarithm problem (GDLP) is hard.*

DLP is believed to be hard. Many existing cryptosystems depends on the intractability of the discrete logarithm, for example, the El-Gamal Encryption reduces its security to the DLP problem.

Computational Diffie-Hellman Problem

the Computational Diffie-Hellman problem(CDHP) is proposed by Diffie and Hellman in [14].

Definition 2.15. Diffie-Hellman Problem Given a group G,and three elements $g, g^a, g^b \in G$, compute g^{ab}. The advantage ϵ of a PPT algorithm $\mathcal{A}$ is defined by

$$\epsilon = Pr[g^a b \leftarrow \mathcal{A}(G, g, g^a, g^b)]$$

□

Decisional Diffie-Hellman Problem

Definition 2.16. Diffie-Hellman Problem Given a group G,and four elements $\{g, g^a, g^b, g^c\} \in G$, determine whatever $c \overset{?}{=} ab$. The advantage ϵ of a PPT algorithm $\mathcal{A}$ is defined by

$$\epsilon = Pr[0 \leftarrow \mathcal{A}(G, g, g^a, g^b, g^c) \wedge c \neq ab] + Pr[1 \leftarrow \mathcal{A}(G, g, g^a, g^b, g^c) \wedge c = ab]$$

□

Gap Diffie-Hellman Group

Gap Diffie-Hellman (GDH) group is a group where the Computational Diffie-Hellman(CDH) problem is hard but the Decisional Diffie-Hellman (DDH) problem is easy. The existences of such groups are shown in [15, 16, 17].

Definition 2.17. ***Gap Diffie-Hellman (GDH)*** *A group G with order q is said to be be GDH if there exist a probabilistic polynomial time algorithm to solve the DDH problem in G and no probabilistic polynomial time algorithm in q to solve CDH problem with non-negligible probability of success.* □

Integer Factorization Problem

The factoring problem is one of the oldest in number theory. It's simple to factor a number, but its time-consuming. Many cryptography elements such as RSA depends on the intractability of factoring larger integers.

Definition 2.18. ***Integer Factorization Problem*** *Given a positive integer n, find primes p_i and positive integers r_i such that $n = \prod p_i^{r_i}$* □

Many algorithms have been devised for determining the prime factors of a given number. At the time of writing the fastest algorithm is *Number Field Sieve* which takes time in $O(e^{c(1/3 \log n)(2/3 \log\log n)})$ with $c \approx 1.9$.

RSA problem

RSA problem [10] was proposed by Rivest, Shamir and Adlemen in 1978. Among all public-key cryptosystems RSA is by far the easiest to understand and implement. RSA gets its security from the difficulties of factoring large numbers.

The RSA function is a One-way trapdoor function and is as follows:

Definition 2.19. ***RSA Function*** *Given a RSA integer $n = pq$, which a product of two distinct large primes p and q, and let e be the integer such that*

$gcd(e, \phi n) = 1$. The RSA function is

$$f_{n,e}(x) \mapsto x^e \bmod n$$

The trapdoor information is d where $ed \equiv 1 \bmod (\phi(n))$. □

The RSA function is easy to compute but difficult to invert without the knowledge of d. This function is assumed to be a hard problem.

Definition 2.20 (RSA Problem). *Let $n = pq$, where p, q are primes and $|p| = |q| = k$. For every polynomial Q and every Polynomial Time Machine, A, there exists an integer k_0 such that $\forall k > k_0$.*

$$Pr[A(n, e, f_{n,e}(x)] < \frac{1}{Q(k)}$$

□

Strong RSA Problem

Definition 2.21 (Strong RSA Problem). *[18, 19, 20]] Given a safe prime product N, and $z \in QR(N)$, find $u \in \mathbf{Z}_N^*$ and $e > 1$ such that $u^e = z(\text{mod } N)$* □

2.4 Proof Systems

2.4.1 Zero-knowledge Protocol

Loosely speaking, a zero-knowledge protocol is a proof system which have the property of being convincing and yielding nothing beyond the validity of the asssertion. For example, you can proof you know a discrete logarithm of $y = g^x$ (mod p) to a verifier without letting the verifier knows about x. Also the verifier is convince that you actually know x with overwhelming property.

We followed the definition of zero-knowledge from [21]:

Definition 2.22. ***Perfect Zero-Knowledge*** *Let (P, V) be an interactive proof system of some language L. We say that (P, V) is* ***perfect zero-knowledge*** *(PZK) if for every probabilistic polynomial time interactive machine V there exist a probabilistic polynomial time algorithm M such that for every $x \in L$ the following two conditions hold:*

1. *With probability at most $1/2$, on input x, machine M output* ***fail****;*
2. *Let $m(x)$ be a random variable describing the distribution of $M(x)$ conditioned on $M(x) \neq$* ***fail****, then the following random variables are identically distributed:*
   - *$< P, V > (x)$ the output of the interactive machine V after interacting with the interactive machine P on common input x;*
   - *$m(x)$ the output of machine M on input x, conditioned not being* ***fail***

M is called a ***perfect simulator*** *for the interaction of V with P* □

Loosely speaking, perfect zero knowledge guarantee after the completion of the protocol, only one bit of information is leak to the verifier(the assure of the proof statement). However this perfect zero-knowledge is slightly too strict for the application on cryptography. The following two definition are relaxed form of PZK.

Definition 2.23. ***Statistical Zero-Knowledge*** *Let (P, V) be an interactive proof system of some language L. We say that (P, V) is* ***statistical zero-knowledge*** *(SZK) if for every probabilistic polynomial time interactive machine V there exist a probabilistic polynomial time algorithm M such that the following two ensembles are statistically close as functions of size of x:*

1. *$\{< P, V > (x)\}_{x \in L}$ The output of the interactive machine V after it interacts with the interactive machine P on common input x;*

2. *$\{M(x)\}_{x \in L}$ The output of the machine m on input x;*

That is, the statistical difference between $< P, V > (x)$ and $M(x)$ is negligible in terms of size of x. □

Definition 2.24. ***Computational Zero-Knowledge*** *Let (P, V) be an interactive proof system of some language L. We say that (P, V) is* ***computational zero-knowledge*** *(CZK) if for every probabilistic polynomial time interactive machine V there exist a probabilistic polynomial time algorithm M such that the following two ensembles are computationally indistinguishable:*

1. *$\{< P, V > (x)\}_{x \in L}$ The output of the interactive machine V after it interacts with the interactive machine P on common input x;*

2. *$\{M(x)\}_{x \in L}$ The output of the machine m on input x;*

M is called a ***simulator*** *for the interaction of V with P* □

Loosely speaking, perfect zero knowledge guarantee after the completion of the protocol, only one bit of information is leak to the verifier(the assure of the proof statement). However this perfect zero-knowledge is slightly too strict for the application on cryptography. The following two definition are relaxed form of PZK.

2.4.2 Proof-of-Knowledge Protocol

A **PoK (Proof-of-Knowledge)** is a three-move interactive protocol consisting of (Prover, Verifier). Common input consists of a public key, PK. Prover has the additional input SK. The three moves are $\mathcal{K}=(\mathcal{T}, \mathcal{C}, \mathcal{S})=($*commit*, *challenge*, *response*$)$.

The security requirement of a proof-of-knowledge protocol are:

1. *Completness* means, with all sides honest, results are as they should be.

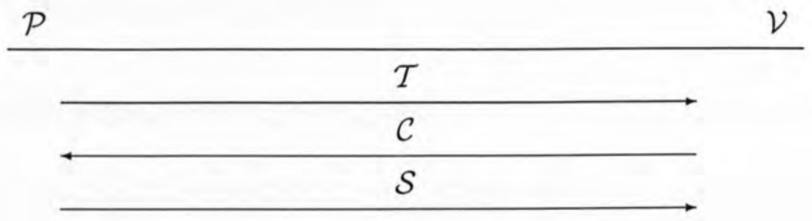

Figure 2.1: Proof of Knowledge (PoK)

2. *Soundness* means two random challenge-response pair to the same commitment result in witness extraction.

3. *Special Soundness* means: any two challenge-response pair with the same commitment result in witness extraction.

2.4.3 Honest-Verifier Zero-Knowledge (HVZK) Proof of Knowledge Protocols (PoKs)

In a **Honest-Verifier Zero-Knowledge** protocol, if the verifier honestly follows the protocol instruction, then the protocol is perfect zero-knowledge. Informally, an **honest-verifier zero-knowledge (HVZK) proof of knowledge (PoK) protocol** for an **NP**-relation R is an two-party protocol such that a prover proves his/her knowledge of a witness x of z in R.

Every HVZK proof can be turned into a signature scheme by setting the challenge to the hash value of the commitment together with the message to be signed [22]. Such a scheme is proven secure by [23] against existential forgery under adaptively chosen message attack [24] in the random oracle model [25]. Following [26], we call these signature schemes "signatures based on proofs of knowledge", SPK for short. Note that there always exists a corresponding

HVZK PoK protocol for every SPK.

2.5 Security Model

2.5.1 Random Oracle Model

The idea of Random Oracle Model (ROM) first appeared in [27] and was later formalized by Bellare and Rogaway [6] . The goal of this protocol is to bridge the disparity betwen the theoreticians' and practioners' view on cryptographic primitives. We capture some key properties of the ROM as follows:

1. the Random Oracle is defined as $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\infty$
2. $\mathcal{H}$ is collision resistant, i.e. it is hard to get distinct x and y such that $\mathcal{H}(x) = \mathcal{H}(y)$
3. $\mathcal{H}$ is well defined: Given $x = y$, then $\mathcal{H}(x) = \mathcal{H}(y)$
4. $\mathcal{H}$ is a random mapping such that given a input x where x has not been queried before, then no party can guess the output of $\mathcal{H}(x)$ correctly with non-negligible probability. Similarly it is hard to guess the preimage for a given value.
5. All hash function computation must be made via the random oracle. No parties in the model can distinguish the values generated by $\mathcal{H}$ from the real hash function.

Under the random oracle model, hash functions are assumed to act as random function (denoted as random oracle $\mathcal{H}$) and publicly accessible by all parties including adversary. After proving correct a protocol in a model where the parties have access to a random oracle and then instantiating that oracle with an appropriate cryptographic primitive such as MD5 or SHA-1. However it is obvious that no hash function would act like a random oracle in reality(all hash function are deterministic), and therefore a protocol which is proven secure in a random oracle model may not be instantiable or even insecure in practice[28, 29]. Despite the strong assumption requirements, the random oracle model provides a foundation for a mathematical proof in protocol security, and till the time of wriing no practical protocol proven secure under the random oracle model hash been broken when used with a "good" hash function, such as SHA-1. For more details on the random oracle model, please refer to [6, 28, 29].

2.5.2 Generic group model (GGM)

We will use the generic group model of [30, 31, 7]. Some highlights below.

Only a restricted set of operations are allowed. They include random generation of integers and group elements, group computations, exponentiations, equality tests. There are only two data types: *group elements* and *non-group data.*

It is assumed the the discrete logarithm problem is uncomputable in the GGM[30].

We restrict ourselves to a polynomial number of steps. Therefore, there are only a polynomial number of unassociated group elements base g, public keys $y_1, \cdots, y_n$, commitments $t_1, \cdots, t_{q_B}$, randomly generated group elements $u_1, \cdots, u_{q_G}$. The computation transcript at each step τ consists of

$$f_\tau = g^{a_{\tau,-1}} \prod_i y_i^{a_{\tau,i}} \prod_{i'} t_{i'}^{b_{\tau,i'}} \prod_{i''} u_{i''}^{c_{\tau,i''}} \tag{2.1}$$

Each computation can only depend on parameters in existence before that step, resulting in zero exponent for parameters that come into existence after that step.

Chapter 3

Signature Scheme

3.1 Introduction

In public key cryptography, one can use his/her private to encrypt a message and the resultant ciphertext can be decrypted back to the original message using the one's public key. If the ciphertext is altered in any manner then the decryption process will not yield the original message, and thus the ciphertext can play the role of a manipulation detection code (MDC). The MDC provides data integrity protection from the message, since anybody can obtain the corresponding public key to decrypt and compare the result to the accompanied message, verifying the correctness of the ciphertext. Moreover, since only the owner of the public key (who possess the private key) have the ability to construct the MDC, the MDC also provided a mean to bind a person to the message. Thus, this usage of public key cryptography can model precisely the property of a signature, i.e. a digital signature.

Syntactically a digital signature scheme can be described as follow:

Definition 3.1. *A digital scheme consists of the following attributes:*

- *a plaintext message space* $\mathcal{M}$: *a set of finite strings over some alphabet;*
- *a signature space* $\mathcal{S}$: *a set of possible signature;*

- *a signing key space $\mathcal{K}$: a set of possible keys for signature creation;*
- *a verification key space $\mathcal{K}'$: a set of possible keys for signature verification;*
- *an (probabilistic) polynomial time key generation algorithm*

$$\textbf{\textit{KEYGEN}}\colon 1^\lambda \rightarrow \mathcal{K} \times \mathcal{K}'$$

- *an (probabilistic) polynomial time signing algorithm*

$$\textbf{\textit{SIGN}}\colon \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{S}$$

- *an (probabilistic) polynomial time verification algorithm*

$$\textbf{\textit{VERIFY}}\colon \mathcal{M} \times \mathcal{S} \times \mathcal{K}' \rightarrow \{\textbf{\textit{accept}}, \textbf{\textit{reject}}\}$$

For any secret key $sk \in \mathcal{K}$ and message $m \in \mathcal{M}$, we denote $s \leftarrow \mathcal{S}_{sk}(m)$ the signature transformation and read it as "s is a signature of m created using the key sk."

