Group-Oriented Encryption and Signature

Student: Ming-Luen Wu
Advisor: Professor Yuh-Dauh Lyuu
Department of Computer Science and Information Engineering
National Taiwan University

Abstract

Computer networks bring tremendous progress to the information-based society. Companies, organizations, and governments have been using computers and networks to process or transmit digital data. But this also results in many different types of security requirements for group-oriented cryptographic applications.

In this thesis we study existing cryptographic tools and then use them to design more complex cryptographic systems. Several fundamental cryptographic primitives are useful not only as stand-alone applications but also as building blocks in the designing of secure cryptographic objects. Using these building blocks, we develop new cryptographic applications, including a full public-key traitor-tracing scheme and a convertible group undeniable signature scheme.

A fully public-key traitor-tracing scheme is a public-key traitor-tracing scheme that allows a subscriber to choose his or her own private decryption key without others learning the key. The distributor of the digital content uses the public data coming from all subscribers to compute a public encryption key. The paid contents are then transmitted to the subscribers, after being encrypted with the public key. Each subscriber can decrypt the data using his or her own secret key. Even if a coalition of subscribers conspire to create a pirate decoder with a tamper-free decryption key, there is a tracing algorithm to trace them. A realization of the scheme is presented in this thesis. Our scheme is long-lived, which means that the subscribers' secret keys need not be regenerated after the pirate key is detected or when subscribers join or leave the system. Finally, our scheme guarantees anonymity.

A group undeniable signature satisfies the following requirements: (1) only group members can anonymously sign on behalf of the group; (2) a verifier must interact with the group manager to verify the signature; (3) the group manager can identify the signer of a valid signature. A convertible group undeniable signature scheme allows the group manager to turn select group undeniable signatures into universally verifiable group signatures. An efficient realization of the scheme is proposed in this thesis. Our scheme is unforgeable, exculpable, unlinkable, and coalition-resistant. The proposed scheme allows the group manager to delegate the ability to confirm

and deny signatures to trusted parties. The sizes of the public key and signatures are independent of the group size.

Contents

Chapter 1

Introduction

Due to the widespread use of computers and communication networks, group-oriented cryptographic techniques are rapidly becoming important concerns for secure data exchange. In this chapter, we give an overview of group-oriented encryption and signature applications.

1.1 Background

Confidentiality and authenticity of a message are two fundamental issues in cryptography. Confidentiality ensures that no adversary will learn anything about the private information held by honest parties. Authenticity ensures that the receiver of a message can verify that the message really comes from the alleged sender. To achieve the two important goals, encryption and signature techniques are used.

In classic encryption schemes, two parties who wish to securely communicate would have to pre-agree on a specific secret key that would help the encryption and decryption. Hence, there must be a secure method to deliver the secret key in advance. However, the process of agreeing on a secret key can be a rather difficult task. In 1976, Diffie and Hellman [78] first introduce a key agreement scheme and introduce the significant notion of public-key cryptography. After that, several important public-key cryptosystems are proposed, such as the RSA and the ElGamal schemes. In a public-key cryptosystem, a public key is used for encryption and a secret key for decryption. The two keys are derived in such a way that computing the secret

key from the public key is computationally infeasible. Furthermore, on the basis of public-key cryptosystems, digital signature schemes can be devised. A signer creates digital signatures with his secret key, and everyone can verify the signatures with the corresponding public key. Analogous to handwritten signatures, digital signatures should be easy to produce and verify, but difficult to forge.

Originally, the encryption and signature schemes are implemented for individual privacy and individual signatures, i.e., a private message is intended for an individual and a signature is created on behalf of an individual. In this setting, often only two parties are involved. However, such a framework might not be sufficient for cryptographic applications. In many cases, the messages created for a group are of much greater importance, and have far more serious consequences than individual messages. It is desirable that there are cryptographic algorithms and protocols suitable for a group of people communicating over an insecure computer network. Accordingly, group-oriented cryptographic techniques are becoming significant considerations in today's information-based society [76].

In particular, many companies, organizations, and government departments have widely used computers and networks to process or transmit digital data. This promotes cryptographic applications involving more than two parties. For example, consider a data provider such as a wired broadcast station needs to broadcast data to a lot of subscribers securely or an employee wants to sign messages on behalf of his company. Because every group may have different needs in different cases, group-oriented cryptographic applications have many varieties. To design appropriate security services, the first step is to determine the group's requirements. It is evident that group-oriented cryptographic applications have more complex requirements than two-party situations. To gear toward more complex applications, understanding useful cryptographic tools is necessary. In this thesis, we clarify numerous fundamental concepts in cryptography, present several elementary cryptographic tools, and propose two useful group-oriented cryptographic applications.

1.2 Group-Oriented Encryption

We say that an encryption scheme is group-oriented if the parties involved in encryption and decryption are more than two in number. To date, many group-oriented encryption applications have been addressed. In the following, we review well-known applications that have appeared in the literature.

1. **Broadcast encryption.** Consider the problem of broadcasting digital contents to a large set of authorized users. Such applications include paid-TV systems, copyrighted CD/DVD distributions, and fee-based online databases. The problem is that anyone connected to a broadcast channel is able to pick up the data, whether they are authorized or not. To prevent unauthorized users from extracting data, the broadcaster encrypts the message and only the authorized users have the decryption keys to recover the data. This issue of secure broadcasting is first addressed in [56]. However, the proposed method carries out n encryptions for each copy of data, where n is the number of subscribers. To improve efficiency, bandwidth requirements, and the keys' storage space, see [12, 16, 17, 56, 89, 127, 143] for further studies.

2. **Traitor tracing.** In broadcast encryption, malicious authorized users, called traitors, may use their personal decryption keys to create a pirate decoder. The resulting pirate decoder allows an unauthorized user to extract the context. To discourage authorized users from revealing their keys, traitor tracing is first introduced by Chor, et al. [57, 58] and studied further in [96, 113, 169, 170, 181, 211, 213]. The idea is an algorithm that uses the confiscated pirate decoder to track down at least one colluder without wrongly accusing noncolluders with high probability. Most of these traitor-tracing schemes use a secret-key encryption scheme to encrypt data. Public-key traitor-tracing schemes are studied in [19, 128, 134, 135]. A public-key traitor tracing allows everyone to perform encryption, and thus anyone can broadcast messages to authorized users securely.

3. **Threshold cryptosystems.** Within a group, various access policies are possible. Depending on the internal organization of the group and the access type of the message imposed by the sender, a different cryptographic scheme with

the corresponding key management policy is needed. Threshold cryptosystems allow one to send encrypted messages to a group, while only a group achieving a "threshold" has the ability to reconstruct the plaintext. Moreover the process of reconstructing a plaintext should not reveal any participant's secret. Threshold cryptosystems are initiated by Desmedt and Frankel in [76, 77] and studied further in [40, 137, 207]. We remark that the "threshold" can be an arbitrary access structure, such as hierarchical structures [101] or t out of n threshold structures [77]. The latter is the usual case.

One focus of this thesis is to investigate traitor tracing. We are the first to introduce the concept of fully public-key traitor tracing and propose a scheme [145, 146]. A fully public-key traitor-tracing scheme is a public-key traitor-tracing scheme in which subscribers can prevent anyone (including the broadcaster) from learning their secret keys. We present the scheme in Chapter 4.

1.3 Group-Oriented Signature

We say that a signature scheme is group-oriented if signing and verification involve more than two parties. Because a signature scheme can often be turned into an authentication scheme (such as the identification schemes in [87, 90]), in the following we will review many group-oriented signature applications that have appeared in the literature.

1. **Group signatures.** A group signature scheme allows a group member to sign messages on behalf of the group without revealing his or her identity. Nevertheless, in case of a later dispute, a designated group manager can open the signature, thus tracing the signer. At the same time, any one—including the group manager—cannot misattribute a valid signature. The concept of group signatures is introduced by Chaum and van Heyst [44]. Camenisch and Stadler present the first scheme in which the sizes of the public key and signatures are independent of the group size [38]. More works on group signatures include [4, 5, 6, 34, 35, 39, 54, 55, 180]

2. **Group identification.** A group identification scheme allows a group member

to convince other parties that he is a member of a group without revealing his identity. For efficiency and different security considerations, many schemes have been proposed in the literature [20, 37, 130, 138, 200, 201]. In particular, Kilian and Petrank introduce identity escrow that is a group identification scheme with revocable anonymity [130]. In a identity escrow scheme, there is a delegated escrow agent to trace the group member that has proved to other parties his group membership. We notice that the property of traceability is also an important requirement for group signatures. Furthermore, Boneh and Franklin introduce identity escrow with subset queries by which a group member can demonstrate membership in an arbitrary subset of groups members in [20]. Camenisch and Lysyanskaya introduce identity escrow with appointed verifiers by which a group member can only convince the appointed verifiers of his membership in [37].

3. **Anonymous credential systems.** A anonymous credential system allows users to obtain credentials from organizations and to demonstrate possession of these credentials anonymously. The property of anonymity requires that the same user can not be linked even if he carries out a lot of demonstrations. In addition, several desirable properties have been considered. For example, non-transferability discourages the users from lending their credentials to others. Anonymous credential systems are initiated by Chaum [52] and further studied in [25, 36, 50, 53, 71, 144].

4. **Multisignatures.** Multisignatures allow more than one user to sign messages together, and anyone can identify the individual signers. Multisignatures are first introduced by Itakura and Nakamura [123] and have been extensively studied in the literature [23, 117, 120, 139, 161]. Several desirable properties have been suggested. For example, Micali et al. propose accountable-subgroup multisignatures by which any subset of users can sign messages and each signer can be identified universally [161]. They refer to the two properties as flexibility and accountability, respectively.

5. **Threshold signatures.** A threshold signature scheme is a generalization of digital signatures, in which only the persons achieving a "threshold" can gener-

ate a valid signature. For example, a (t, n) threshold signature scheme requires the cooperation of t or more persons for generating a valid signature, where n is the group size and $t \leq n$. Threshold signatures are first introduced by Desmedt and Frankel [76, 77] and studied further in [99, 100, 117].

6. **Threshold group signatures.** A threshold group signature scheme is a generalization of group signatures, in which only the members achieving a "threshold" can represent the group to generate signatures anonymously and the identities of signers of a signature can be revealed in case of later disputes. An example is a (t, n) threshold group signature by which the cooperation of t or more group members is necessary to generate signatures on behalf of the group, where n is the group size and $t \leq n$. This definition is first presented in [219]. (The (t, n) threshold group signature schemes are called as (t, n) threshold-multisignature schemes in [139, 140]. We notice that in a multisignature scheme the identities of signers are often public and the public keys of signers are needed to verify a signature. At the same time, anonymity and traceability are two essential properties of a group signature scheme. Hence, it is more accurate to call the (t, n) threshold-multisignature schemes in [139, 140] (t, n) threshold group signature schemes.)

In this thesis, we introduce a new type of signature called a group undeniable signature. We also propose the first convertible group undeniable signature scheme. The detailed contents will be presented in Chapter 5. Here we give a simple description of a (convertible) group undeniable signature.

Group undeniable signatures. A group undeniable signature scheme allows a group member to sign on behalf of a group without revealing his identity, and the verification of a signature can only be done with cooperation of the group manager. Furthermore, the group manager must be able to track down the signers in case of a later dispute [147, 149].

Convertible group undeniable signatures. A convertible group undeniable signature scheme is a group undeniable signature in which the group manager can turn selective group undeniable signatures into ordinary group signatures

without compromising the security of the secret key needed to generate signatures. Obviously, convertible group undeniable signatures are more powerful than group signatures [148, 150].

1.4 Contributions and Organization of the Thesis

Chapter 2 provides mathematical and cryptographic foundations. We introduce basic contents of algebra, number theory, and complexity theory, which underly our schemes in this thesis. We also clarify some notions that are important in cryptography, including intractable problem assumptions and randomized algorithms. Moreover, definitions of many fundamental cryptographic terms are summarized: indistinguishability of probability ensembles, interactive proof systems, proofs of knowledge, zero-knowledge proof systems, witness indistinguishability and hiding, and hash functions.

In Chapter 3, we present numerous basic cryptographic tools that can be used as building blocks for group-oriented cryptographic applications. These tools include encryption and decryption algorithms, commitment schemes, identification protocols, and digital signature schemes. In many cases, a single building block is not sufficient to solve a complex cryptographic problem. Instead, different basic tools must be combined to achieve the desirable security requirements. In addition, we also demonstrate how to show that an identification protocol is a zero-knowledge proof of knowledge. This proof is an important technique for characterizing properties of cryptographic protocols.

In Chapter 4, we propose a fully public-key traitor-tracing scheme in which each subscriber can choose his or her own private decryption key without others learning the key. The distributor of the digital content utilizes the public data coming from all subscribers to compute a public encryption key. The paid contents are then transmitted to the subscribers, after being encrypted with the public key. Each subscriber can decrypt the data using his or her own secret key. Even if a coalition of subscribers conspire to create a pirate decoder with a tamper-free decryption key, we have a tracing algorithm to trace them. Our scheme is long-lived, which means that the subscribers' secret keys need not be regenerated after the pirate key is detected or

when subscribers join or leave the system. Finally, our scheme guarantees anonymity.

In Chapter 5, we introduce a new type of signature for a group of persons called a group undeniable signature, which satisfies the following requirements: (1) only group members can anonymously sign on behalf of the group; (2) a verifier must interact with the group manager to verify the signature; (3) the group manager can identify the signer of a valid signature. A convertible group undeniable signature scheme allow the group manager to turn select group undeniable signatures into universally verifiable group signatures. They are more suitable than group signatures in applications where signatures are generated for sensitive, nonpublic data. We propose the first convertible group undeniable signature scheme. Furthermore, our scheme is unforgeable, exculpable, unlinkable, and coalition-resistant. The proposed scheme allows the group manager to delegate the ability to confirm and deny signatures to trusted parties. The sizes of the public key and signatures are independent of the group size.

Finally, concluding remarks are given in Chapter 6.

Chapter 2

Foundations

This chapter provides an introduction to the topics of algebra, number theory, and fundamental cryptography. There are numerous books devoted to algebra and number theory [116, 118, 132, 196, 205, 206]. References for computational aspects of algebra number theory are [59, 131]. A reference to complexity theory is the book by Papadimitriou [177]. For fundamental cryptography, we refer the reader to [73, 108, 157, 202, 210, 214].

2.1 Complexity-Theoretic Preliminaries

In this section, we review the concept of randomized algorithms and the definitions of several complexity classes such as P, NP, and BPP.

2.1.1 Randomized Algorithms

Randomized algorithm is important in cryptography. Several algorithms used in encryption and digital signature schemes often involve random choices and therefore are probabilistic. For example, the encryption algorithms in Goldwasser and Micali's scheme [111] and ElGamal's scheme [85] are probabilistic, and the signing algorithms in ElGamal's scheme [85] and the DSA [93] are also probabilistic. For a fixed input, different runs of a randomized algorithm may give different results, and it is inevitable that the analysis of a randomized algorithm involves probabilistic statements. Moreover, when studying the security of cryptographic schemes, adversaries are usually

modelled as randomized algorithms. We give simple definitions of deterministic algorithms and randomized algorithms below.

Definition 2.1.1. *Given an input x, a deterministic algorithm A is a finite sequence of steps or computations that output y in a deterministic way. The output y of a deterministic algorithm A is completely determined by its input x.*

Definition 2.1.2. *Given an input x, a randomized algorithm A may toss a coin a finite number of times during its computation of the output y, and a step may depend on the results of previous coin tosses. The number of coin tosses may depend on the outcome of the previous ones, but it is bounded by some constant t_x for a given input x. The coin tosses are independent and the coin is a fair one, i.e., each side appears with probability $1/2$. Tossing a coin is counted as one step. Often a randomized algorithm is called a probabilistic algorithm if it works for practical purposes but has a theoretical chance of being wrong.*

A formal computation model for a deterministic algorithm is a deterministic Turing machine (DTM) and that for a probabilistic algorithm is a probabilistic Turing machine (PTM). A deterministic Turing machine is a finite state machine having an infinite read-write tape and the state transitions are completely determined by the input. In a probabilistic Turing machine, the state transitions are determined by the input and the output of coin tosses.