For any secret key $sk \in \mathcal{K}$, the corresponding public key $pk \in \mathcal{K}'$, and any message $m \in \mathcal{M}$, it is necessary

$$\textbf{\textit{VERIFY}}_{pk}(m,s) = \begin{cases} \textbf{\textit{accept}}, \text{ with probability } 1 & \text{if } s \leftarrow \mathcal{S}_{sk}(m); \\ \textbf{\textit{reject}}, \text{ with overwhelming probability} & \text{if } s \not\leftarrow \mathcal{S}_{sk}(m). \end{cases}$$

where the probability space is taken over $\mathcal{S}, \mathcal{M}, \mathcal{K}$ and $\mathcal{K}'$. □

Note that the integer λ of the key generation algorithm **KEYGEN** is called the security parameter, which provides the size of output signing/verification keys. Since the **KEYGEN** is efficient with running time polynomial time in the size of its input, the input value should be unary encoded.

In the following sections we'll explore the security of a signature schemes.

3.2 Security Notation for Digital Signature

In order to discuss about the security of the digital signature scheme, we need to first define the powerfulness of the adversary and the degree of severity of success in forging a signature. Here we classify the attack in ascending severity.

1. **Key-Only Attack:** Only the verification key is available to the adversary. The adversary is not able to get any signature samples to study.

2. **Known Signature Attack:** In addition to the verification key, the adversary collects a number of message-signature pairs chosen from the legal signer.

3. **Chosen Message Attack:** The adversary is allow to ask the legal signer to sign message chosen by the adversary.

The severity of success in breaking a signature scheme is listed as follows in ascending order:

1. **Existential Forgery:** The adversary can forge a signature of some message, not necessary of his choice.

2. **Selective Forgery:** The adversary can succeed in producing a signature of some of the messages of his choice.

3. **Universal Forgery:** Without knowing the secret key the adversary can produce signature of any message of his choice.

4. **Total Break:** The adversary can compute the secret key of the signer based on the public key and chosen signature-message pairs.

3.3 Security Proof for Digital Signature

The security of digital signature depends on **unforgeablility**. In provable security, security proofs are usually done with "reduction-to-contradition" philosophy. By assuming a particular problem is hard (e.g. Discrete Log Problem), then the reduction of the signature forging problem to that hard problem will yield a contradiction if the signature forging is easy.

In our thesis all signature scheme will be proved under the strongest adaptive chosen-message attack in the random oracle model.

3.3.1 Random Oracle Model for Signature Scheme

In an random oracle model based technique for security proof, both random oracles and a special agent called simulator will be used. The simulator controls every random oracle used in the model, and therefore everyone in the random oracle are unwittingly making random oracle query to the simulator. In addition, the simulator should able to simulate these oracles perfectly in probabilistic polynomial time. Typically a signature scheme consist of two random oracles:

- **Hash Oracle** $\mathcal{H}$: Given a string x of finite length, output random hash value $\mathcal{H}(x)$. This models the hash function used in reality.

- **Signing Oracle** $\mathcal{SO}$. Given a public key pk, a message m, output a signature σ such that $\mathbf{verify}(\sigma, m, pk) = \mathbf{accept}$. This models the ability that the adversary can obtain signature-message pairs with any message of its choice.

3.3.2 Adaptive Chosen Message Attack

The existential forgery under adaptive chosen message attack is the strongest security for digital signature. All our signature schemes proposed in this thesis are proven secured against existential forgery under adaptive chosen message attack in the random oracle model. Here we provide an informal definition for the game of adaptive chosen-message attack (adaptive CMA):

Definition 3.2. ***Existential Forgery under Adaptive Chosen-Message Attack*** *Let k be a positive integer. An adaptive forger against a signature scheme (**KEYGEN**, **SIGN**, **VERIFY**) is a probabilistic polynomial time (in k) algorithm. It takes an input public key pk, where $(pk, sk) \leftarrow \textbf{\textit{KEYGEN}}(1^k)$, and tries to forge signatures with respect to pk. The forge is allowed to request, and obtain, signatures of messages of its choice. This is modeled by allowing the forger to access to the signing and hash algorithms, both polynomial (in k) times.*

*The forger is said to $(t(k), Adv(k))$-break the signature scheme if in polynomial time $t(k)$ with non-negligible probability $Adv(k)$ it outputs a valid forgery (m, s) such that $\textbf{verify}(s, m, pk) = \textbf{accept}$ where m is a recognizable message according the hash functions used in the scheme but is not one which has been input to a **sign** earlier by the signer.*

3.4 Schnorr Identification and Schnorr Signature

Schnorr identification protocol is a basic building block for many cryptographic protocols. The security of this protocol can be reduced to the discrete logarithm problem. Also the Schnorr identification possess honest verification zero-knowledge properties, which means that if the verifier honestly follows the protocol instruction, then the protocol is perfect zero-knowledge.

Definition 3.3. Schnorr's Identification Protocol Common Input:

- p, q: two primes satisfying $q|p-1$;

- g: with $ord_p(g) = q$;
- y: $y = g^{-a} \pmod p$

We denote the tuple (p, q, g, y) be Alice's public key.

Private Input of Alice: $a < q$;

Protocol:

1. Alice picks a random number $r \leftarrow \mathbf{N}$ and compute $Commit \leftarrow g^r \pmod p$. she send $Commit$ to Bob;
2. Bob picks $Challenges \leftarrow \; bfZ_q$ and send $Challenge$ to Alice;
3. Alice compute $Response \leftarrow r + aChallenge \pmod q$ and send $Response$ to Bob;
4. Bob checks $Commit \overset{?}{=} g^{Response}y^{Challenge} \pmod p$; He abort and rejects if the checking shows error, othervise Bob accepts.

The Schnorr identification protocol can be converted to a digital signature scheme by the following transformation:

$$Challenge \leftarrow H(Message, Commit)$$

This transformation method is called Fiat-Shamir Transformation [32].

3.4.1 Schnorr's ROS assumption

Schnorr [7] presented a then-new algorithm to compute the parallel one-more forgery of Schnorr (resp. Okamoto-Schnorr) blind signatures. He showed the equivalence of the parallel one-more unforgery of those two blind signatures and the ROS Problem, in the random oracle model plus the generic group model. His technique also applied to many other blind signatures. In this paper, we will use the following form of Schnorr's ROS Problem:

The ROS Problem: Given $1 \leq q_B \leq q_H$, typically $q_B << q_H$, and all computations are in Z_q. Compute a $q_H \times q_B$ matrix $\mathbf{A}$, such that the probability of computing the following problem is non-negligible:

> Given random $\hat{c} = [\hat{c}_1, \cdots, \hat{c}_{q_H}]$, compute $J \subset \{1, \cdots, q_H\}$ with $|J| = q_B + 1$, $j_0 \in J$, $\{\alpha_j : j \in J\}$ with $\alpha_{j_0} \neq 0$, and β such that $\sum_{j \in J} \alpha_j [\mathbf{A}_j, \hat{c}_j] = \beta$ and $\{\mathbf{A}_j : j \in J \setminus \{j_0\}\}$ are linearly independent.

Note $\mathbf{A}_j$ denote the j-th row vectors of $\mathbf{A}$, and $[\mathbf{A}_j, \hat{c}_j]$ denotes the lengthened vector by one more entry $\hat{c}_j$.

3.5 Blind Signature

Blind signature scheme protects the confidentiality of the message, i.e. the signer will not know about the content that one is going to sign. Consider someone give you a envelop which contains a carbon copy paper and a unknown message. When you sign the envelope on the outside, the carbon copy paper will imprint your signature over the message.

The blind signature is a key to many cryptographic services which provides anonymity and privacy, for example.

A **blind signature** consists of the tuple (BlindSigner, Warden, Verifier) where the three components form an interactive protocol as follows:

1. Common input to all three parties: PK. Additional input to BlindSigner: SK.
2. BlindSigner sends t' (commitment) to Warden.
3. Warden sends t to Verifier.
4. Verifier sends message m to Warden.

5. Warden sends c' to BlindSigner.

6. BlindSigner sends s' to Warden.

7. Warden sends s to Verifier.

8. Verifier confirms that (t, s) is a valid signature on m w.r.t. PK.

Typically, Warden is instantiated as a tuple of mappings (f_t, f_c, f_s) and that in various moves do the following:

1. Warden randomly generates Δ_c and Δ_s, computes $t := f_t(PK, t', \Delta_c, \Delta_s)$, and sends t to Verifier.

2. Verifier sends m to Warden.

3. Warden computes $c := H(t, m)$ $c' := f_c(PK, t', \Delta_c, \Delta_s, c)$ and sends c' to BlindSigner.

4. BlindSigner computes s' and sends it to Warden.

5. Warden computes $s = f_s(PK, t', \Delta_c, \Delta_s, t, c, c', s')$ and sends it to Verifier.

If (t', c', s') is a valid PoK, then so is (t, c, s). We call these type of interactive proof **Transfer Proof of Knowledge (TPoK)**.

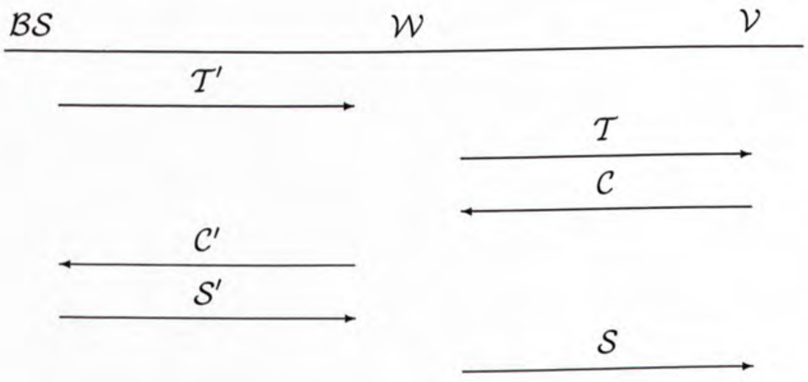

Figure 3.1: Transferred Proof of Knowledge (TPoK)

Some examples below.

Schnorr blind signature[33]: Relation $R = \{(y = g^x, x) | x \in \{1, \cdots, q\}\}$ with $(\mathcal{T}, \mathcal{C}, \mathcal{S} : \mathcal{T} = g^{\mathcal{S}} y^{\mathcal{C}})$ and $\mathcal{T} = \mathcal{T}' g^{\Delta_s} y^{\Delta_c}$,

Okamoto-Schnorr blind signature [34]: Relation $R = \{(g^{x_1} h^{x_2}, (x_1, x_2)) | x_1, x_2 \in \{1, \cdots, q\}\}$. with $(\mathcal{T}, \mathcal{C}, \mathcal{S}) = (\ g^{r_1} h^{r_2},\ c,\ (s_1, s_2) = (x_1 + r_1 c, x_2 + r_2 c)\ )$

Definition 3.4. Blindness: *The signer of a blind signature has no information about the message during and after a blind signature/TPoK protocol. Given any message-signature pair, the signer cannot find out when and for whom it was signed.* □

Chapter 4

Spontaneous Anonymous Group (SAG) Signature

4.1 Introduction

In this chapter we will introduce the concept of spontaneous anonymous group (SAG) signature. We define SAG signatures the set of signature schemes which provides privacy to the signature signer, while the receiver can validate the signature and assure that the signer belongs to a certain set of people. This kind of signature can be instantiated by the use of ring signatures and signature blinding techniques. In this chapter we will go through these techniques which our works based on. First we will visit the precursor of ring signature: group signature.

4.2 Background

4.2.1 Group Signature

Group signatures, formalized by Chaum[35], is a generalization of a membership authentication schemes which can convince the verifier that he belongs to a certain group without revealing his identity. In general, a group signature scheme will possess the following properties:

1. Only registered members of the group can sign messages
2. Receiver can verify that the signature is a valid group signature, however one cannot discover which group member generate it.
3. If there's any dispute, an group authority (called the group manager) can "*open*" the signature to revoke the identity of the signer.

Group signature offers many attractive applications involving anonymity and privacy. For example, in a company there a server which stores confidential business secrets and only the board of executives are permitted to access those data. Any access to the server required authentication of the user. At the same time the company also wants to protect privacy, which the user's credential is kept secret during normal use. In case the system is being abused, the internal auditor can revoke the identity of the misbehave users for further actions.

4.2.2 Threshold Signature

In traditional cryptography the goal is for a single sender to encrypt or sign a message which is intent for a single receiver. However there's application which requires the share of power, for example, a important company decision may require a majority number of shareholders to sign with.

Secret sharing[36, 37] is one of the most earliest form to share the power. The article "*How to share a secret*", written by Shamir [36], proposed a secret sharing scheme. A secret is divided into among n entities, in which any t of them can reconstruct the whole secret by using Lagrange polynomial interpolation by a central trusted party.

Since the scheme proposed by Shamir is not suited for signature applications as the share holders need to reveal their secret shares to a trusted party, this trusted party possess more power then other users which violates the principle of power sharing. Threshold signature is different from secret sharing schemes in which the power to reconstruct the secret key is shared. Desmedt[38] proposed a threshold signature scheme in which the signature is generated by recombining t partial signatures and eliminate the use of trusted party.

The security goals of threshold cryptography are consists of but not limited to the followings:

- **Unforgeability** The unforgeability of a threshold signature is a generalization of unforgeability of ordinary digital signature. For a t-threshold signature scheme, a valid signature must be made by a set of t or more signer who possess the corresponding secret shares.

- **Robustness** Robustness is the ability for the scheme to withstand incorrect partial shares. The scheme can detect and prevent the malformed shares to corrupt the signing process. The malformed shares can either be corrected or discard.

- **Proactive Security** The scheme should able to reduce the damage inflicted by stolen partial shares. Many literature in proactive secret sharing scheme[39, 40, 41] and forward security [42, 43, 44]targets this problem.

- **No Trusted Dealer** No one should have the power to obtain any shared secret from others during the signing process.

Many threshold cryptographic schemes have been proposed. Gennaro et. al. [45] proposed a robust threshold signature scheme based on digital signature scheme (DSS) which is an industry standard. Boldyreva[46] proposed a threshold signature scheme based on Gap Diffie Hellman Group. More threshold schemes can be found in [47, 48, 49].