Often the coin tosses in a randomized algorithm are considered as internal coin tosses. A second way to look at a randomized algorithm A is to consider the output of the coin tosses as an additional input, which is supplied by an external coin-tossing device. In this view, the model of a randomized algorithm is a deterministic machine. We denote by A_D the corresponding deterministic algorithm. It takes as input A's input x and the outcome r of the coin tosses. When A_D wants to make the next step, it reads the next bit of r and acts accordingly.

Given x, the output $A(x)$ of a randomized algorithm A is a random variable induced by the coin tosses. Let $A(x) = y$ denote the random event "A outputs y on input x." By $Pr[A(x) = y]$, we mean the probability of this event. Assume the number of coin tosses for A is exactly t_x. Then the possible outcomes r of the coin tosses are the binary strings of length t_x, and because the coin tosses are independent,

we have the probability of an outcome r is $1/2^{t_x}$. Hence

$$\Pr[A(x) = y] = \Pr[A_D(x, r) = y] = \frac{|\{r|A_D(x, r) = y\}|}{2^{t_x}}.$$

Probability Notation

Let A be an algorithm. By $A(\cdot)$ we denote that A has one input. By $A(\cdot, \ldots, \cdot)$ we denote that A has several inputs. By $A_{(\cdot)}$ we denote that A is an indexed family of algorithms.

The notation $y \leftarrow A(x)$ denotes that y is obtained by running A on input x. If A is deterministic, then y is unique. If A is probabilistic, then y is a random variable. The notation $x \stackrel{p_S}{\leftarrow} S$, for a set S, means that x is randomly selected from S according to a probability distribution p_S. If the distribution is clear from the context, we simply write $x \leftarrow S$. If p_S is the uniform distribution, we write $x \stackrel{u}{\leftarrow} S$. In this way the members of S are chosen randomly. Let expression $x \in_R S$ denote that x is chosen randomly from set S.

Let B be a boolean function. The notation $(B(y_n) : \{y_i \leftarrow A_i(x_i)\}_{1 \le i \le n})$ denotes the event that $B(y_n)$ is TRUE after the value y_n is obtained by successively running algorithms $A_1, \ldots, A_n$ on inputs $x_1, \ldots, x_n$. The statement

$$\Pr[B(y_n) : \{y_i \leftarrow A_i(x_i)\}_{1 \le i \le n}] = p$$

means that the probability that $B(y_n)$ is TRUE after the value y_n is obtained by running algorithms $A_1, \ldots, A_n$ on inputs $x_1, \ldots, x_n$ is p, where the probability is over the random choices of the probabilistic algorithms involved.

2.1.2 Computational Complexity

The efficiency of an algorithm is measured with respect to the resource required to solve the problem. The resource may include time, storage space, random bits, numbers of processors, etc. Typically, the main focus is time. The running time of an algorithm on a particular input is the number of steps executed, expressed as a function of the input size. The worst-case running time of an algorithm is an upper bound on the running time for any input. The average-case running time of an algorithm is the average running time over all inputs of a fixed size. Often it is

difficult to derive the exact running time of an algorithm. To compare running time of algorithms, the standard asymptotic notation is used.

Definition 2.1.3 (Order Notation).

1. *(Asymptotic upper bound)* $f(n) = O(g(n))$ *if there exits a positive constant* c *and a positive integer* n_0 *such that* $0 \leq f(n) \leq cg(n)$ *for all* $n \geq n_0$.

2. *(Asymptotic lower bound)* $f(n) = \Omega(g(n))$ *if there exists a positive* c *and a positive integer* n_0 *such that* $0 \leq cg(n) \leq f(n)$ *for all* $n \geq n_0$.

3. *(Asymptotic tight bound)* $f(n) = \Theta(g(n))$ *if there exit positive constants* c_1 *and* c_2*, and a positive integer* n_0 *such that* $c_1 g(n) \leq f(n) \leq c_2 g(n)$ *for all* $n \geq n_0$.

4. *(The o-notation)* $f(n) = o(g(n))$ *if for any positive constant* $c > 0$ *there exits a constant* $n_0 > 0$ *such that* $0 \leq f(n) < cg(n)$ *for all* $n \geq n_0$.

Intuitively, $f(n) = O(g(n))$ means that f grows no faster asymptotically than g, and $f(n) = o(g(n))$ means that g is an upper bound for f that is not asymptotically tight. $f(n) = \Omega(g(n))$ means that f grows at least as fast asymptotically as g to within a constant multiple. If both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, then $f(n) = \Theta(g(n))$. The expression $o(1)$ is often used to denote a term $f(n)$ with $\lim_{n\to\infty} f(n)$=0.

Definition 2.1.4. *A polynomial-time algorithm is an algorithm that has a worst-case running time of the form* $O(n^k)$*, where* n *is the input size and* k *is a constant. Any algorithm whose running time cannot be so bounded is called an exponential-time algorithm.*

Definition 2.1.5. *A probabilistic polynomial-time algorithm is a probabilistic algorithm that has a running time of the form* $O(n^k)$*, where* n *is the input size and* k *is a constant. The running time of a probabilistic algorithm is measured as the number of steps in the model of algorithms, i.e., the number of steps of the probabilistic Turing machine. Tossing a coin is one step in this model.*

Let A be a probabilistic algorithm. The worst-case running time $time_A(x)$ of A on input x is the maximum number of steps that A needs to generate the output

$A(x)$. The expected running time $etime_A(x)$ of A on input x is the average number of steps that A needs to generate the output $A(x)$, i.e.,

$$etime_A(x) = \sum_{t=1}^{\infty} t \cdot Pr[time_A(x) = t].$$

Let $\mathcal{P}$ be a computational problem and A be a probabilistic algorithm for $\mathcal{P}$. The worst-case running time of A for $\mathcal{P}$ is

$$t_A = \max\{time_A(x) : \text{ for all instances } x \text{ of } \mathcal{P}\}.$$

The expected running time of A for $\mathcal{P}$ is

$$et_A = \max\{etime_A(x) : \text{ for all instances } x \text{ of } \mathcal{P}\}.$$

Definition 2.1.6 (Monte Carlo Algorithms/Las Vegas Algorithms). *Let $\mathcal{P}$ be a computational problem.*

1. *A Monte Carlo algorithm A for $\mathcal{P}$ is a probabilistic algorithm A, whose running time $time_A(x)$ for all instance x of $\mathcal{P}$ is bounded by a polynomial $Q(|x|)$ and which yields a correct answer to $\mathcal{P}$ with a probability of at least $2/3$.*

2. *A Las Vegas algorithm A for $\mathcal{P}$ is a probabilistic algorithm, whose running time $etime_A(x)$ for all instance x of $\mathcal{P}$ is bounded by a polynomial $Q(|x|)$ and which always yields a correct answer to $\mathcal{P}$.*

Definition 2.1.7. *A subexponential-time algorithm is an algorithm that has a worst-case running time of the form $O(e^{(b+o(1)n^a(\ln(n))^{1-a}})$, where n is the input size, b is a positive constant, and a is a constant satisfying $0 < a < 1$.*

A subexponential-time algorithm is slower than a polynomial-time algorithm yet faster than an algorithm whose running time is exponential in the input size. Observe that for $a = 0$ the running time $O(e^{(b+o(1)n^a(\ln(n))^{1-a}})$ is polynomial $O(n^a)$, while for $a = 1$ the running time $O(e^{(b+o(1)n^a(\ln(n))^{1-a}})$ is exponential $O(e^{bn})$

For simplicity, computational problems are often modelled as decision problems: decide whether a given $x \in \{0,1\}^*$ belongs to a language $L \subseteq \{0,1\}^*$. Computational problems are classified by the most efficient known algorithm for solving them.

Definition 2.1.8 (P). *The complexity class* P *is the set of decision problems that can be solved by deterministic polynomial-time algorithms.*

Definition 2.1.9 (NP). *The complexity class* NP *is the set of decision problems for which a YES answer can be verified by polynomial-time deterministic algorithms given some extra information, called a witness.*

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. We say that R is polynomially bounded if there exists a polynomial Q such that $|w| \leq Q(|x|)$ holds for all (x,w) in R. Furthermore, R is an NP-relation if it is polynomially bounded and if there exists a polynomial-time algorithm for deciding membership of pairs (x,w) in R. Let $L_R = \{x | \exists w \text{ such that } (x,w) \in R\}$ be the language defined by R. A language L is in NP if there exists an NP-relation $R_L \subseteq \{0,1\}^* \times \{0,1\}^*$ such that $x \in L$ if and only if there exists a w such that $(x,w) \in R_L$. Such a w is called a witness of the membership of x in L. The set of all witnesses of x is denoted by $R_L(x)$.