4.3 SAG signatures

Recently a fundamental alternative has gained wide interests. In the *spontaneity* paradigm to group cryptography, there is no group secret. There is also no setup. Any single entity can arbitrarily and spontaneously conscript $n-1$ diversion members to form a group, and complete a signature without the participation, or even knowledge, of the diversion members. The resulting signature can be proven to be from one of the n group members. Yet the actual signer remains anonymous (signer-indistinguishable), with irrevocable, exculpable anonymity. The only requirement is that each group member has a published public key, for the purpose of signature verification. There are also t-out-of-n threshold versions where t entities joint to spontaneously conscript $n-t$ diversion members.

Compared with traditional threshold signature schemes, spontaneous group signatures achieved the definition goal quoted at the beginning of this section. Yet there is no group secret. There is no group setup which requires the participation of non-insider members.

Due to its flexibility and the ease (or lack) of setup, SAG cryptography has been deemed perfectly suitable for applications in ad hoc groups [2, 50, 16].

> Any t members of a group of n members can jointly demonstrate a knowledge concerning the group that no combination of $t-1$ or fewer members can demonstrate.

There are threshold signature schemes that require no less than t members to jointly generate. There are threshold decryption schemes (cryptosystems) that require no less than t members to jointly decrypt.

Since its inception, group cryptography and threshold cryptography have traditionally been achieved through the secret sharing technique. Also since its inception [51], anonymous (insider-indistinguishable) group cryptography has traditionally been achieved by the technique of blind signatures or other forms of transfer proof-of-knowledge (TPoK). For further details, see [51, 52]

Compared with traditional privacy (anonymity) protection schemes, spontaneous group cryptography is naturally anonymous. It achieves anonymity without using blinding techniques. Furthermore, the anonymity in spontaneous anonymous group (SAG) cryptography is very strong: in its basic version, the anonymity is unconditional (information-theoretic), irrevocable, and exculpable. The last property means that even if all communication sessions and all secret keys are subpoenaed, the anonymity cannot be revealed. Variants of SAG cryptography achieved different tradeoffs in anonymity based on candidate hard problems and optional revocability and optional culpability.

We summarize the properties of SAG signature as follow:

- Unforgeability The unforgeability of a threshold signature is a generalization of unforgeability of ordinary digital signature. For a (t, n) threshold spontaneous anonymous group signature scheme, a valid signature must be made by a set of t or more signer who possess the corresponding secret shares.

- Unlinkability The unlinkability properties means that the identity of the signers cannot be reveal even if all the communication sessions and secret keys are subpoenaed. The anonymity is unconditional secure.

- Spontaneity Unlink ordinary group signature, there is no setup stage for an entity joining the group. The formation of group is ad-hoc.

- Robustness Robustness is the ability for the scheme to withstand incorrect partial shares. The scheme can detect and prevent the malformed shares to corrupt the signing process. The malformed shares can either be corrected or discard.

- Proactive Security The scheme should able to reduce the damage inflicted by stolen partial shares. Many literature in proactive secret sharing scheme[39, 40, 41] and forward security [42, 43, 44]targets this problem.

- No Trusted Dealer No one should have the power to obtain any shared secret from others during the signing process.

4.4 Formal Definitions and Constructions

In this section we will give a formal definition for SAG signatures and illustrate some of the SAG signature constructions using ring-type and CDS structures. First an formal SAG signature is as follow:

Definition 4.1. *Let $L = \{PK_1, \cdots, PK_n\}$ be a list of n public keys, θ be an integer, $1 \le \theta \le n$, m be a message, and $\sigma = (t_1, \cdots, t_n, c_1, \cdots, c_n, s_1, \cdots, s_n)$ be a tuple. Let H, H_1, $\cdots$, H_n be full-domain collision-free secure hashing functions. The tuple $(L, n, \theta, m, \sigma)$ is a* ring-type SAG signature [2, 3] *if the following all hold:*

1. $\theta = 1$
2. *For each i, $1 \le i \le n$, we have $c_i = H_i(L, n, m, t_{i-1})$ and (t_i, c_i, s_i) is a valid PoK conversation w.r.t. PK_i. (t_0 is interpreted as t_n.)*

The tuple is a CDS1-type SAG signature [1] *if the following all hold*

1. *Each tuple (t_i, c_i, s_i) is a valid PoK conversation w.r.t. PK_i, $1 \le i \le n$.*
2. *The polynomial f interpolated from $f(i) = c_i$, $0 \le i \le n$, has degree at most $n - \theta$, where $c_0 = H(L, n, \theta, m, t_1, \cdots, t_n)$.*

The tuple is a CDS2-type SAG signature *if the following all hold*

1. *Each tuple (t_i, c_i, s_i) is a valid PoK conversation w.r.t. PK_i, $1 \le i \le n$.*
2. *For each $1 \le j \le \theta$, $\sum_{1 \le i \le n} i^j c_i = H_j(L, n, \theta, m, t_1, \cdots, t_n)$.*

□

To conserve bandwidth and storage space requirements, the representation of an SAG signature can be shortened. For example, if $(t_1, \cdots, t_n)$ can be efficiently constructed from $(c_1, \cdots, c_n, s_0, \cdots, s_n)$, then it can be omitted from the representation. In ring-type SAG signatures, $(c_2, \cdots, c_n)$ can be further omitted since they can be constructed from $(c_1, s_1, \cdots, s_n)$. In the context of this thesis the representation of a SAG signature will not be shorten for the ease of easy understanding.

4.4.1 Ring-type construction

Ring type construction is proposed by Rivest et. al.[2]. It is called "ring" type due to the cyclic structure of the protocol construction. Here we present the construction of ring-type SAG signatures:

Given a list of public keys $L = \{PK_1, \cdots, PK_n\}$, a message m, a suitable hash function H, a secret key SK_π corresponding to PK_π, a ring-type SAG signature can be constructed as follows:

1. Randomly generate a commitment $\mathcal{T}_\pi$.
2. For each $i = \pi + 1, \cdots, n, 1, \cdots, \pi - 1$, compute $\mathcal{C}_i = H(L, m, \mathcal{T}_{i-1})$ and then simualate a PoK conversation $(\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$ w.r.t. PK_i.
3. For $i = \pi$, compute $\mathcal{C}_i = H(L, m, \mathcal{T}_{i-1})$ and then compute a PoK conversation $(\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$ using the secret key SK_i.
4. Output SAG signature $(L, n, \theta = 1, m, \sigma)$ where $\sigma = (\mathcal{C}_1, \mathcal{S}_1, \cdots, \mathcal{S}_n)$ (thus achieving bandwidth conservation).

4.4.2 CDS-type construction

CDS type construction is proposed by Cramer et. al. [1]. This construction is based on the difficulties to fit n independent points to a polynomial with degree less then $n-1$. Here we present the construction of CDS-type SAG signatures as follows:

Given list of public keys $L = \{PK_1, \cdots, PK_n\}$, message m, suitable hash function H. Let $I \subset \{1, \cdots, n\}$, $|I| = t$. Given secret keys $\{SK_\pi : \pi \in I\}$, generate SAG signature as follows:

1. For each $i \notin I$, simulate a PoK conversation $(\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$.

2. For each $\pi \in I$, randomly pick $\mathcal{T}_\pi$.

3. Compute $\mathcal{C}_0 = H(L, n, \theta, m, \mathcal{T}_1, \cdots, \mathcal{T}_n)$, and solve for $\mathcal{C}_\pi$'s, $\pi \in I$, such that the polynomial f interpolated from $f(i) = \mathcal{C}_i$, $0 \leq i \leq n$, has degree no more than $n - \theta$. (resp. for CDS2-type, solve for $\mathcal{C}_\pi$'s, $\pi \in I$, such that $\sum_{1 \leq i \leq n} i^j \mathcal{C}_i = H_j(L, n, \theta, m, t_1, \cdots, t_n)$, $1 \leq j \leq \theta$.)

4. For each $\pi \in I$, compute a PoK conversation $(\mathcal{T}_\pi, \mathcal{C}_\pi, \mathcal{S}_\pi)$ using SK_π.

5. Output an SAG signature $(L, n, \theta, m, \sigma)$ where $\sigma = (f, \mathcal{S}_1, \cdots, \mathcal{S}_n)$ (achieving bandwidth conservation).

4.5 Discussion

The SAG signature has statistical zero-knowledge property (SZK) about its actual signers. Therefore, the signer anonymity is unconditional and exculpable, which means that even with infinite computation power it is not possible extract the identity of the signer from the signature unless the signer reveal himself. Furthermore, the SAG signature is a group signature which requires essentially no setup, especially in terms of group key setup or secret sharing of the group key. Any one user can conscript the public keys of another $n-1$ users to form an SAG signature without the participation or even knowledge of the conscripted diversion signers. Such properties make SAG signatures useful in diverse applications including whistle blowing[2], e-voting [53], and ad hoc group cryptography [50].

Chapter 5

Blind Spontaneous Anonymous Signature

5.1 Introduction

Here we proposed a blind spontaneous anonymous signature scheme is a combination of SAG signature and blind signatures. By allowing a user to obtain a blind signature from one of the group members, this user can construct a SAG signature on behave of the group. Our construction combines the message unlinkability properties from the blind signature and unconditional signer anonymity from the spontaneous anonymous signature, and is the first digital signature scheme provides the following special properties:

1. As a blind signature, the singing group member will not know about the content of the message.
2. As a SAG signature, the verifier is convinced that the signature is generated by one of the group member, however he cannot tell whom did it.
3. All parties except the user, are unable to figure out who signer the Blind SAG signature. Message-signature relationships and signature-signer relationships are unlinkable during and after the protocol.

4. As a result, the verifier is convinced that the user is authenticated by one of the group member.

Credential systems, including electronic cash, digital passport, online voting systems, etc. which have privacy concern can be build by using our blind sag signature scheme.

5.2 Definition

Definition 5.1 (A Blind SAG Signature). *A blind SAG signature is a $(n+3)$-tuple $(\mathcal{G}, \mathcal{I}_1, \cdots, \mathcal{I}_n, \mathcal{U}, \mathcal{V})$, comprising $(n+1)$ interactive turing machines $\mathcal{I}_1, \cdots, \mathcal{I}_n$ and $\mathcal{U}$, and two algorithms $\mathcal{G}$ and $\mathcal{V}$.*

- *$\mathcal{G}$ is a probabilistic polynomial-time key-generation algorithm which takes an input security parameter 1^k and outputs a pair (PK, SK) of public and secret key.*
- *$\mathcal{I}_i$, for $i = \{1, \cdots n\}$ and $\mathcal{U}$ are polynomially-bounded probabilistic Interactive Turing Machines, where all machines are given the (separate) tapes. Let (PK_i, SK_i) be a key pair of $\mathcal{I}_i$ generated by inputting 1^{k_i} to $\mathcal{G}$. Let $L = \{PK_1, \cdots, PK_n\}$ denote a list of public keys and $J \subset \{1, \cdots, n\}$, where $|J| = t$. For each $\mathcal{I}_i$ is given on its input tape a secret key SK_i and corresponding public key, PK_i. $\mathcal{U}$ is given on its input tape a message $m \in \{0,1\}^*$ and L. The length of inputs are polynomial in security parameter, k where $k =_{min} \{k_i : i = 1, \cdots, n\}$. For $i \in J$, $\mathcal{I}_i$ and $\mathcal{U}$ engage in the interactive protocols of some polynomial in security parameter number of rounds. At the end of the protocols, $\mathcal{I}_i$ outputs* completed *or* not-completed *and $\mathcal{U}$ outputs either (L, σ, m) or* fail.
- *$\mathcal{V}$ is a probabilistic polynomial-time algorithm that takes (L, σ, m) and outputs either* accept *or* reject.

□

5.2.1 Security Model

In order to prove our scheme is secure in the random oracle model, we will need the following oracles during the simulation:

1. $\mathcal{SO}$ (Signing Oracle): Upon input a public key PK' and any message m', it outputs a valid signature σ'.
2. SAGSign (SAG Signing Oracle): Upon input public key list L', length n', threshold θ', message m', it outputs a valid SAG signature $(L', n', \theta', m, \sigma')$.
3. BlindSign (Blind Signing Oracle): Upon query, it conducts a 4-move interactive protocol with the querier $\mathcal{Q}$ as follows:
   (a) Move-0: $\mathcal{Q}$ sends PK'.
   (b) Move-1: BlindSign sends a *commitment* t to $\mathcal{Q}$.
   (c) Move-2: $\mathcal{Q}$ sends a challenge c to BlindSign.
   (d) Move-3: BlindSign returns s such that (t, s) forms a valid PoK w.r.t. PK'.
4. Random Oracle $\mathcal{H}$: Upon receiving a query, it outputs a random number. All query-reply pairs are kept in record and no same reply for different queries.

5.2.2 Definitions of security notions

Definition 5.2. *(Completeness) If all parties are honest in following the protocols, then the output of the interactions with various oracles will produce valid signatures.* □

Game UF

1. (Setup) Upon input a security parameter 1^λ, generate parameters n, θ, and invoke KeyGen n times to generate key pairs (SK_i, PK_i), $1 \leq i \leq n$. The above, except the secret keys, are published.

2. A forger, $\mathcal{F}$ makes q_B (resp. q_S, q_H, q_A) queries to the BlindSigner (resp. $\mathcal{SO}$, random oracle, SAGSign).

3. $\mathcal{F}$ delivers $> q_B/\theta$ valid SAG signatures $(L_i, n_i, \theta, m_i, \sigma_i)$, $1 \leq i \leq q_B+1$, none of which coincides with any SAGSign query output.

Remark: For simplicity, we require $\mathcal{F}$ to deliver SAG signatures with the same threshold θ, and each public key used in SAG signatures delivered by $\mathcal{F}$ must have been generated in the Setup Phase of Game UF. In this paper, we restrict ourselves to at most a polynomially many queries in terms of the security parameter.