Definition 2.1.10 (Bounded-Probability Polynomial-Time, BPP). *Let L be the language of some decision problem. We say that L is recognized by the probabilistic polynomial-time algorithm A if for every $x \in L, \Pr[A(x) = 1] \geq 2/3$ and for every $x \notin L, \Pr[A(x) = 0] \geq 2/3$.* BPP *is the class of languages that can be recognized by a probabilistic polynomial-time algorithm.*

Problems in P are considered easy, and problems not in P are considered hard. It is widely believed that the class P is strictly smaller than the class NP. Whether NP=P is the most important open problem in complexity theory.

We will consider as efficient only randomized algorithm whose running time is bounded by a polynomial in the length of the input. A problem is called intractable (or computationally infeasible) if no probabilistic polynomial-time algorithm could solve it, whereas one that can be solved using a probabilistic polynomial-time algorithm is called tractable (or computationally feasible).

All the above complexity classes are defined in terms of worst-case complexity. However, in cryptography, average-case complexity of a problem is a more significant measure than its worst-case complexity. This is because a cryptosystem must be unbreakable in most cases, which implies that it will be intractable to break the cryptosystem on the average. Hence, a necessary condition for a secure cryptographic

scheme is that the corresponding cryptanalysis problem must be intractable on the average.

2.2 Algebra and Number Theory

Algebra and number theory play an important role in contemporary cryptography. Most public-key cryptosystems and secure protocols are based on problems from number theory. In this section, we give several well-known results on algebra and number theory. Most of the proofs are omitted because they can be found in most textbooks on algebra and number theory.

2.2.1 Integer Arithmetic

Let $\mathbb{Z}$ denote the set of integers $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$ and $\mathbb{N} = \{n \in \mathbb{Z} | n > 0\}$ denote the set of natural integers. Let $a, b \in \mathbb{Z}, a \neq 0$. We can divide b by a with a remainder r. Of particular importance for divisibility is the following algorithm.

Theorem 2.2.1 (The Division Algorithm). *If $a \in \mathbb{N}$ and $b \in \mathbb{Z}$, then there exist unique integers $q, r \in \mathbb{Z}$ with $0 \leq r < a$, and $b = aq + r$. The number q is called the quotient and r is called the remainder of the division.*

The number r is also called the remainder of b modulo a. We write $b \bmod a$ for r. The number q is the integer quotient of b and a. We write $b \operatorname{div} a$ for q. An integer a divides an integer b (equivalently, a is a divisor of b, or a is a factor of b) if there is some $c \in \mathbb{Z}$, with $b = ac$. We write $a|b$ for "a divides b." A nonnegative integer d is the greatest common divisor of a and b if (1) $d|a$ and $d|b$; (2) If $t \in \mathbb{Z}$ divides both a and b, then t divides d. The greatest common divisor is denoted by $\gcd(a, b)$. If $\gcd(a, b) = 1$, then a is called relatively prime to b, or prime to b for short. An integer $p \geq 2$ is said to be prime if its only positive divisors are 1 and p. Otherwise, p is composite. Primes plays an important role in modern cryptography. Fortunately, there are very fast probabilistic primality tests [165, 190, 209] (with a high probability) for finding the correct answer to the question whether a given number is prime or not. More recently, Agrawal, Kayal, and Saxena present the first

deterministic, $\tilde{O}(\log_2(n)^{12})$ time algorithm for testing if a number n is prime [1]. The notation $\tilde{O}(t(n))$ denotes $O(t(n)\text{poly}(\log(t(n))))$, where $t(n)$ is some function of n.

In addition, numbers can be factored into products of primes.

Theorem 2.2.2 (Fundamental Theorem of Arithmetic). *Let $n \in \mathbb{N}, n \geq 2$. There exist pairwise distinct primes $p_1, \ldots, p_k$ and exponents $e_1, \ldots, e_k \in \mathbb{N}, e_i \geq 1, i = 1, \ldots k$, such that*

$$n = \prod_{i=1}^{k} p_i^{e_i}.$$

The primes $p_1, \ldots, p_k$ and exponents $e_1, \ldots, e_k$ are unique.

Efficient algorithms exist for the addition, subtraction, multiplication, and division of numbers. Let $a, b, m \in \mathbb{N}, a, b \leq m, k = \lfloor \log_2(m) \rfloor + 1$. Note that k is the binary length of m. The binary length of m is usually denoted by $|m|$, and we only use the notation if it cannot be confused with the absolute value. By the classic grade school method, the numbers of bit operations for the computations of $a + b$ and $a - b$ are $O(k)$, whereas for $a \cdot b$ and a div b are $O(k^2)$. Multiplication can be improved to $O(k \log_2(k) \log_2 \log_2(k))$ if a fast multiplication algorithm is used [2]. It is easy to multiply two numbers, but we do not have a practical algorithm for factoring extremely large numbers so far.

In the following, we introduce the Euclidean algorithm and the extended Euclidean algorithm. The Euclidean algorithm efficiently computes $\gcd(a, b)$. It can be extended such that not only $\gcd(a, b)$ but also the coefficients d and e of the linear combination $\gcd(a, b) = da + eb$ are computed. Moreover, the extended Euclidean algorithm can be used to compute the inverse of an element in a multiplicative group.

Theorem 2.2.3 (The Euclidean Algorithm). *Let $a, b \in \mathbb{Z}$ $(b \geq a > 0)$, and set $b = r_{-1}$ and $a = r_0$. By repeatedly applying the division algorithm, we obtain $r_{j-1} = r_j q_{j+1} + r_{j+1}$ with $0 < r_{j+1} < r_j$ for all $0 \leq j < n$, where n is the least nonnegative number such that $r_{n+1} = 0$, in which case $\gcd(a, b) = r_n$.*

Theorem 2.2.4 (The Extended Euclidean Algorithm). *Let $a, b \in \mathbb{N}$, and let q_i for $i = 1, 2, \ldots, n + 1$ be the quotients obtained from the application of the Euclidean algorithm to find $g = \gcd(a, b)$, where n is the least nonnegative integer such that*

$r_{n+1} = 0$. *If* $s_{-1} = 1, s_0 = 0$, *and*

$$s_i = s_{i-2} - q_{n-i+2}s_{i-1},$$

for $i = 1, 2, \ldots, n+1$, *then*

$$g = s_{n+1}b + s_n a.$$

Suppose $a, b \leq m, k = \lfloor \log_2(m) \rfloor + 1$. The Euclidean algorithm has a running time of $O(k^2)$ bit operations, as does the extended Euclidean algorithm.

2.2.2 Basic Algebra

Let S be a nonempty set. An binary operation $*$ defined on S is a mapping from $S \times S$ to S. Let $a * b$ denote the result of $*$ applied to the elements $a, b \in S$. The operation $*$ is associative if $a * (b * c) = (a * b) * c$ holds for all $a, b, c \in S$, and commutative if $a * b = b * a$ holds for all $a, b \in S$. An element e in S is called an identity element if e satisfies $e * a = a * e = a$ for al $a \in S$. An inverse of an element $a \in S$ is an element $b \in S$ such that $a * b = b * a = e$.

Group

Definition 2.2.1 (Group). *Let G be a non-empty set and $*$ an operation defined on G. Then the pair $(G, *)$ is called a group if*

1. *The operation $*$ is associative.*
2. *G contains an identity element, say e.*
3. *Every element in G has an inverse under $*$.*

A group $(G, *)$ is called abelian or commutative if the operation $*$ is commutative. We speak of a finite group $(G, *)$ of order n if G is finite sets of cardinality n. The order of $(G, *)$ is denoted by $|G|$ or $\mathrm{ord}(G)$.

Often $(G, *)$ is denoted simply by G. It can easily be seen that the identity element is unique, so is the inverse of any element. If the operation is called addition, the identity element is denoted as 0 and the inverse element of a as $-a$. If the operation

is multiplicative, the identity element is denoted as 1 and the inverse of an element a as $1/a$ or a^{-1}. Unless stated otherwise, we use the multiplicative notation when dealing with arbitrary groups. So a^k mean that a is multiplied k-times by itself, and a^{-k} denotes $(1/a)^k$. The following are typical groups.

1. $(\mathbb{Z}, +) : \mathbb{Z}$ is the set of all integers and $+$ is the regular additive operation. The identity is 0, and the inverse of a is $-a$.

2. $(\mathbb{Z}_n, +) : \mathbb{Z}_n = \{0, 1, \ldots n-1\}$ and $+$ is the congruent additive operation (modulo n). The identity is 0, and the inverse of a is $n - a$.

3. $(\mathbb{Z}_n^*, *) : \mathbb{Z}_n^* = \{a | a \in \mathbb{Z}_n, \gcd(a, n) = 1\}$ and $*$ is the congruent multiplicative operation (modulo n). The identity is 1, and the inverse of a can be computed by the extended Euclidean algorithm.

Definition 2.2.2 (Cyclic Group). *A group G is cyclic if there is $g \in G$ such that every element $a \in G$ can be written in the form g^k for some $k \in \mathbb{Z}$. That is, $G = \{g^i | i \geq 0\}$. We call such g a generator of G and write $\langle g \rangle = G$ to indicate that g generates G.*

Definition 2.2.3. *Let G be a group and $a \in G$. The order of a, denoted by* $\mathrm{ord}(a)$*, is the smallest positive integer n such that $a^n = 1$, provided that such an integer exits. If such an n does not exit, then the order of a is defined to be ∞.*

We remark that if G is finite, there are exponents $n \in \mathbb{N}$, with $a^n = 1$.

Definition 2.2.4 (Subgroup). *Let $(G, *)$ be a group. We say that $(H, *)$ is a subgroup of G if $H \subseteq G$ and $(H, *)$ is a group.*

Fact 2.2.5. *Let G be a group. For any $a \in G$, $\langle a \rangle = \{g^i | i \geq 0\}$ is a subgroup of G.*

Fact 2.2.6. *Let G be a finite group and 1 is the identity element of G. Then $a^{|G|} = 1$ for all $a \in G$.*

Fact 2.2.7 (Lagrange's Theorem). *If H is a subgroup of a finite group G, then $|H|$ divides $|G|$.*

Fact 2.2.8. *Let G be a finite group and $a \in G$. Let $n \in \mathbb{N}$ with $a^n = 1$. Then* $\mathrm{ord}(a)$ *divides n. In particular, the order of any element of a finite group divides the order of the group.*

Fact 2.2.9. *Let G be cyclic and $|G| = n$. Then g is a generator of G if and only if $g^{n/p} \neq 1$ for every prime factor p of n.*

Fact 2.2.10. *Let G be a finite cyclic group and g be a generator of G. Then*

1. *The element $b = g^i$ has order $|G|/\gcd(|G|, i)$. In particular, b is a generator of G if and only if $\gcd(|G|, i) = 1$. Hence, if $|G|$ is prime, then every element different from 1 is a generator of G.*

2. *Suppose $d||G|$. The G has exactly $\phi(d)$ elements of order d. In particular, G has $\phi(|G|)$ generators.*

3. *For all divisors q of $|G|$, let G_q be the subgroup of G generated by $g^{|G|/q}$. Then the groups G_q are all the subgroups of G. In particular, every subgroup of G is cyclic, and for each divisor q of $|G|$ there is a unique subgroup of G of order q, namely G_q.*

Ring and Field

We now consider a situation that two operations are defined on a set. The first will be denoted by $a + b$, the second by $a * b$.

Definition 2.2.5 (Ring). *The triple $(R, +, *)$ is called a ring, if*

1. *$(R, +)$ is a abelian group.*

2. *The operation $*$ is associative.*

3. *Distributivity holds, i.e., for all $a, b, c \in R$, $a * (b + c) = a * b + a * c$ and $(b + c) * a = b * a + c * a$.*

If the operation $*$ is commutative on R, then the ring $(R, +, *)$ is called commutative. The following are typical rings.

1. $(\mathbb{Z}, +, *)$: $\mathbb{Z}$ is the set of all integers. The operations $+$ and $*$ are regular addition and multiplication, respectively.

2. $(\mathbb{Z}_n, +, *) : \mathbb{Z}_n = \{0, 1, \ldots n-1\}$. The operations $+$ and $*$ are under modular n.

Definition 2.2.6 (Field). *A triple $(F, +, *)$ is called a field, if*

1. *$(F, +)$ is a abelian group. Its identity element is denoted by 0.*

2. *$(F - \{0\}, *)$ is a group. The multiplicative identity element is denoted by 1.*

3. *Distributivity holds.*

A commutative field is a field for which $(F - \{0\}, *)$ is commutative. Every finite field is commutative. The following are typical fields.

1. $(\mathbb{Q}, +, *) : \mathbb{Q}$ is the set of rational numbers.

2. $(\mathbb{Z}_p, +, *) : \mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ and $+$ and $*$ are computed under modular p, where p is prime.

2.2.3 Modular Arithmetic

Let $a, b \in \mathbb{Z}$. Then a is said to be congruent to b modulo n, denoted by $a \equiv b \pmod{n}$ if n divides $(a - b)$. The integer n is called the modulus of the congruence. The integer modulo n, denoted $\mathbb{Z}_n$ is the set of integers $\{0, 1, \ldots, n-1\}$. Addition, subtraction, and multiplication in $\mathbb{Z}_n$ are performed modulo n. The multiplicative inverse of a modulo n is an integer $x \in \mathbb{Z}_n$ such that $ax \equiv 1 \pmod{n}$. If such an x exists, then it is unique, and a is said to be invertible, or a unit. The multiplicative inverse of a is denoted by a^{-1}. Let $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | \gcd(a, n) = 1\}$. If n is prime, then $\mathbb{Z}_n^* = \{a | 1 \leq a \leq n-1\}$. Note that $(\mathbb{Z}_n, +)$ and $(\mathbb{Z}_n^*, *)$ are abelian groups.

In the following, we first present important definitions and facts about modular arithmetic. Then we describe several tractable problems that can be solved in polynomial time.

Definitions and Facts

Definition 2.2.7. *Let n be a positive integer. The Euler phi function or the Euler totient function $\phi(n)$ is defined to be the number of positive integers not exceeding n which are relatively prime to n.*

Definition 2.2.8. *Let a and n be relatively prime positive integers. Then, the least positive integer x such that $a^x \equiv 1 \pmod{n}$ is called the order of a modulo n, denoted by $\mathrm{ord}_n a$.*

Definition 2.2.9. *If g and n are relatively prime integers with $n > 0$ and if $\mathrm{ord}_n g = \phi(n)$, then g is called a primitive root modulo n.*

Definition 2.2.10. *Let $n \in \mathbb{N}$ and $a \in \mathbb{Z}$. We say that a is a quadratic residue modulo n if $a \not\equiv 0 \pmod{n}$ and the congruence $x^2 \equiv a \pmod{n}$ has a solution $x \in \mathbb{Z}$. If $a \not\equiv 0 \pmod{n}$ and the congruence $x^2 \equiv a \pmod{n}$ has no solution, we say that a is a quadratic nonresidue modulo n.*

In most cases, we are only interested in the quadratic residues a which are relatively to the modulus n.

Definition 2.2.11. *$QR_n = \{a \in \mathbb{Z}_n^* | a$ is a quadratic residue modulo $n\}$. On the contrary, $QNR_n = \{a \in \mathbb{Z}_n^* | a$ is a quadratic nonresidue modulo $n\}$*

Note that for any n, QR_n is a subgroup of $\mathbb{Z}_n^*$, while QNR_n is not a subgroup of $\mathbb{Z}_n^*$ because at least $1 \notin \mathrm{QNR}_n$.

Definition 2.2.12. *Let p be a prime > 2, and let $a \in \mathbb{Z}$ be prime to p.*

$$\left(\frac{a}{p}\right) = \begin{cases} +1 \text{ if } a \in QR_p, \\ -1 \text{ if } a \in QNR_p, \end{cases}$$

is called the Legendre symbol of $a \bmod p$. For $a \in \mathbb{Z}$ with $p|a$, we set $(\frac{a}{p}) = 0$.