Definition 5.3. *(Parallel One-more Unforgeability (p1m-uf)) A blind SAG signature scheme is parallel one-more unforgeable (against adaptive chosen-message, chosen-public-key active attackers) if no PPT adversary can successfully complete Game UF with non-negligible probability.* □

Remark: Specializing to $n = \theta = 1$, the above definition is defining p1m-uf of classic blind signatures.

Definition 5.4. ***Blindness:*** *A blind SAG signature scheme has* blindness *if the probability distribution of the signature produced by Warden is indistinguishable from the probability distribution of the signatures produced by Warden conditioned on the blindsign conversation that produced it.* □

Roughly speaking,

$$\Pr\left\{\begin{array}{l|l}\text{SAG signature} & \text{BlindSign Oracle}\\ \text{by Warden} & \text{conversation}\end{array}\right\} = \Pr\left\{\begin{array}{l}\text{SAG signature}\\ \text{by Warden}\end{array}\right\}$$

5.3 Constructing blind SAG signatures

We present the constructions of our blind SAG signatures.

5.3.1 Blind SAG signature: CDS-type [1]

Given a list of n public keys, $L = \{PK_1, \cdots, PK_n\}$, message m, threshold θ, and θ accesses to blind signer w.r.t. public keys from L, the following protocol generates a CDS1-type SAG signature (resp. CDS1-type, CDS2-type) $(L, n, \theta, m, \sigma)$:

1. Select $I \subset \{1, \cdots, n\}$, $|I| = \theta$.
2. For each $i \in \{1, \cdots, n\} \setminus I$, generate PoK triple (t_i, c_i, s_i) w.r.t. PK_i.
3. In θ sessions of the TPoK protocol, one for each $i \in I$, act as Warden equipped with $\mathsf{BlindSigner}_{PK_i}$ w.r.t. PK_i, as follows:
   (a) Obtain commitment t'_i from $\mathsf{BlindSigner}_{PK_i}$, for each $i \in I$.
   (b) For each $i \in I$, compute $\Delta_{s,i}$, $\Delta_{c,i}$, and $t_i = f_t(PK_i, t'_i, \Delta_{c,i}, \Delta_{s,i})$.
   (c) Compute $c_0 = H(L, n, \theta, m, t_1, \cdots, t_n)$.
   (d) Compute c_i for all $i \in I$ such that the polynomial f interpolated from $f(i) = c_i$, $0 \leq i \leq n$, has degree at most $n - \theta$. (resp. for CDS2-type, solve for $\mathcal{C}_i$'s, $i \in I$, such that $\sum_{1 \leq i \leq n} i^j \mathcal{C}_i = H_j(L, n, \theta, m, t_1, \cdots, t_n)$, $1 \leq j \leq \theta$.)
   (e) For each $i \in I$, compute $c'_i = f_c(PK_i, t'_i, \Delta_{c,i}, \Delta_{s,i}, c_i)$, and send c'_i to $\mathsf{BlindSigner}_{PK_i}$.
   (f) For each $i \in I$, receive s'_i from $\mathsf{BlindSigner}$ i, and compute $s_i = f_s(PK_i, t'_i, \Delta_{c,i}, \Delta_{s,i}, c_i, s'_i)$.
4. Output $\sigma = (f, s_1, \cdots, s_n)$.

The blind signature for individual index i is referred to as the *underlying blind signature scheme* of the blind SAG signature scheme.

5.3.2 Blind SAG signature: ring-type [2, 3]

Given a list of n public keys $L = \{PK_1, \cdots, PK_n\}$, message m and accesses once to $\mathsf{BlindSigner}_{PK_i}$ w.r.t. $PK_i \in L$, the following protocol generates a ring-type SAG signature (L, n, m, σ):

1. Select $\pi \in \{1, \cdots, n\}$.
2. Interact as Warden with $\mathsf{BlindSigner}_{PK_\pi}$ to obtain a commitment t'_π, and compute $t_\pi = f_t(PK_\pi, t'_\pi, \Delta_{c,\pi}, \Delta_{s,\pi})$ with randomly generated $\Delta_{c,\pi}$ and $\Delta_{s,\pi}$).
3. Sequentially for each $i = \pi{+}1, \cdots, n, 1, \pi{-}1$, compute $c_i = H(L, m, n, t_{i-1})$, and then simulate a PoK triple (t_i, c_i, s_i) w.r.t. PK_i.
4. Finish the interaction with $\mathsf{BlindSigner}_{PK_\pi}$ by
   (a) Compute and send $c'_\pi = f_c(PK_\pi, t'_\pi, \Delta_{c,\pi}, \Delta_{s,\pi}, c_\pi)$.
   (b) Receive s'_π nd compute $s_\pi = f_s(PK_\pi, t'_\pi, \Delta_{c,\pi}, \Delta_{s,\pi}, c_\pi, s'_\pi)$.
5. Output $\sigma = (c_1, \cdots, c_n, s_1, \cdots, s_n)$.

5.4 Security Analysis

We prove the completeness, the blindness, and the parallel one-more unforgeability of our blind SAG signature schemes. In the process, we also prove an extension of Schnorr's [7] ROS result from single public key to multiple public keys.

5.4.1 Multi-key parallel one-more unforgeability of blind signature

The following results are well-known.

Theorem 5.5. *[7] The parallel one-more unforgeability (p1m-uf) of Schnorr (resp. Okamoto-Schnorr) blind signature is equivalent to the ROS Problem in the random oracle model plus the generic group model.*

Proof. In Schnorr's security model [7], all queries to blindsign are w.r.t. a single public key PK. We generalize it to *multiple-key parallel one-more unforgeability* (mk-p1m-uf) by allowing the Adversary to query blindsign with K different (PK_i) , $1 \le i \le n$, a total of q_B times in order to produce a total of $q_B + 1$ signatures each of which is verifiable against some members of the set of public keys $\{PK_1, \cdots, PK_K\}$. We will need this result.

Theorem 5.6. *The multiple-key parallel one-more unforgeability (mk-p1m-uf) of Schnorr (resp. Okamoto-Schnorr) blind signature is equivalent to the ROS Problem in the random oracle model plus the generic group model.*

Proof of Theorem We mimick Schnorr's [7] proof. The generic mk-p1m attacker is as follows:

1. Obtain commitments: $t_{k,i}$, $1 \le k \le K$, $1 \le i \le q_{B,k}$; where $\sum_k q_{B,k} = q_B$.
2. Compute and then send challenges $c_{k,i}$, $1 \le k \le K$, $1 \le i \le q_{B,k}$.
3. Receive responses $s_{k,i}$. Output $q_B + 1$ signatures $(\hat{t}_{\ell,j}, \hat{s}_{\ell,j})$ on messages $\hat{m}_{\ell,j}$ where $\hat{t}_{\ell,j} = g^{\hat{s}_{\ell,j}} y_\ell^{\hat{c}_{\ell,j}}$, $\hat{c}_{\ell,j} = H(\hat{t}_{\ell,j}, \hat{m}_{\ell,j})$; and $1 \le \ell \le K$, $1 \le j \le \hat{q}_{B,\ell}$, $\sum_\ell \hat{q}_{B,\ell} = q_B + 1$

The oracle conversations can be arbitrarily interleaved. The hash query $\hat{c}_{\ell,j} = H(\hat{t}_{\ell,j}, \hat{m}_{\ell,j})$ must have been made.

Let $f_{\tau(\ell,j)} = \hat{c}_{\ell,j}$, for some index mapping τ.

In Eq(2.1), we can treat $u_i = y_{q_B+i}$. The u_i's can be used as public keys in querying the Signing Oracle. If they are not used as such, then set $q_{B,q_B+i} = 0$. They cannot be used as public keys in the delivered signatures, if the conditions so require. Therefore, we can omit the u's w.l.o.g. Expanding the subscript of the t's from one to two according to the current convention, we obtain

$$\begin{aligned} f_{\tau(\ell,j)} &= g^{\hat{s}_{\ell,j}} y_\ell^{\hat{c}_{\ell,j}} \\ &= g^{a_{\tau(\ell,j),-1}} \prod_{k'} y_{k'}^{a_{\tau(\ell,j),k'}} \prod_k \prod_i t_{k,i}^{b_{\tau(\ell,j),k,i}} \\ &= g^{a_{\tau(\ell,j),-1}} \prod_{k'} y_{k'}^{a_{\tau(\ell,j),k'}} \prod_k \prod_i (g^{s_{k,i}} y_k^{c_{k,i}})^{b_{\tau(\ell,j),k,i}} \end{aligned}$$

and

$$1 = g^{\Delta_{s,\ell,j}} \prod_{k'} y_{k'}^{\Delta_{c,\ell,j,k'}}, \text{ for each } \ell, j,$$

where

$$\begin{aligned} \Delta_{s,\ell,j} &= -\hat{s}_{\ell,j} + a_{\tau(\ell,j),-1} + \sum_k \sum_i s_{k,i} b_{\tau(\ell,j),k,i} \\ \Delta_{c,\ell,j,k'} &= -\hat{c}_{\ell,j}\delta(\ell,k') + a_{\tau(\ell,j),k'} + \sum_i c_{k',i} b_{\tau(\ell,j),k',i}. \end{aligned}$$

where the Kronecker delta $\delta(u,v) = 1$ when $u = v$ and equals 0 otherwise. Note that the last two Δ-coefficients are computable by the generic adversary, but not by the Simulator. Therefore rewinding will not enable the Simulator to extract any secret key.

Case (1): $\Delta_{s,\ell,j} = \Delta_{c,\ell,j,k'} = 0$ for all ℓ, j, k'. Then the generic adversary has solved the ROS Problem:

$$\hat{c}_{\ell,j} = a_{\tau(\ell,j),\ell} + \sum_i c_{\ell,i} b_{\tau(\ell,j),\ell,i}, \text{ all } \ell, j.$$

where $\hat{c}_{\ell,j}$'s are $q_B + 1$ hash outputs.

Case (2): the opposite. Then the generic adversary has computed a nontrivial linear dependence among discrete logrithms of $y_{k'}$, i.e. the generic adversary has solved the one-more discrete logarithm problem.

Remark: In the generic group model (GGM), the above linear dependence is a form of *discrete logarithm collision.* It can be deducted in GGM that the probability of a PPT algorithm being able to compute a discrete logarithm collision, including the kind above, is negligible. □

5.4.2 Security of our blind SAG signatures

The security proof of our blind spontaneous group signature is based on the multiple-key parallel one-more unforgeability of Schnorr signature, random oracle model(ROM) and the Generic Group Model(GGM).

Theorem 5.7. (Completeness*) Our blind SAG signature has completeness.*

Proof. Reader can show the completeness by trivial calulations. □

Theorem 5.8. (Blindness) *Assume L, n, θ are fixed. Our ring-type (resp. CDS1-type, CDS2-type) blind SAG signature has blindness provided the underlying blind signature also has it.*

Proof. Proof Sketch: Denote the SAGBlindSign session communication transcripts by $\mathcal{K}_i = (\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$, $1 \le i \le \theta$, and the SAG signature in question by $(L, n, \theta, m, \sigma)$ where $\sigma{=}(t_1, \cdots, t_n, c_1, \cdots, c_n,\ s_1, \cdots, s_n)$. By the message indistinguishability of the underlying blind signatures, (t_i, c_i, s_i) is zero-knowledge w.r.t. $(\mathcal{T}_i, \mathcal{C}_i, \mathcal{S}_i)$. Furthermore, the non-blind SAG signatures protocol are witness indistinguishability about which secret key actually generated it. Therefore, the resulting blind SAG signature has message indistinguishability and signer anonymity. □

Theorem 5.9. *(*Unforgeability*) Our ring-type (resp. CDS1-type with $\theta = 1$, CDS2-type with $\theta \ge 1$) SAG blind signature based on Schnorr or Okamoto-Schnorr blind signature is parallel one-more unforgeable (p1m-uf) provided Schnorr's ROS Problem is hard, in the generic group model (GGM) plus the random oracle model (ROM).*

Proof. We prove for **CDS1-type**, $\theta = 1$, first. The generic attacker in GGM of p1m-uf of blind Schnorr SAG signature is as follows:

1. Input: a list of public keys $L = \{y_1, \cdots, y_n\}$.
2. Receive commitments $t_{k,i}$, $1 \le i \le q_{B,k}$ from $\mathsf{BlindSigner}_{PK_i}$. Note $\sum_k q_{B,k} = q_B$.
3. Send challenges $c_{k,i}$, receive responses $s_{k,i}$.
4. Output SAG signatures $\sigma_j = (\hat{t}_{j,1}, \cdots, \hat{t}_{j,n}, \hat{c}_{j,1}, \cdots, \hat{c}_{j,n}, \hat{s}_{j,1}, \cdots, \hat{s}_{j,n})$ on message $\hat{m}_j$, $1 \le j \le q_B + 1$.

The queries $\hat{c}_{j,0} = H(L, n, \theta, \hat{m}_j, \hat{t}_{j,1}, \cdots, \hat{t}_{j,n})$, $1 \le j \le q_B + 1$, must have been made. Let the Lagrange interpolation be indicated $\sum_{0 \le \ell' \le n} \gamma_{\ell'} \hat{c}_{j,\ell'} = 0$. By GGM, there exists a index mapping τ such that, for $1 \le j \le q_B + 1$ and $1 \le \ell \le n$,

$$\begin{aligned}
\hat{t}_{j,\ell} &= g^{\hat{s}_{j,\ell}} y_\ell^{\hat{c}_{j,\ell}} \\
&= g^{a_{\tau(j,\ell),-1}} \prod_{\ell'} y_{\ell'}^{a_{\tau(j,\ell),\ell'}} \prod_{k,i} t_{k,i}^{b_{\tau(j,\ell),i}} \\
&= g^{a_{\tau(j,\ell),-1}} \prod_{\ell'} y_{\ell'}^{a_{\tau(j,\ell),\ell'}} \prod_{k,i} (g^{s_{k,i}} y_k^{c_{k,i}})^{b_{\tau(j,\ell),i}} \\
1 &= g^{\Delta_{s,j,\ell}} \prod_{\ell'} y_{\ell'}^{\Delta_{c,j,\ell,\ell'}} \text{ where} \\
\Delta_{s,j,\ell} &= -\hat{s}_{j,\ell} + a_{\tau(j,\ell),-1} + \sum_{k,i} s_{k,i} b_{\tau(j,\ell),k,i}, \text{ all } j, \ell \\
\Delta_{c,j,\ell,\ell'} &= -\hat{c}_{j,\ell}\delta(\ell,\ell') + \sum_{\ell'} a_{\tau(j,\ell),\ell'} + \sum_{k,i} c_{k,i} b_{\tau(j,\ell),k,i}, \text{ all } j, \ell, \ell'
\end{aligned}$$

The negligibility of discrete logarithm collision leads to

$$\begin{aligned}
0 &= -\hat{c}_{j,\ell'} + \sum_{\ell'} a_{\tau(j,\ell'),\ell'} + \sum_{k,i} c_{k,i} b_{\tau(j,\ell'),k,i}, \text{ all } j, \ell \\
-\gamma_0 \hat{c}_{j,0} &= \sum_{\ell'=1}^{n} \gamma_{\ell'} \hat{c}_{j,\ell'} \\
&= \sum_{1 \le \ell' \le n} \gamma_{\ell'} (\sum_{\ell'} a_{\tau(j,\ell'),\ell'} + \sum_{k,i} c_{k,i} b_{\tau(j,\ell'),k,i})
\end{aligned}$$

for $1 \leq j \leq q_B+1$. The generic adversary has solved the above ROS Problem, where $\hat{c}_{j,0}$ are q_B+1 hash outputs.

CDS2-type, $\theta \geq 1$. Similar to the above, with the following modificaitons: The hash queries are $\hat{c}_{j,0}^{(\theta')} = H_{\theta'}(L, n, \theta, \hat{m}_j, \hat{t}_{j,1}, \cdots, \hat{t}_{j,n})$, $1 \leq j \leq q_B+1$, $1 \leq \theta' \leq \theta$, have been made. The ROS Problem is

$$\begin{aligned} -\gamma_0^{(\theta')}\hat{c}_{j,0}^{(\theta')} &= \sum_{\ell'=1}^{n} \gamma_{\ell'}^{(\theta')}\hat{c}_{j,\ell'} \\ &= \sum_{1\leq \ell' \leq n} \gamma_{\ell'}^{(\theta')}\Big(\sum_{\ell'} a_{\tau(j,\ell'),\ell'} + \sum_{k,i} c_{k,i} b_{\tau(j,\ell'),k,i}\Big) \end{aligned}$$

where $\hat{c}_{j,0}^{(\theta')}$'s are q_B+1 hash outputs expressed in terms of q_B commitments $c_{k,i}$'s.

ring-type: The proof is similar and omitted. □

Remark: The reduction in Theorem 5.5 is actually to the ROS Problem or the Discrete Logarithm Problem (DLP). The reduction in Theorem 5.6 (resp. Theorem 5.9) is actually to the ROS Problem or the *one-more discrete log (1mDL)* problem. (The 1mDL Problem: compute all discrete logarithms $\log_g y_i$ for $1 \leq y \leq q_{DL}+1$, given g and $y_1, \cdots, y_{q_{DL}+1}$ and a total of q_{DL} queries to a Corruption Oracle, which returns the discrete logarithms of qualified query values.) In the GGM, it can be deducted that the probability of computing dicrete log collisions, which include DLP and 1mDL, is negligible for PPT algorithms.

5.5 Discussion

We have constructed blind SAG signatures, both ring-type and CDS-type. We have reduced their parallel one-more unforgeability against adaptive chosen-plaintext, adaptive chosen-public-key attackers, to the parallel one-more unforgeability of the component blind signature, and a candidate hard problem, in two cases: where the underlying blind signature is the Schnorr (resp.

Okamoto-Schnorr) blind signature.

The security and privacy (anonymity) of the blind SAG signature based on Schnorr blind signature is an interesting topic. The result of Schnorr[7] reduced the security of the Schnorr blind signature to the ROS (Randomized Oversampled Solvable system) Assumption. Recently, Wagner [54] gave a sub-exponential time algorithm to solve the ROS problem. If the array entries are all elements of a binary field, then the ROS Problem can be solved in polynomial time by a method from [55] or [56].

It will be interesting to generalize Schnorr ROS reduction to the Schnorr-based blind SAG signature. Since Schnorr identification scheme does not have zero-knowledge, it will also be interesting to explore the exact zero-knowledge properties of that blind SAG signature.

Chapter 6

Linkable Spontaneous Anonymous Group Signature

6.1 introduction

The notion of linkable spontenous group signature was introduced by Liu, et al. [57]. They are based on SAG signatures, but added with linkability in which such signatures allow anyone to determine if two signatures are signed by the same group member. Such signatures are said to be *"linked"*. If the signer only signed once on behalf of a group, the signer can still retain the anonymity as in the conventional SAG signature scheme. However if the user signs more than once on behalf for the same group, then valid parties can revoke the identity of the signer from these signatures. This properties have many useful application in electronic cash, whistle blowing, electronic voting systems[57].

6.2 Related work

Linkable Threshold SAG Signatures. In [57], a (d, n)-threshold extension to its original linkable SAG signature scheme is constructed by concatenating d linkable SAG signatures. We note that the construction, though simple and trivial, is not efficient. In particular, the space and time complexities are both $O(dn)$. Here we propose a construction with time and space complexities both being $O(n)$.

Event-Oriented Linkability. In [57], one can tell if two SAG signatures are linked or not if and only if they are signed on behalf of the same group of members. We call this "*group-oriented*" linkability. We introduce a new linking criterion that we call "*event-oriented*" linkability, in which one can tell if two signatures are linked if and only if they are signed for the same event, despite the fact that they may be signed on behalf of different groups. Event-oriented linkable SAG signatures are comparatively more flexible in application. e.g. For a dynamic ad-hoc groups the set of group members keep changing frequently, the group-oriented linkable SAG signature will mostly signed on behave of different groups, which renders group-oriented linkability virtually useless.

Another advantage of using event-oriented linkability is that you may control the linkability between different events. Consider another scenario: The CEOs of a company vote for business decisions. Using linkable SAG signatures, they can vote anonymously by SAG-signing their votes. However, as the group is fixed throughout the polls, votes among polls can be linked by anybody and information can be derived which means anonymity is in jeopardy.This can be prevented when an event-oriented scheme is used.

6.3 Basic Building Blocks

In this section, we describe some three-move interactive honest verifier zero-knowledge proof of knowledge (HVZK PoK) protocols that we will use as basic building blocks for our event-oriented linkable threshold SAG signature scheme. These protocols all work in finite cyclic groups of quadratic residues modulo safe prime products. For each $i = 1, \ldots, n$, let N_i be a safe-prime product and define the group $G_i \doteq QR(N_i)$ such that its order is of length $\ell_i - 2$ for some $\ell_i \in \mathbf{N}$. Also let g_i, h_i be generators of G_i such that their relative discrete logarithms are not known.

Let $1 < \epsilon \in \mathbf{R}$ be a parameter and let $\mathcal{H} : \{0,1\}^* \to \mathbf{Z}_q$ be a strong collision-resistant hash function, where q is a κ-bit prime for some security parameter $\kappa \in \mathbf{N}$. Define $\mathcal{N} \doteq \{1, \ldots, n\}$ and $\Gamma_i \doteq \{-2^{\ell_i}q, \ldots, (2^{\ell_i}q)^{\epsilon}\}$.

Our security proof will require the following mathematical assumption:

Definition 6.1 (Strong RSA Problem). *[18, 19, 20]] Given a safe prime product N, and $z \in QR(N)$, it is infesible to find $u \in \mathbf{Z}_N^*$ and $e > 1$ such that $u^e = z (\bmod N)$ in time polynomial in length of N*

Definition 6.2 (Decisional Diffie-Hellman (DDH) over QR(N) Assumption). *Given a generator g of a cyclic group $QR(N)$, where N is a composite of two primes, the distribution ensembles (g^x, g^y, g^z) and (g^x, g^y, g^{xy}), where $x, y, z \in_R [1, ord(g)]$, are computationally indistinguishable by all PPT algorithm in time polynomial in the size of N.*

6.3.1 Proving the Knowledge of Several Discrete Logarithms

This protocol is a straightforward generalization of the protocol for proving the knowledge of a discrete logarithm over groups of unknown order in [18]. This allows a prover to prove to a verifier the knowledge of n discrete logarithms $x_1, \ldots, x_n \in \mathbf{Z}$ of elements $y_1, \ldots, y_n$ and to the bases $g_1, \ldots, g_n$ respectively. Using the notation in [26], the protocol is denoted by:

$$PK\{(\alpha_1, \ldots, \alpha_n) : \bigwedge_{i=1}^{n} y_i = g_i^{\alpha_i}\}.$$

A prover $\mathcal{P}$ knowing $x_1, \ldots, x_n \in \mathbf{Z}$ such that $y_i = g_i^{x_i}$ for all $i = 1, \ldots, n$ can prove to a verifier $\mathcal{V}$ his/her knowledge as follows.

- (Commit.) $\mathcal{P}$ chooses $r_i \in_R \mathbf{Z}_{(2^{\ell_i}q)^\epsilon}$ and computes $t_i \leftarrow g_i^{r_i}$ for all $i = 1, \ldots, n$. $\mathcal{P}$ sends $(t_1, \ldots, t_n)$ to $\mathcal{V}$.
- (Challenge.) $\mathcal{V}$ chooses $c \in_R \mathbf{Z}_q$ and sends it to $\mathcal{P}$.
- (Response.) $\mathcal{P}$ computes, for all $i = 1, \ldots, n$, $s_i \leftarrow r_i - cx_i$ (in $\mathbf{Z}$). $\mathcal{P}$ sends $(s_1, \ldots, s_n)$ to $\mathcal{V}$.

$\mathcal{P}$ verifies by checking, for all $i = 1, \ldots, n$, if $t_i \overset{?}{=} g_i^{s_i} y_i^c$.

Theorem 6.3. *If the Strong RSA assumption holds, the protocol in Section 6.3.1 is an Honest Verifier Zero-Knowledge Proof of Knowledge (HVZK-PoK)protocol.*

Proof. We omit the proof as it is a straightforward extension of the proof of Lemma 1 in [18]. □

By applying Fiat-Shamir transformation, the HVZK PoK protocol can be turned into a signature scheme by replacing the challenge by the hash of the commitment together with the message M to be signed:

$$c \leftarrow \mathcal{H}((g_1, y_1)|| \cdots ||(g_n, y_n)||t_1|| \cdots ||t_n||M)$$

In this case, the signature is $(c, s_1, \ldots, s_n)$ and the verification becomes:

$$c \overset{?}{=} \mathcal{H}((g_1, y_1)|| \ldots ||(g_n, y_n)||g_1^{s_1} y_1^c|| \ldots ||g_n^{s_n} y_n^c||M)$$

Following [26], we denote this signature scheme by:

$$SPK\{(\alpha_1, \ldots, \alpha_n) : \bigwedge_{i=1}^{n} y_i = g_i^{\alpha_i}\}(M)$$

6.3.2 Proving the Knowledge of d Out of n Equalities of Discrete Logarithms

This protocol is constructed using the techniques described in [58], by combining the PoK for discrete logarithm in [18] and the secret sharing scheme due to Shamir [59]. This allows a prover to prove to a verifier his/her knowledge of some d out of n integers $x_1, \ldots, x_n$, where $x_i = \log_{g_i} y_i = \log_{h_i} v_i$ for all $i = 1, \ldots, n$. The protocol is denoted by:

$$PK\left\{(\alpha_1, \ldots, \alpha_n) : \bigvee_{\mathcal{J} \subseteq \mathcal{N}, |\mathcal{J}|=d} \left(\bigwedge_{i \in \mathcal{J}} y_i = g_i^{\alpha_i} \wedge v_i = h_i^{\alpha_i}\right)\right\}$$

A prover $\mathcal{P}$ knowing, for all $i \in \mathcal{I}$, $x_i \in \mathbf{Z}$ such that $y_i = g_i^{x_i}$ and $v_i = h_i^{x_i}$, where $\mathcal{I}$ is some subset of $\mathcal{N}$ such that $|\mathcal{I}| = d$, can prove his/her knowledge to a verifier $\mathcal{P}$ as follows.

- (Commit.) $\mathcal{P}$ does the following: For $i \in \mathcal{N} \backslash \mathcal{I}$, select $c_i \xleftarrow{R} \mathbf{Z}_q$. For all $i \in \mathcal{N}$, select $r_i \xleftarrow{R} \mathbf{Z}_{(2^{\ell_i} q)^{\epsilon}}$. Compute

$$t_i \leftarrow \begin{cases} g_i^{r_i}, & i \in \mathcal{I}; \\ g_i^{r_i} y_i^{c_i}, & i \in \mathcal{N} \backslash \mathcal{I}, \end{cases} \quad \text{and } T_i \leftarrow \begin{cases} h_i^{r_i}, & i \in \mathcal{I}; \\ h_i^{r_i} v_i^{c_i}, & i \in \mathcal{N} \backslash \mathcal{I}. \end{cases}$$

  $\mathcal{P}$ sends $(t_1, \ldots, t_n, T_1, \ldots, T_n)$ to $\mathcal{V}$.

- (Challenge.) $\mathcal{V}$ chooses $c \in_R \mathbf{Z}_q$ and sends it to $\mathcal{P}$.

- (Response.) $\mathcal{P}$ does the following: Compute a polynomial f of degree $\leq n - d$ over $\mathbf{Z}_q$ such that $f(0) = c$ and $f(i) = c_i$ for all $i \in \mathcal{N} \backslash \mathcal{I}$. Compute $c_i \leftarrow f(i)$ for all $i \in \mathcal{I}$. Set

$$s_i \leftarrow \begin{cases} r_i - c_i x_i, & i \in \mathcal{I}; \\ r_i, & i \in \mathcal{N} \backslash \mathcal{I}. \end{cases}$$

  $\mathcal{P}$ sends $(f, s_1, \ldots, s_n)$ to $\mathcal{V}$.