Definition 2.2.13. *Let $n \in \mathbb{Z}$ be a positive odd number and $n = \prod_{i=1}^{r} p_i^{e_i}$ be the decomposition of n into primes. Let $a \in \mathbb{Z}$. Then*

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{r} \left(\frac{a}{p_i}\right)^{e_i}$$

is called the Jacobi symbol of $a \bmod n$.

Definition 2.2.14. *A universal exponent of the positive integer n is a positive integer U such that*

$$a^U \equiv 1 \pmod{n}$$

for all integers a relatively prime to n.

Definition 2.2.15. *The least universal exponent of the positive integer n is called the minimal universal exponent of n, denoted by $\lambda(n)$.*

Fact 2.2.11 (Chinese Remainder Theorem). *Let $m_1, m_2, \ldots, m_n$ be positive integers that are relatively prime in pairs. Then for any given integers $b_1, b_2, \ldots, b_n$, the system of congruences*

$$x \equiv b_i \pmod{m_i}, 1 \leq i \leq n$$

has a unique solution modulo $M = m_1 m_2 \cdots m_n$. The solution is given by $x = \sum_{i=1}^{n} b_i M_i y_i \bmod M$, where $M_i = M/m_i$ and $M_i y_i \equiv 1 \pmod{m_i}$.

Assume k is the length of every modulus. The computational complexity of applying the Chinese remainder theorem is $O_B(M(kn)\log(n)) + O_B(nM(k)\log(k))$, where $M(x)$ is the time of multiplying two x bit integers, and O_B indicates order of magnitude in bit operations [2, Theorem 8.21].

Fact 2.2.12. *Simultaneous congruences*

$$x \equiv b_i \pmod{m_i}, 1 \leq i \leq n$$

have a solution if and only if $\gcd(m_i, m_j)$ divides $b_i - b_j$ for all pairs of integers (i, j) with $1 \leq i < j \leq n$, in which case the solution is unique modulo $M = \operatorname{lcm}(m_1, m_2, \ldots, m_n)$ and is given by the Chinese remainder theorem with the said M.

Fact 2.2.13. *Let $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ be the prime-power factorization of the positive integer n. Then*

$$\phi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \cdots (1 - \frac{1}{p_k}).$$

Furthermore,

$$\phi(n) > \frac{n}{e^\gamma \log_{10}(\log_{10}(n)) + \frac{2.6}{\log_{10}(\log_{10}(n))}} \text{ (see [116, 197])},$$

where Euler's constant $\gamma = 0.5772\ldots$. *This inequality implies, for example, that*

$$\phi(n) > \frac{n}{6\log_{10}(\log_{10}(n))}, \text{ for } n \geq 1.3 \times 10^6.$$

Fact 2.2.14 (Fermat's Little Theorem). *If* p *is a prime and* p *does not divide* $a \in \mathbb{Z}$, *then*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Fact 2.2.15 (Euler's Theorem). *Let* $n \in \mathbb{N}$ *and* $a \in \mathbb{Z}$. *If* $\gcd(a, n) = 1$, *then*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

Fermat's little theorem and Euler's theorem are special cases of Fact 2.2.6.

Fact 2.2.16. *Suppose that* g *and* n *are relatively prime with* $n > 0$. *Then* $g^i \equiv g^j \pmod{n}$ *if and only if* $i \equiv j \pmod{\text{ord}_n g}$.

Fact 2.2.17. *The positive integer* n *possesses a primitive root if and only if*

$$n = 2, 4, p^t, \text{ or } 2p^t,$$

where p *is an odd prime and* t *is a positive integer.*

Fact 2.2.18. *If the positive integer* n *has a primitive root, then it has a total of* $\phi(\phi(n))$ *incongruent primitive roots.*

Fact 2.2.19. *Let* p *be a prime. Then* $\mathbb{Z}_p^*$ *is cyclic, and the number of generators is* $\phi(p-1)$.

Fact 2.2.20. *Let* p *be a prime. Then* $x \in \mathbb{Z}_p^*$ *is a primitive root if and only if* $x^{(p-1)/q} \not\equiv 1 \pmod{p}$ *for every prime* $q|p-1$.

Fact 2.2.21. *Let* p *be a prime* > 2 *and* $g \in \mathbb{Z}_p^*$ *be a primitive root of* $\mathbb{Z}_p^*$. *Let* $a \in \mathbb{Z}_p^*$. *Then* $a \in QR_p$ *if and only if* $a \equiv g^t \pmod{p}$ *for some even number* $t, 0 \leq t \leq (p-2)$. *Furthermore, exactly half of the elements of* $\mathbb{Z}_p^*$ *are quadratic residues, i.e.,* $|QR_p| = |QNR_p| = (p-1)/2$.

Fact 2.2.22 (Euler's Criterion). *Let* p *be a prime* > 2 *and* $a \in \mathbb{Z}$. *Then*

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \bmod p.$$

Euler's criterion can be generalized to the kth residuosity as follows.

Fact 2.2.23. *A number a is an eth residue of $\mathbb{Z}_p^*$ if and only if $a^{(p-1)/d} \equiv 1 \pmod{p}$, where $d = \gcd(e, p-1)$.*

Fact 2.2.24 ([6, 98]). *Let $n = p_1 p_2$, where $p_1 \neq p_2, p_1 = 2q_1 + 1, p_2 = 2q_2 + 1$, and p_1, p_2, q_1, q_2 are all prime numbers. Then the following holds.*

1. *The order of $g_0 \in \mathbb{Z}_n^*$ is equal to $q_1 q_2$ or $2q_1 q_2$ if and only if $\gcd(g_0 + 1, n) = 1$ and $\gcd(g_0 - 1, n) = 1$.*
2. *For any g_0 such that $\gcd(g_0 + 1, n) = 1$ and $\gcd(g_0 - 1, n) = 1$, $\langle g_0^2 \rangle \subset \mathbb{Z}_n^*$ is a cyclic subgroup of order $q_1 q_2$.*

Fact 2.2.25. *Let M be a positive integer with odd prime factorization $M = p_1 p_2 \cdots p_n$. Then the following holds.*

1. *$\lambda(M) = \mathrm{lcm}(\phi(p_1), \phi(p_2), \ldots, \phi(p_n))$.*
2. *There exists an integer g such that $\mathrm{ord}_M g = \lambda(M)$, the largest possible order of an integer modulo M.*
3. *Let r_i be a primitive root modulo p_i. The solution of simultaneous congruences $x \equiv r_i \pmod{p_i}$, $i = 1, 2, \ldots, n$, produces such an integer g.*

The above fact implies that if M is a product of large primes $p_i = 2q_i + 1$ where q_i are also primes, then there exists a g whose order contains large prime factors. The reason is that

$$\lambda(M) = \mathrm{lcm}(p_1 - 1, p_2 - 1, \ldots, p_n - 1) = 2q_1 q_2 \cdots q_n. \tag{2.1}$$

Tractable Problems in $\mathbb{Z}_n$

As in $\mathbb{Z}$, basic group operations in $\mathbb{Z}_n$ can be performed efficiently, i.e., in time polynomial in the group size. Let n be an integer and $k = \lfloor \log_2(n) \rfloor + 1$. By the ordinary method, the numbers of bit operations for the computations of $(a+b) \bmod n$ and $(a-b) \bmod n$ are $O(k)$, whereas the number of bit operations for $(a \cdot b) \bmod n$ is $O(k^2)$. Modular inversion $a^{-1} \bmod n$ can be computed in $O(k^2)$ using the extended

Euclidean algorithm. Modular exponentiation $a^t \bmod n$ can be performed with at most $2 \cdot |t|$ modular multiplications using the repeated squaring method. Hence, an exponentiation in $\mathbb{Z}_n^*$ can be computed in $O(k^3)$.

There are hardware implementations for performing modular multiplication fast. In particular, Norris and Simmons use the concept of delayed-carry adders to produce a hardware modular multiplier which computes the product of two t-bit operands modulo a t-bit modulus in $2t$ clock cycles [172]. Brickell improves the idea to produce a modular multiplier requiring only $t+7$ clock cycles [32]. Enhancements of Brickell's method are given by Walter [218]. Koç provides a comprehensive survey of hardware methods for modular multiplication [133].