$\mathcal{P}$ verifies by checking if (1) f is a polynomial of degree $\leq n-d$ over $\mathbf{Z}_q$, (2) $f(0) \stackrel{?}{=} c$, and (3) $t_i \stackrel{?}{=} y_i^{f(i)} g_i^{s_i}$ and $T_i \stackrel{?}{=} v_i^{f(i)} h_i^{s_i}$, for all $i = 1, \ldots, n$.

Theorem 6.4. *If the Strong RSA assumption holds, the protocol in Sec. 6.3.2 is an Honest Verifier Zero-Knowledge Proof of Knowledge (HVZK PoK) protocol.*

Proof. To prove the theorem, it suffices to show that the protocol is correct, sound and statistical HVZK.

- (Correctness.) Straightforward.

- (Soundness.) It suffices to show how a witness can be extracted if given two valid protocol conversations with the same commitment but different challenges. Denoting the two conversation transcripts by $\langle(t_1, \ldots, t_n, T_1, \ldots, T_n), (c), (f, s_1, \ldots, s_n)\rangle$ and $\langle(t_1, \ldots, t_n, T_1, \ldots, T_n), (c'), (f', s'_1, \ldots, s'_n)\rangle$, we have $c \neq c'$ and thus $f(0) \neq f'(0)$. As the degrees of f and f' are at most $n-d$, there are at least d distinct values $\pi_1, \ldots, \pi_d \in \{1, \ldots, n\}$ such that $f(\pi_i) \neq f'(\pi_i)$ for all $i = 1, \ldots, d$. Using arguments in [18], $f(\pi) - f'(\pi)$ divides $s'_\pi - s_\pi$ and therefore an integer $\hat{x}$ such that $y_\pi = g_\pi^{\hat{x}_\pi}$ and $v_\pi = h_\pi^{\hat{x}_\pi}$ can be computed as: $\hat{x}_\pi \leftarrow (s_\pi - s'_\pi)/(f'(\pi) - f(\pi))$.

  Hence a witness $(\hat{x}_{\pi_1}, \ldots, \hat{x}_{\pi_d})$ can be computed from two such transcripts.

- (Statistical HVZK.) To simulate a transcript, a simulator $\mathcal{S}$ first chooses uniformly at random a polynomial f' of degree $n-d$ over $\mathbf{Z}_q$. For all $i = 1, \ldots, n$, $\mathcal{S}$ picks uniformly at random $s'_i \in_R \mathbf{Z}_{(2^{\ell_i} q)^\epsilon}$ and computes $t'_i \leftarrow g_i^{s'_i} y_i^{f'(i)}$. The simulated transcript is: $\langle(t'_1, \ldots, t'_n, T'_1, \ldots, T'_n), (f'(0)), (f', s'_1, \ldots, s'_n)\rangle$.

  To prove that the simulation is statistical indistinguishable from real protocol conservations, one should consider, for each $i = 1, \ldots, n$, the probability distribution $P_{S_i}(s_i)$ of the responses of the prover and the probability distribution $P_{S'_i}(s'_i)$ according to which $\mathcal{S}$ chooses s'_i. The statistical distance between the two distributions can be computed to be at most: $2(2^{\ell_i})(q-1)/(2^{\ell_i}q)^{\epsilon} \leq 2/(2^{\ell_i}q)^{\epsilon-1}$. The result follows.

$\square$

Again by employing Fiat-Shamir transformation, this protocol can be turned into a signature scheme by replacing the challenge by the hash of the commitment together with the message M to be signed:

$$c \leftarrow \mathcal{H}((g_1,y_1,h_1,v_1)||\dots||(g_n,y_n,h_n,y_n)||t_1||\dots||t_n||T_1||\dots||T_n||M)$$

In this case, the signature is $(f, s_1, \dots, s_n)$ and step (3) of the verification becomes:

$$\begin{aligned} c \overset{?}{=} \mathcal{H}(&(g_1,y_1,h_1,v_1)||\dots||(g_n,y_n,h_n,y_n)|| \\ &y_1^{c_1}g_1^{s_1}||\dots||y_n^{c_n}g_n^{s_n}||v_1^{c_1}h_1^{s_1}||\dots||v_n^{c_n}h_n^{s_n}||M) \end{aligned}$$

We denote this signature scheme by:

$$SPK\left\{(\alpha_1,\dots,\alpha_n): \bigvee_{\mathcal{J}\subseteq\mathcal{N},|\mathcal{J}|=d}\left(\bigwedge_{i\in\mathcal{J}} y_i = g_i^{\alpha_i} \wedge v_i = h_i^{\alpha_i}\right)\right\}(M)$$

6.4 Security Model

We give our security model and define relevant security notions.

6.4.1 Syntax

A *linkable threshold SAG signature*, (LTRS) scheme, is a tuple of five algorithms (Key-Gen, Init, Sign, Verify and Link).

- $(sk_i, pk_i) \leftarrow$ Key-Gen(1^{λ_i}) is a PPT algorithm which, on input a security parameter $\lambda_i \in \mathbf{N}$, outputs a private/public key pair (sk_i, pk_i). We denote by $\mathcal{SK}$ and $\mathcal{PK}$ the domains of possible secret keys and public keys, resp. When we say that a public key corresponds to a secret key or vice versa, we mean that the secret/public key pair is an output of Key-Gen.

- param $\leftarrow$ Init(λ) is a PPT algorithm which, on input a security parameter λ, outputs the set of security parameters param which includes λ.
- $\sigma' = (e, n, d, \mathcal{Y}, \sigma) \leftarrow$ Sign$(e, n, d, \mathcal{Y}, \mathcal{X}, M)$ which, on input event-id e, group size n, threshold $d \in \{1, \ldots, n\}$, a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, a set $\mathcal{X}$ of d private keys whose corresponding public keys are all contained in $\mathcal{Y}$, and a message M, produces a signature σ'.
- 1/0 $\leftarrow$ Verify(M, σ') is an algorithm which, on input a message-signature pair (M, σ') returns 1 or 0 for accept or reject, resp. If accept, the message-signature pair is *valid.*
- 1/0 $\leftarrow$ Link (σ_1', σ_2') is an algorithm which, upon input two valid signature pairs, outputs 0 or 1 for linked or unlinked. In case of linked it additionally outputs the public key pk^* of the suspected "double-signer".

Remark: Our linkability is *accusatory* meaning it outputs the public key of the suspected "double signer". The linkability in [57] is not accusatory – it only outputs linked or unlinked without suspect identity.

Correctness.

LTRS schemes must satisfy:

- (Verification Correctness.) Signatures signed according to specification are accepted during verification.
- (Linking Correctness.) If two signatures are signed for the same event according to specification, then they are linked if and only if the two

signatures share a common signer. In the case of linked, the suspect output by Link is exactly the common signer.

6.4.2 Notions of Security

Security of LTRS schemes has three aspects: unforgeability, anonymity and linkability. Before giving their definition, we consider the following oracles which together model the ability of the adversaries in breaking the security of the schemes.

- $pk_i \leftarrow \mathcal{JO}(\perp)$. The *Joining Oracle*, on request, adds a new user to the system. It returns the public key $pk \in \mathcal{PK}$ of the new user.
- $sk_i \leftarrow \mathcal{CO}(pk_i)$. The *Corruption Oracle*, on input a public key $pk_i \in \mathcal{PK}$ that is a query output of $\mathcal{JO}$, returns the corresponding secret key $sk_i \in \mathcal{SK}$.
- $\sigma' \leftarrow \mathcal{SO}(e, n, d, \mathcal{Y}, \mathcal{V}, \mathcal{X}, M)$. The *Signing Oracle*, on input an event-id e, a group size n, a threshold $d \in \{1, \ldots, n\}$, a set $\mathcal{Y}$ of n public keys, a subset $\mathcal{V}$ of $\mathcal{Y}$ with $|\mathcal{V}| = d$, a set of secret keys $\mathcal{X}$ whose corresponding public keys are all contained in $\mathcal{V}$, and a message M, returns a valid signature σ'.

Remark: An alternative approach to specify the $\mathcal{SO}$ is to exclude the signer set $\mathcal{V}$ from the input and have $\mathcal{SO}$ select it according to suitable random distribution. We do not pursue that alternative further.

Unforgeability.

Unforgeability for LTRS schemes is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$:

1. $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.

2. $\mathcal{A}$ may query the oracles according to any adaptive strategy.

3. $\mathcal{A}$ gives $\mathcal{S}$ an event-id $e \in \mathcal{EID}$, a group size $n \in \mathbf{N}$, a threshold $d \in \{1, \ldots, n\}$, a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, a message $M \in \mathcal{M}$ and a signature $\sigma \in \Sigma$.

$\mathcal{A}$ wins the game if: (1) Verify(M,σ')=1, (2) all of the public keys in $\mathcal{Y}$ are query outputs of $\mathcal{JO}$, (3) at most $(d-1)$ of the public keys in $\mathcal{Y}$ have been input to $\mathcal{CO}$, and (4) σ is not a query output of $\mathcal{SO}$ on any input containing M. We denote by $\mathbf{Adv}_{\mathcal{A}}^{unf}(\lambda)$ the probability of $\mathcal{A}$ winning the game.

Definition 6.5 (unforgeability). *An LTRS scheme is unforgeable if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{unf}(\lambda)$ is negligible.*

Linkable Anonymity.

Anonymity for LTRS schemes is defined in the following game:

Game LA

1. (*Initialization Phase*) $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.

2. (*Probe-1 Phase*) $\mathcal{A}$ may query the oracles according to any adaptive strategy.

3. (*Gauntlet Phase*) $\mathcal{A}$ gives $\mathcal{S}$ event-id e_g, group size n_g, threshold $d_g \in \{1, \ldots, n_g\}$, message M_g, a set $\mathcal{Y}_g$ of n public keys all of which are query outputs of $\mathcal{JO}$, a subset $\mathcal{V}_g$ of $\mathcal{Y}_g$ with $|\mathcal{V}_g| = d_g$, a set of secret keys $\mathcal{X}_g$ with $|\mathcal{X}_g| = d_g - 1$ and whose corresponding secret keys are all contained in $\mathcal{V}_g$. The lone public key $y_g \in \mathcal{V}_g$ whose corresponding secret key is not contained in $\mathcal{X}_g$ has never been queried to $\mathcal{CO}$ and has been included in the insider set $\mathcal{V}$ in any query to Signing Oracle $\mathcal{SO}$.

Then $\mathcal{S}$ flips a fair coin to select $b \in \{\mathsf{real}, \mathsf{ideal}\}$. Case b=real: $\mathcal{S}$ queries $\mathcal{CO}$ with y_g to obtain its corresponding secret key x_g, and computes σ'_g = Sign (e_g, n_g, d_g, $\mathcal{Y}_g$, $\mathcal{X}_g \cup \{x_g\}$, M_g), Case b=ideal: $\mathcal{S}$ computes σ'_g = $\mathcal{SO}$ (e_g, n_g, d_g, $\mathcal{Y}_g$, $\mathcal{V}_g$, $\mathcal{X}_g$, M_g).

$\mathcal{S}$ sends σ'_g to $\mathcal{A}$.

4. (*Probe-2 Phase*) $\mathcal{A}$ queries the oracles adaptively, except that y_g cannot be queried to $\mathcal{CO}$ or included in the insider set $\mathcal{V}$ of any query to $\mathcal{SO}$.

5. (*End Game*) $\mathcal{A}$ delivers an estimate $\hat{b} \in \{\mathsf{real}, \mathsf{ideal}\}$ of b.

$\mathcal{A}$ wins the game if $\hat{b} = b$. Define the *advantage* of $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathcal{A}}^{Anon}(\lambda) = \Pr[\mathcal{A} \text{ wins}] - 1/2.$$

Definition 6.6 (Linkable-anonymity). *An LTRS scheme is linkably-anonymous if for any PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{Anon}(\lambda)$ is negligible.*

Remark: Linkable anonymity is a form of computational zero-knowledge: the attacker cannot computationally distinguish the real world from the ideal world. Note that the anonymity notions in [60, 61, 62] appear to be also computational zero-knowledge. Our attacker model is not a fully active attacker: queries relevant to the gauntlet public key, y_g, are ruled out. The anonymity in [57] is also with respect to the above model. We note that [60], p.623, argued that anonymity and linkability cannot coexist in their security model.

Linkability.

Linkability for LTRS schemes is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$:

1. $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.

2. $\mathcal{A}$ may query the oracles according to any adaptive strategy.

3. $\mathcal{A}$ gives $\mathcal{S}$ an event-id $e \in \mathcal{EID}$, group sizes $n_1, n_2 \in \mathbf{N}$, thresholds $d_1 \in \{1, \ldots, n_1\}, d_2 \in \{1, \ldots, n_2\}$, sets $\mathcal{Y}_1$ and $\mathcal{Y}_2$ of public keys in $\mathcal{PK}$ of sizes n_1 and n_2 resp., messages $M_1, M_2 \in \mathcal{M}$ and signatures $\sigma_1, \sigma_2 \in \Sigma$.

$\mathcal{A}$ wins the game if (1) all public keys in $\mathcal{Y}_1 \cup \mathcal{Y}_2$ are query outputs of $\mathcal{JO}$, (2) Verify(M_i,σ_i')=1 for $i = 1, 2$, (3) $\mathcal{CO}$ has been queried at most $(d_1 + d_2 - 1)$ times, and (4) Link(σ_1', σ_2')=0. We denote by $\mathbf{Adv}_{\mathcal{A}}^{Link}$ the probability of $\mathcal{A}$ winning the game.

Definition 6.7 (Linkability). *An LTRS scheme is linkable if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{Link}$ is negligible.*

Non-Slanderability.

Non-Slanderability for LTRS schemes is defined in the following game between the Simulator $\mathcal{S}$ and the Adversary $\mathcal{A}$ in which $\mathcal{A}$ is given access to oracles $\mathcal{JO}$, $\mathcal{CO}$ and $\mathcal{SO}$:

1. $\mathcal{S}$ generates and gives $\mathcal{A}$ the system parameters param.

2. $\mathcal{A}$ may query the oracles according to any adaptive strategy.

3. $\mathcal{A}$ gives $\mathcal{S}$ a signature $\sigma_1 \in \Sigma$ and a tuple $(n_2, d_2, \mathcal{Y}_2, M_2, \sigma_2)$.

$\mathcal{A}$ wins the strong game if (1) Verify(M_2,σ_2')=1, (2) Link(σ_1', σ_2'))=1, and (3) none of the public keys in $\mathcal{V}$ has been input to $\mathcal{CO}$. $\mathcal{A}$ wins the weak game if the following additional constraint holds: (4) the signature σ_1 is a query output of $\mathcal{SO}$. (Let $(e, n_1, d_1, \mathcal{Y}_1, \mathcal{V}, M_1)$ be the associated input tuple), We denote by $\mathbf{Adv}_{\mathcal{A}}^{SNS}(\lambda)$ (resp. $\mathbf{Adv}_{\mathcal{A}}^{WNS}(\lambda)$) the probability of $\mathcal{A}$ winning the strong (resp. weak) game.

Definition 6.8 (Non-Slanderability). *An LTRS scheme is strongly (resp. weakly) non-slanderable if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{SNS}(\lambda)$ (resp. $\mathbf{Adv}_{\mathcal{A}}^{WNS}(\lambda)$) is negligible.*

Security.

Summarizing we have:

Definition 6.9 (Security of LTRS Schemes). *An LTRS scheme is secure if it is unforgeable, linkably-anonymous, linkable and weakly non-slanderable.*

6.5 Our Construction

6.5.1 An Linkable Threshold SAG Signature Scheme

In this section, we give a concrete construction of an LTRS scheme. We then show that such a construction is secure under the security model defined in the previous section.

- Key-Gen. On input a security parameter ℓ_i, the algorithm randomly picks two distinct primes p_i, q_i of the form $p_i = 2p'_i + 1$ and $q_i = 2q'_i + 1$, where p'_i, q'_i are both $((\ell_i - 2)/2)$-bit primes, and sets $N_i \leftarrow p_i q_i$. It then picks a random generator g_i of $QR(N_i)$ and a random $x_i \in_R \mathbf{Z}_{p'_i q'_i}$ and computes $y_i \leftarrow g_i^{x_i}$. It picks a strong collision-resistant hash function $H_i : \{0,1\}^* \rightarrow \{h | \langle h \rangle = QR(N_i)\}$. It sets the public key to $pk_i \leftarrow (\ell_i, N_i, g_i, y_i, H_i)$, and the secret key to $sk_i \leftarrow (p_i, q_i, x_i)$. Finally it outputs (sk_i, pk_i).

- Init. On input security parameters $\ell \in \mathbf{N}$, $1 < \epsilon \in \mathbf{R}$ and $\kappa \in \mathbf{N}$, the algorithm randomly picks a κ-bit prime q and a strong collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbf{Z}_q$. It outputs the system parameters $\mathsf{param} = (\ell, \epsilon, \kappa, q, H)$.

- Sign. On input the system parameters $\mathsf{param} = (\ell, \epsilon, \kappa, q, H)$, an event-id $e \in \{0,1\}^*$, a group size $n \in \mathbf{N}$, a threshold $d \in \{1, \ldots, n\}$, a public key set $\mathcal{Y} = \{pk_1, \ldots, pk_n\}$, where each $pk_i = (\ell_i, N_i, g_i, y_i, H_i)$ is s.t. $\ell_i \geq \ell$, a private key set $\mathcal{X} = \{sk_{\pi_1}, \ldots, sk_{\pi_d}\}$, where each $sk_{\pi_i} = (p_{\pi_i}, q_{\pi_i}, x_{\pi_i})$ corresponds to $pk_{\pi_i} \in \mathcal{Y}$, and a message $M \in \{0,1\}^*$, Define $\mathcal{N} = \{1, \ldots, n\}$ and $\mathcal{I} = \{\pi_1, \ldots, \pi_d\} \subseteq \mathcal{N}$, the algorithm does the following:

  1. For all $i \in \mathcal{N}$, compute $h_{i,e} \leftarrow H_i(\mathsf{param}, pk_i, e)$ and the tags
  $$\tilde{y}_{i,e} \leftarrow \begin{cases} h_{i,e}^{x_i}, & i \in \mathcal{I}; \\ h_{i,e}^{a_i}, & i \in \mathcal{N} \backslash \mathcal{I},\ a_i \overset{R}{\leftarrow} \mathbf{Z}_{\lfloor N_i/4 \rfloor}. \end{cases}$$
  2. Compute a signature $(f, s_1, \ldots, s_n)$ for
  $$SPK\left\{(\alpha_1, \ldots, \alpha_n) : \bigvee_{\mathcal{J} \subseteq \mathcal{N}, |\mathcal{J}| = d} \left(\bigwedge_{i \in \mathcal{J}} y_i = g_i^{\alpha_i} \wedge \tilde{y}_{i,e} = h_{i,e}^{\alpha_i} \right) \right\}(M).$$
  In particular, this requires the knowledge of $x_{\pi_1}, \ldots, x_{\pi_d}$. We will refer to this signature scheme as SPK_1.
  3. Compute a signature $(c, s'_1, \ldots, s'_n)$ for
  $$SPK\left\{(\beta_1, \ldots, \beta_n) : \bigwedge_{i=1}^{n} \tilde{y}_{i,e} = h_{i,e}^{\beta_i} \right\}(M).$$
  In particular, this requires the knowledge of x_i for all $i \in \mathcal{I}$ and a_i for all $i \in \mathcal{N} \backslash \mathcal{I}$. We will refer to this signature scheme as SPK_2.
  4. The signature is
  $$\sigma \leftarrow \langle ((\tilde{y}_{1,e}, \ldots, \tilde{y}_{n,e}), (f, s_1, \ldots, s_n), (c, s'_1, \ldots, s'_n) \rangle.$$
  Note that a signature is composed of three parts: the tags, a signature for SPK_1 and a signature for SPK_2.

- Verify. On input a tuple $(\mathsf{param}, e, n, d, \mathcal{Y}, M, \sigma)$, the algorithm parses param into $(\ell, \epsilon, \kappa, q, H)$, $\mathcal{Y}$ into $\{pk_1, \ldots, pk_n\}$, where $pk_i = (\ell_i, N_i, g_i, y_i, H_i)$, and σ into $\langle((\tilde{y}_1,\ldots,\tilde{y}_n),(f,s_1,\ldots,s_n),(c,s'_1,\ldots,s'_n))\rangle$. If any $\ell_i < \ell$, the algorithm returns with 0. Otherwise it does the following:

1. For $i \in \mathcal{N}$, compute $h_{i,e} \leftarrow H_i(\mathsf{param}, pk_i, e)$.
2. Verify if $(f, s_1, \ldots, s_n)$ is a correct signature for SPK_1.
3. Verify if $(c, s'_i, \ldots, s'_n)$ is a correct signature for SPK_2.

- Link. On input a tuple $(\mathsf{param}, e, (n_1, d_1, \mathcal{Y}_1, M_1, \sigma_1), (n_2, d_2, \mathcal{Y}_2, M_2, \sigma_2))$ s.t., for $j = 1, 2$, $\mathsf{Verify}(M_j, \sigma'_j)=1$, the algorithm first parses, for $j = 1, 2$, $\mathcal{Y}_j$ into $\mathcal{Y}_j = \{pk_1^{(j)}, \ldots, pk_{n_j}^{(j)}\}$ and σ_j into

$$\langle((\tilde{y}_{1,e}^{(j)},\ldots,\tilde{y}_{n,e}^{(j)}),(f^{(j)},s_1^{(j)},\ldots,s_n^{(j)}),(c^{(j)},s_1'^{(j)},\ldots,s_n'^{(j)}))\rangle.$$

  If there exists $\pi_1 \in \{1, \ldots, n_1\}$ and $\pi_2 \in \{1, \ldots, n_2\}$ s.t. $pk_{\pi_1}^{(1)} = pk_{\pi_2}^{(2)}$ and $\tilde{y}_{\pi_1,e}^{(1)} = \tilde{y}_{\pi_2,e}^{(2)}$, it returns 1 and additionally $pk_{\pi_1}^{(1)}$. Otherwise it returns 0.

The correctness of the about signature scheme is straightforward.

6.5.2 Security

We state the security theorems here and provide proof sketches.

Theorem 6.10 (Unforgeability). *Our construction is unforgeable under the Strong RSA assumption in the random oracle model.*

(*Proof Sketch*) Roughly speaking, our underlying SAG signature is similar to [63] which has proven unforgeable, and that implies unforgeability with our linkable SAG signatures. □

Theorem 6.11 (Linkable-anonymity). *Our construction is anonymous under the Strong RSA assumption and DDH over QR(N) assumption in the random oracle model.*

Proof. Simulating Signing Oracle, $\mathcal{SO}$: Upon input (e, n, d, $\mathcal{Y}$, $\mathcal{V}$, $\mathcal{X}$, M), generate a valid signature as follows: For each $i \in \mathcal{Y} \backslash \mathcal{V}$, randomly generate a_i and compute $\tilde{y}_{i,e} = h_{i,e}^{a_i}$. For each $i \in \mathcal{V}$, randomly generate a_i and backpatch the random oracle to $h_{i,e} = H_i(\mathsf{param}, pk_i, e) = g_i^{a_i}$ and compute $\tilde{y}_{i,e} = y^{a_i}$. Ensure consistency with other oracles from the beginning. Generate $c_0, \cdots, c_n$ such that they interpolate a polynomial f with degree $\leq n - d$ and $f(i) = c_i$ for $0 \leq i \leq n$. For each i, simulate the corresponding 3-move conversation in Step (2) of Sign with randomly generated responses $s_1, \cdots, s_n$ to produce the commitments. Backpatch the random oracle so that the commitments are hashed to c_0. This completes up to Step (2) in Sign. The rest is easy: Randomly generate challenge c, simulate the SPK in Step (3) of Sign with randomly generate responses $s'_1, \cdots, s'_n$.

Setting up the gauntlet for solving DDH: Similar to proof of anonymity in [57]. Let Q_J be the number of $\mathcal{JO}$ queries. Denote the Gauntlet DDH Problem as ($\hat{N}$, $\hat{g}$, $\hat{g}^{\alpha}$, $\hat{g}^{\beta}$, $\hat{g}^{\gamma}$) where $\gamma = \alpha\beta$ with probability 1/2. In the Gauntlet Phase, Simulator $\mathcal{S}$ sets up the witness extraction mechanism as follows: Randomly select $i^* \in \{1, \cdots, Q_J\}$. Return $pk^* \leftarrow (\hat{l}, \hat{N}, \hat{g}, \hat{g}^{\alpha}, \hat{H})$ in the i^*-th $\mathcal{JO}$ query, backpatch Random Oracle $\mathcal{HO}_{i^*}$ to $h_{i,e} = \hat{g}^{\beta}$. There is a non-negligible probability that $pk^* = y_g$, the gauntlet public key. Generate the Gauntlet signature σ'_g with $\tilde{y}_{i,e} = \hat{g}^{\gamma}$ and simulate the SPK's. With 1/2 probability, $\alpha\beta = \gamma$ and it can be shown that the gauntlet signature is indistinguishable from one generated using Sign. Otherwise, with 1/2 probability, $\alpha\beta \neq \gamma$ and it can be shown that σ'_g is indistinguishable from one generated using $\mathcal{SO}$.

If $\mathcal{A}$ returns $\hat{b} = 1$, $\mathcal{S}$ answers Yes to the DDH question. Otherwise, $\mathcal{S}$ answers No. $\mathcal{S}$'s advantage in DDH equals $\mathcal{A}$'s advantage in winning Game LA. □

Theorem 6.12 (Linkability). *Our construction is linkable under the Strong RSA assumption in the random oracle model.*

Proof. Similar to proof of linkability in [57]. If Adversary can produce two unlinked signatures, then he is rewound twice to produce two sets of witnesses of set-size d_1 and d_2 respectively. If the two sets overlap, then the threshold signatures should have already been linked. If the two sets do not overlap, then we would have obtained a total of $d_1 + d_2$ witnesses while Adversary only corrupted at most d_1+d_2-1 witnesses, which have negligible probability under the Strong RSA assumption. $\square$

Theorem 6.13 (Non-slanderability). *Our construction is weakly non-slanderable under the Strong RSA assumption in the random oracle model.*

Proof. The weak non-slanderability is protected by Step (3) of Sign. Given a signature from $\mathcal{SO}$, Adversary does not know the discrete logarithm of any $\tilde{y}_i$, and therefore cannot produce a signature containing some $\tilde{y}_j$ and prove knowledge of logarithm of $\tilde{y}_j$ as in Sign's Step (3). $\square$

Remark: Our scheme does not have strong non-slanderability: User j and User k can agree to use the same $\tilde{y}_i$ to slander User i. User i can vindicate himself by proving that logarithm of his public key y_i does not equal to the logarithm of $\tilde{y}_i$. However, that can be a hassle.

Summarizing, we have:

Theorem 6.14 (Security). *Our construction is secure under the Strong RSA Assumption and the DDH over QR(N) Assumption in the random oracle model.*

6.5.3 Discussions

Separable Linkable SAG Signatures. We achieved separable linkable SAG signatures where individual users choose their own safe RSA modulus N_i. In our construction, individual user's key pair are constrained to reside in Discret Logarithm (DL) over a composite moduli. In fact, our method can be easily modified to allow user key pairs from DL over a prime modulus, i.e. $(sk, pk)=$ $(x,\ y = g^x \pmod{P_i})$. Therefore, our signatures can be easily modified to support a mixture of composite DL and prime DL.