We now present several number-theoretic problems that can be solved in probabilistic polynomial time. Let p is prime. First, consider computations in $\mathbb{Z}_p$. The following problems is tractable.

1. Finding a primitive root modulo p when the prime factors of $p-1$ are known.

   By Fact 2.2.20, we can use the following algorithm to obtain a primitive root modulo p.

   Algorithm 2.2.1.

   input: prime p

   output: a primitive root modulo p

   1. Randomly choose an integer g, with $0 < g < p-1$
   2. if $g^{(p-1)/q} \not\equiv 1 \pmod{q}$ for all primes q dividing $p-1$.
   3. then output g
   4. else go to 1

   Because
   $$\phi(p-1) > \frac{(p-1)}{6\log_{10}(\log_{10}(p-1))},$$
   the expected iteration to find a primitive root is $O(\log_{10}(\log_{10}(p)))$. Note that if the prime factors of $p-1$ are not known, no efficient algorithms are known for the generation of primitive roots.

2. Testing whether an element is a quadratic residue modulo p.

   The problem is called the quadratic residuosity problem and can be solved by Euler's criterion.

3. Testing whether an element is an eth residue modulo p.
   The problem is called eth residuosity problem and can be solved by Fact 2.2.23.

4. Computing the eth root modulo p when $\gcd(e, p-1) = 1$.
   The problem is called eth root problem. When $\gcd(e, p-1) = 1$, $a^{e^{-1}} \bmod p$ is a e'th root of a modulo p, where $e^{-1}e \equiv 1 \pmod{p-1}$.

5. Computing square roots of a quadratic residue in $\mathbb{Z}_p$.
   Because $\gcd(2, p-1) \neq 1$, computing the square root is not easy. We do not know any polynomial-time algorithm that computes $a^{1/2} \bmod p$ deterministically. Nevertheless, there exits a probabilistic polynomial-time algorithm that does it. We describe the algorithm as follows.

   Algorithm 2.2.2.

   input: (a, p), where $a \in \mathrm{QR}_p$ and p is an odd prime
   output: $a^{1/2} \bmod p$

   1. if $p \equiv 3 \bmod 4$
   2. then output $a^{(p+1) \text{ div } 4} \bmod p$
   3. else
   4. randomly choose $b \in \mathrm{QNR}_p$
   5. $i \leftarrow (p-1) \text{ div } 2; j \leftarrow 0$
   6. repeat
   7. $i \leftarrow i \text{ div } 2; j \leftarrow j \text{ div } 2$
   8. if $a^i b^j \equiv -1 \bmod p$
   9. then $j \leftarrow j + (p-1) \text{ div } 2$
   10. until $i \equiv 1 \pmod 2$
   11. output $a^{(i+1) \text{ div } 2} b^{j \text{ div } 2} \bmod p$

   Suppose $p \equiv 1 \bmod 4$. Let $(p-1)/2 = 2^\ell r$, with r odd and $\ell \geq 1$. The above algorithm randomly chooses a quadratic nonresidue $b \in \mathrm{QNR}_p$ and then finds an exponent s such that $a^r b^{2s} = 1$. Therefore, $a^{r+1} b^{2s} = a$ and $a^{(r+1)/2} b^s$ is a square root of a. To get a quadratic nonresidue we can randomly choose an element b of $\mathbb{Z}_p^*$ and test by Euler's criterion whether b is a nonresidue. Because half of the elements in $\mathbb{Z}_p^*$ are nonresidues, we expect to get a nonresidue after two random choices. Moreover s is obtained in ℓ steps.

6. Determining the order of a group element when the prime factorization of the group order is known.

   By Fact 2.2.8, we can use the following algorithm to find the order of a group element a modulo p efficiently.

   Algorithm 2.2.3.

   input: prime p, an element $a \in \mathbb{Z}_p^*$, and the prime factorization
   $$p - 1 = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$$
   output: the order of a

   1. $t \leftarrow p - 1$
   2. for $i = 1$ to k do
   3. while $a^{t/p_i} \equiv 1 \pmod{p}$ do
   4. $t \leftarrow t/p_i$
   5. output t

Let n be a composite number. Consider computations in $\mathbb{Z}_n$. The following problems are believed to be hard if the factorization of n is unknown but become tractable if the opposite is true.

1. Testing if an element is a quadratic residue in $\mathbb{Z}_n$.

2. Computing the square root of a quadratic residue in $\mathbb{Z}_n$.

   This is provably as hard as factoring n. Assume $n = pq$. When the factorization of $n = pq$ is known, we compute the square root of $a \in \mathbb{Z}_n^*$ by first computing the square root in $\mathbb{Z}_p$ of a mod p and the square root in $\mathbb{Z}_q$ of a mod q and then using the Chinese remainder theorem to obtain roots of x in $\mathbb{Z}_n$.

3. Computing eth roots modulo n when $\gcd(e, \phi(n)) = 1$.

   Suppose $n = pq$. Then the problem is the so-called RSA problem. $\phi(n)$ can be found by factoring n. Thus, if factoring is easy, then so is the RSA problem.

2.2.4 Intractable Problems

The Discrete Logarithm Assumption

Let G be a finite cyclic group and g be a generator of G. The discrete logarithm of some element $x \in G$, denoted $\log_g(x)$, is the unique integer $a, 0 \leq a < |G|$, such that

$x = g^a$. The discrete logarithm (DL) problem is the following: Given G, g, and an element $x \in G$, find the integer $a, 0 \leq a < |G|$, such that $x = g^a$. It is unknown whether an efficient algorithm for the DL problem exits. All known algorithms have exponential or subexponential running time, and it is widely believed that the problem is intractable. We state the assumption more precisely as follows.

Definition 2.2.16. *Let $I = \{(G, g)|G$ is a cyclic group, $g \in G$ a generator $\}$ and $I_k = \{(G, g) \in I|\text{ord}(G) = k\}$. For every probabilistic polynomial-time algorithm A, every positive polynomial Q, and all sufficiently large k,*

$$Pr[y = \log_g(x) : (G, g) \xleftarrow{u} I_k, x \xleftarrow{u} G, y \leftarrow A(G, g, x)] < \frac{1}{Q(k)}.$$

This is called the discrete logarithm assumption.

The Diffie-Hellman and Decision Diffie-Hellman Assumptions

The Diffie-Hellman (DH) problem is the following: Given a finite cyclic group G, a generator g of G, and the two elements g^a and g^b, find the element g^{ab}. It is believed that the DH problem is intractable. We make this precise in the following.

Definition 2.2.17. *Let $I = \{(G, g)|G$ is a cyclic group, $g \in G$ a generator $\}$ and $I_k = \{(G, g) \in I|\text{ord}(G) = k\}$. For every probabilistic polynomial-time algorithm A, every positive polynomial Q, and all sufficiently large k,*

$$Pr[y = g^{ab} : (G, g) \xleftarrow{u} I_k, a \xleftarrow{u} \mathbb{Z}_{|G|}, b \xleftarrow{u} \mathbb{Z}_{|G|}, y \leftarrow A(G, g, g^a, g^b)] < \frac{1}{Q(k)}.$$

This is called the Diffie-Hellman assumption.

Obviously, if the DL problem can be solved in polynomial time, then the DH problem can be solved in polynomial time. For some groups, the DH and the DL problems have been proved to be computationally equivalent. [18, 153, 154, 155].

The decision Diffie-Hellman (DDH) problem is the following: Given a finite cyclic group G, a generator g of G, and the three elements g^a, g^b, and g^c, decide whether the elements g^c and g^{ab} are equal. It is believed that the problem is intractable. This is made precise in the following.

Definition 2.2.18. *Let* $I = \{(G, g)|G$ *is a cyclic group,* $g \in G$ *a generator* $\}$ *and* $I_k = \{(G, g) \in I|\text{ord}(G) = k\}$. *For every probabilistic polynomial-time algorithm* A*, every positive polynomial* Q*, and all sufficiently large* k*,*

$$Pr[c' = c : (G, g) \xleftarrow{u} I_k, a \xleftarrow{u} \mathbb{Z}_{|G|}, b \xleftarrow{u} \mathbb{Z}_{|G|}, z_0 = ab \bmod |G|, z_1 \xleftarrow{u} \mathbb{Z}_{|G|}, c \xleftarrow{u} \{0, 1\},$$
$$c' \leftarrow A(G, g, g^a, g^b, g^{z_c})] < \frac{1}{2} + \frac{1}{Q(k)}.$$

This is called the decision Diffie-Hellman assumption.