Ring-type SAG signature. Although our construction utilizes the CDS-type structure, meaning the structure from Cramer, et al. [58], the technique can be easily adapted to construct the first separable linkable SAG signature of the RST-type, meaning the structure from Rivest, et al. [5]. Simply follow [57] but use different $\tilde{y}_i$ for different users i instead using a single $\tilde{y}$. $\tilde{y}_i = h^{a_i}$ with randomly generated a_i except $\tilde{y}_s = h^{x_s}$ with signer s. Then simulate the Proof-of-Knowledge $\{(x_i) : y_i = g^{x_i} \wedge \tilde{y}_i = h^{x_i}\}$ along the *ring*, computing $Hash(commitments_i) = challenge_{i+1}$ and simulating, except for the actual signer. The resulting linkable SAG signature is separable, supporting a mixture of composite DL and prime DL key pairs.

Bandwidth Efficiency. The length of our signature is $O(n)$ (n being the group size). This improves upon [57] whose length if $O(nd)$. However, our scheme is not non-interactive while [57] is.

Event-IDs. Event-ids should be chosen carefully to according specific applications. We give two examples here. (1) When an event-oriented linkable (threshold) SAG signature scheme is used to leak sequences of secrets, the whistle-blower should choose a unique event-id when leaking the first secret and stick to using the same in the sequel. This makes sure that the sequence of secrets cannot be linked to other sequences. (2) When used in electronic voting, it is usually the voting organizer (e.g. the government) who decides on an event-id. Each eligible voter should therefore, before they cast a vote, make sure that the event-id has not been used in any previous voting event, so as to secure the intended privacy.

Linkability in Threshold SAG Signatures. Linkability in threshold SAG signatures requires a more precise definition. In particular, there are two possible flavors: two signatures are linked if and only if (1) they are signed by exactly the same set of signers, or (2) they involve a common signer. We call signatures of the former type "*coalition-linkable*" while those of the latter type "*individual-linkable*".

In a coalition-linkable scheme, users are able to sign multiple times without their signatures being linked, as long as they are not collaborating with exactly the same set of signers again. However, in an individual-linkable scheme, a user signing more than once will have the signatures linked, no matter who other collaborating signers are. The scheme we present in this paper falls into the later category.

Chapter 7

Conclusion

In our thesis, we introduce the first blind spontaneous group signature scheme with thresholding option. With our modular construction, any SAG signature schemes can be combined with blind signature by using transfer proof of knowledge technique. It inherits special features from these two signatures to create new and created new properties: the actual signer cannot point out whatever a message-signature pair is generated by oneself. We managed to prove the security of both ring-type and the CDS type blind SAG signature is reducible to Schnorr's Randomized Oversampled Solvable (ROS) assumption. It is existential unforgeable under adaptive chosen message attack in the generic group model and random oracle model. Both signer anonymity and message-signer linkability remains unconditional. This perfect anonymity properties assures the privacy the user and can help enforce the freedom of speech in many applications.

Follows we given here the first separable linkable SAG signature scheme, which also supports an efficient thresholding option. We have also presented the security model and reduce the security of our scheme to well-known hardness assumptions. In particular, we have introduced the security notions of *accusatory linkability* and *non-slanderability* to linkable SAG signatures. The anonymity and linkability provides a good starting point for applications such as electronic cash and electronic voting systems.

There are still a number of open problems in this area such as to provide an efficient way to provide efficient proof of interacting protocols or an constructing short linkable signatures. We believe there remain plenty of research opportunities in this field.

Bibliography

[1] R. Cramer, I. Damgård, and B. Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols, in *Proc. CRYPTO 94*, pages 174–187, Springer-Verlag, 1994, Lecture Notes in Computer Science No. 839.

[2] R. Rivest, A. Shamir, and Y. Tauman, How to leak a secret, in *Proc. ASIACRYPT 2001*, pages 552–565, Springer-Verlag, 2001, Lecture Notes in Computer Science No. 2248.

[3] M. Abe, M. Ohkubo, and K. Suzuki, 1-out-of-n signatures from a variety of keys, in *Proc. ASIACRYPT 2002*, pages 415–432, Springer-Verlag, 2002, Lecture Notes in Computer Science No. 2501.

[4] D. Chaum and E. van Heyst, Group signatures, in *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265, Springer-Verlag, 1991.

[5] R. L. Rivest, A. Shamir, and Y. Tauman, How to leak a secret, in *ASIACRYPT 2001*, pages 552–565, Springer-Verlag, 2001.

[6] M. Bellare and P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73, ACM Press, 1993.

[7] C. P. Schnorr, Security of blind discrete log signatures against interactive attacks, in *ICICS*, Springer, 2001, Lecture Notes in Computer Science No. 2229.

[8] J. Daemen and V. Rijmen, J. daemen and v. rijmen. aes proposal: Rijndael. www.esat.kuleuven.ac.be/ rijmen/rijndael/.

[9] R. Rivest, The rc4 encryotion algorithm, 1992.

[10] R. L. Rivest, A. Shamir, and L. M. Adleman, Communications of the ACM **21**, 120 (1978).

[11] E. Biham and R. Chen, Near-collisions of sha-0., in *CRYPTO*, pages 290–305, 2004.

[12] M. O. Rabin, Digitalized signatures as intractable as factorization, Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[13] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby, SIAM J. Comput. **28**, 1364 (1999).

[14] W. Diffie and M. E. Hellman, IEEE Trans. Inform. Theory , IT (1976).

[15] D. Boneh, B. Lynn, and H. Shacham, Short signatures from the weil pairing, in *Proc. ASIACRYPT 2001*, pages 514–532, Springer-Verlag, 2001, Lecture Notes in Computer Science No. 2248.

[16] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, in *Proc. EUROCRYPT 2003*, pages 416–432, Springer-Verlag, 2003, Lecture Notes in Computer Science No. 2656.

[17] A. Joux and K. Nguyen, Separating decision diffie-hellman from diffie-hellman in cryptographic groups, Report 2001/003, 2001.

[18] J. Camenisch and M. Michels, A group signature scheme based on an RSA-variant, rs RS-98-27, brics, 1998.

[19] E. Fujisaki and T. Okamoto, Statistical zero knowledge protocols to prove modular polynomial relations, in *CRYPTO'97*, pages 16–30, Springer-Verlag, 1997.

[20] E. Fujisaki and T. Okamoto, A practical and provably secure scheme for publicly verifiable secret sharing and its applications, in *Eurocrypt '98*, volume 1403 of *LNCS*, pages 32–46, Springer-Verlag, 1998.

[21] O. Goldreich, *Foundation of Cryptography, Book 1*, 2001, Cambridge University Press.

[22] A. Fiat and A. Shamir, How to prove yourself: Practical solution to identification and signature problems, in *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Springer-Verlag, 1987.

[23] D. Pointcheval and J. Stern, Security proofs for signature schemes, in *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398, Springer-Verlag, 1996.

[24] S. Goldwasser, S. Micali, and R. L. Rivest, SIAM J. Comput. **17**, 281 (1988).

[25] M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73, ACM Press, 1993.

[26] J. Camenisch and M. Stadler, Efficient group signature schemes for large groups (extended abstract), in *CRYPTO'97*, pages 410–424, Springer-Verlag, 1997.

[27] A. Fiat and A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in *Proc. CRYPTO 86*, pages 186–194, Springer-Verlag, 1987, Lecture Notes in Computer Science No. 263.

[28] R. Canetti, O. Goldreich, and S. Halevi, The random oracle methodology, revisited., in *Proc.* 13*th ACM Symp. on Theory of Computing*, pages 209–128, ACM Press, 1998.

[29] S. Goldwasser and Y. Tauman, ePrint **2003**.

[30] V. Nechaev, Mathematical Notes 55 , 165 (1994).

[31] V. Shoup, Lower bounds for discrete logarithms and related problems, in *Proc. EUROCRYPT 97*, pages 256–266, Springer-Verlag, 1997, Lecture Notes in Computer Science No. 1233.

[32] W. Diffie and M. E. Hellman, IEEE Transactions on Information Theory **IT-22**, 644 (1976).

[33] C. Schnorr, Efficient identication and signatures for smart cards, in *Proc. CRYPTO 89*, pages 239–252, Springer-Verlag, 1990, Lecture Notes in Computer Science No. 435.

[34] T. Okamoto, Provably secure and practical identification schemes and corresponding signature schemes, in *Proc. CRYPTO 92*, pages 31–53, Springer-Verlag, 1993, Lecture Notes in Computer Science No. 740.

[35] D. Chaum and E. van Heyst, Group signatures, in *Proc. EUROCRYPT 91*, pages 257–265, Springer-Verlag, 1991, Lecture Notes in Computer Science No. 547.

[36] A. Shamir, How to share a secret, in *Communications of the ACM*, volume 22(2), pages 612–613, ACM Press, 1979.

[37] G. R. Blakley, Safeguarding cryptographic keys, in *Proc. AFIPS National Computer Conference*, volume 48, pages 313–317, 1979.

[38] Y. Desmedt, Some recent research aspects of threshold cryptography, in *Proc. First International Workshop on Information Security, ISW 97*, pages 158–173, Springer-Verlag, 1997, Lecture Notes in Computer Science No 1196.

[39] A. Herzberg, S. Jarecki, , and H. K. M. Yung, Proactive sercret sharing, in *Proc. CRYPTO 95*, pages 339–352, Springer-Verlag, 1995, Lecture Notes in Computer Science No. 963.

[40] Y. Frankel, P. Gemmall, P. Mackenzie, and M. Yung, Proactive rsa, in *Proc. CRYPTO 97*, pages 440–454, Springer-Verlag, 1997, Lecture Notes in Computer Science No. 1294.

[41] T. Rabin, A simplified approach to threshold and proactive rsa, in *Proc. CRYPTO 98*, pages 89–104, Springer-Verlag, 1998, Lecture Notes in Computer Science No. 1462.

[42] M. Bellare and S. K. Miner, A forward-secure digital signature scheme., in *CRYPTO*, pages 431–448, 1999.

[43] M. Abdalla and L. Reyzin, A new forward-secure digital signature scheme., in *ASIACRYPT*, pages 116–129, 2000.

[44] Y. Dodis, A. Sahai, and A. Smith, On perfect and adaptive security in exposure-resilient cryptography., in *EUROCRYPT*, pages 301–324, 2001.

[45] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, Robust threshold DSS signatures, in *Proc. EUROCRYPT 96*, pages 354–371, Springer-Verlag, 1996, Lecture Notes in Computer Science No. 1070.

[46] A. Boldyreva, Efficient threshold signature, multisignature, and blind signature schemes based on the Gap-Diffie-Hellman-group signature scheme, in *PKC'03*, pages 31–46, Springer-Verlag, 2003, Lecture Notes in Computer Science No. 2567.

[47] Y. Frankel and Y. Desmedt, Parallel reliable threshold multisignature, in *Technical Report TR-92-04-02*, University of Wisconsin-Milwaukee, 1992.

[48] C. Li, M. Hwang, and N. Lee, Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders, in *Proc. EUROCRYPT 94*, pages 194–204, Springer-Verlag, 1994, Lecture Notes in Computer Science No 950.

[49] V. Shoup, Practical threshold signature, in *Proc. EUROCRYPT 2000*, pages 207–220, Springer-Verlag, 2000, Lecture Notes in Computer Science No. 1807.

[50] E. Bresson, J. Stern, and M. Szydlo, Threshold ring signatures and applications to ad-hoc groups, in *Proc. CRYPTO 2002*, pages 465–480, Springer-Verlag, 2002, Lecture Notes in Computer Science No. 2442.

[51] D. Chaum, CACM **29**, 1030 (1985).

[52] J. Camenisch and A. ', An efficient system for non-transferable anonymous credentials with optional anonymity revocation, in *Proc. EUROCRYPT 2001*, pages 93–118, Springer-Verlag, 2001, Lecture Notes in Computer Science No. 1294.

[53] J. K. Liu, V. K. Wei, and D. S. Wong, **2004** (2004).

[54] D. Wagner, A generalized birthday problem, in *Proc. CRYPTO 2002*, pages 288–303, Springer-Verlag, 2002, Lecture Notes in Computer Science No. 2442.

[55] M. Bellare and D. Micciancio, a new paradigm for collision-free hasing: incrementality at reduced cost, in *Proc. EUROCRYPT 97*, Springer-Verlag, 1997, Lecture Notes in Computer Science No. 1233.

[56] J. K. Liu, V. K. Wei, and D. S. Wong, eprint **2004** (2004).

[57] J. K. Liu, V. K. Wei, and D. S. Wong, Linkable spontaneous anonymous group signature for ad hoc gr oups (extended abstract), in *ACISP'04*, volume 3108 of *LNCS*, pages 325–335, Springer-Verlag, 2004.

[58] R. Cramer, I. Damgard, and B. Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols, in *CRYPTO'94*, pages 174–187, Springer-Verlag, 1994.

[59] A. Shamir, Commun. ACM **22**, 612 (1979).

[60] M. Bellare, D. Micciancio, and B. Warinschi, Foundations of group signatures: formal definitions, simplified requirements and a construction based on general assumptions, in *EUROCRYPT'03*, volume 2656 of *LNCS*, Springer-Verlag, 2003.

[61] M. Bellare, H. Shi, and C. Zhang, Foundations of group signatures: the case of dynamic groups, Cryptology ePrint Archive, Report 2004/077, 2004, http://eprint.iacr.org/.

[62] A. Kiayias and M. Yung, Group signatures: provable security, efficient constructions, and anonymity from trapdoor-holders, Cryptology ePrint Archive, Report 2004/076, 2004, http://eprint.iacr.org/.

[63] J. K. Liu, V. K. Wei, and D. S. Wong, A separable threshold ring signature scheme, in *ICISC 2003*, volume 2971 of *LNCS*, pages 12–26, Springer-Verlag, 2003.