Clearly, an efficient algorithm to solve the DH problem implies one for the DDH problem.

Shoup shows that any generic algorithm must perform $\Omega(p^{1/2})$ group operations for the two problems, where p stands for the largest prime divisor of the group order in the case of the DH problem, and for the smallest prime divisor of the group order in the case of the DDH problem [208]. A generic algorithm does not exploit any special properties of the encodings of group elements except that each group element is encoded as a unique bit string.

The Representation Assumption

Let G be a group and $g_1, \ldots, g_r \in G$ be pairwise distinct generators of G. A representation of some element $y \in G$ is an r-tuple $(a_1, \ldots, a_r), 0 \leq a_i \leq |G| - 1$ for all $1 \leq i \leq r$ such that

$$y = \prod_{i=1}^{r} g_i^{a_i}.$$

The representation problem is the following: Given $G, g_1, \ldots, g_r$, and an element $y \in G$, find integers $a_1, \ldots, a_r, 0 \leq a_i \leq |G| - 1$, such that

$$y = \prod_{i=1}^{r} g_i^{a_i}.$$

The representation problem is a generalization of the DL problem. It is believed that the representation problem is intractable. We make this statement precise in the following.

Definition 2.2.19. *Let* $I = \{(G, g_1, \ldots, g_r)|G$ *is a cyclic group,* $g_1, \ldots, g_r \in G$ *generators*$\}$. *Let* $I_k = \{(G, g_1, \ldots, g_r) \in I|\text{ord}(G) = k\}$. *For every probabilistic*

polynomial-time algorithm A, every positive polynomial Q, and all sufficiently large k,

$$Pr[g_1^{a_1} \cdots g_r^{a_r} = y : (G, g_0, \ldots, g_r) \xleftarrow{u} I_k, y \xleftarrow{u} G, \\ (a_1, \ldots, a_r) \leftarrow A(G, g_1, \ldots, g_r, y)] < \frac{1}{Q(k)}.$$

This is called the representation assumption.

If the generators $g_1, \ldots, g_r$ are all chosen randomly, finding two different representations of an element is as hard as the DL problem. Brands gives a thorough discussion on the representation problem in [26].

The Assumption of Equality of Discrete Logarithms

Let G be a group and $g_0, g_1 \in G$ be distinct generators of G. The problem of equality of discrete logarithms is the following: Given G, g_0, g_1, and two elements $y_0, y_1 \in G$, decide whether $\log_{g_0}(y_0)$ is equivalent to $\log_{g_1}(y_1)$. It is believed that there is no efficient algorithm to solve this problem. We make this precise in the following.

Definition 2.2.20. *Let $I = \{(G, g_0, g_1) | G$ is a cyclic group, $g_0, g_1 \in G$ generators$\}$ and $I_k = \{(G, g_0, g_1) \in I | \mathrm{ord}(G) = k\}$. Let*

$$EDL_{G,g_0,g_1} : G \times G \to \{0,1\}, EDL_{G,g_0,g_1}(x_0, x_1) = \begin{cases} 1 \text{ if } \log_{g_0} y_0 = \log_{g_1} y_1, \\ 0 \text{ otherwise}, \end{cases}$$

be a function. For every probabilistic polynomial-time algorithm A, every positive polynomial Q, and all sufficiently large k,

$$Pr[b = EDL_{G,g_0,g_1}(y_0, y_1) : (G, g_0, g_1) \xleftarrow{u} I_k, y_0 \xleftarrow{u} G, y_1 \xleftarrow{u} G, \\ b \leftarrow A(G, g_0, g_1, y_0, y_1)] < \frac{1}{2} + \frac{1}{Q(k)}.$$

This is called the assumption of equality of discrete logarithms (EDL).

The Factoring Assumption

The integer factorization problem is the following: Given a positive integer n, find its prime factorization, i.e., find pairwise distinct primes p_i and positive integer e_i such

that $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. All known factoring algorithms have an exponential running time. Therefore, it is widely believed that the factors of n cannot be computed efficiently. The following assumption make this statement more precise. For simplicity, assume $n = pq$.

Definition 2.2.21. *Let $I = \{n | n = pq, p$ and q are distinct primes, $|p| = |q|\}$ and $I_k = \{n \in I | n = pq, |p| = |q| = k\}$. For every probabilistic polynomial-time algorithm A, every positive polynomial Q, and all sufficiently large k,*

$$Pr[A(n) = p : n \overset{u}{\leftarrow} I_k] < \frac{1}{Q(k)}.$$

This is called the factoring assumption.

Furthermore, it is known that factoring $n = pq$ is equivalent to computing square roots in $\mathbb{Z}_n^*$ [189].

The RSA Assumption

Let $I = \{(n, e) \in I | n = pq, p \neq q$ primes, $0 < e < \phi(n), e$ prime to $\phi(n)\}$. The family

$$RSA = (RSA_{n,e} : \mathbb{Z}_n^* \to \mathbb{Z}_n^*, x \mapsto x^e)_{(n,e) \in I}$$

is called the RSA family.

Consider an $(n, e) \in I$, and let $d \in \mathbb{Z}_{\phi(n)}^*$ be the inverse of e mod $\phi(n)$. We have $x^{ed} \equiv 1 \bmod n$. This shows that $RSA_{n,e}$ is a bijection and that the inverse function is also an RSA function, namely $RSA_{n,d} : \mathbb{Z}_n^* \to \mathbb{Z}_n^*, y \mapsto y^d$.

$\mathrm{RSA}_{n,e}$ can be computed by an efficient modular exponentiation algorithm. The inverse d of e can be easily computed by the extended Euclidean algorithm if $\phi(n) = (p-1)(q-1)$ is known. No algorithm to compute $\mathrm{RSA}_{n,e}^{-1}$ in polynomial time is known if p, q and d are kept secret (d or p, q are called the trapdoor information for the RSA function).

To date, factoring n is the only known method to totally break RSA. All known factoring algorithms have an exponential running time. Therefore, it is widely believed that RSA cannot be efficiently inverted. The following assumption makes this more precise.

Definition 2.2.22. *Let $I_k = \{(n,e) \in I | n = pq, |p| = |q| = k\}$. For every probabilistic polynomial-time algorithm A, every positive polynomial Q, and all sufficiently large k,*

$$Pr[x = RSA_{n,d}(y) : (n,e) \xleftarrow{u} I_k, y \xleftarrow{u} \mathbb{Z}_n^*, x \leftarrow A(n,e,y)] < \frac{1}{Q(k)}.$$

This is called the RSA assumption.

The Quadratic Residuosity Assumption

Let $I = \{n : n = pq, p, q \text{ distinct primes}, |p| = |q|\}$ and let

$$J_n^{+1} = \left\{ x \in \mathbb{Z}_n^* | \left(\frac{x}{n}\right) = +1 \right\}$$

be the elements with Jacobi symbol $+1$. QR_n is a proper subset of J_n^{+1}.

Consider the functions

$$PQR_n : J_n^{+1} \to \{0,1\}, PQR_n(x) = \begin{cases} 1 \text{ if } x \in \mathrm{QR}_n, \\ 0 \text{ otherwise.} \end{cases}$$

It is believed that there is no efficient algorithm which, without knowing the factors of n, is able to decide whether $x \in J_n^{+1}$ is a quadratic residue. The following assumption make this precise.

Definition 2.2.23. *Let $I_k = \{n : n = pq, |p| = |q| = k\}$. For every probabilistic polynomial-time algorithm A, every positive polynomial Q, and all sufficiently large k,*

$$Pr[y = PQR_n(x) : n \xleftarrow{u} I_k, x \xleftarrow{u} J_n^{+1}, y \leftarrow A(n,x)] < \frac{1}{2} + \frac{1}{Q(k)}.$$

This is called the quadratic residuosity assumption.

Note that the factoring assumption follows from the RSA assumptions and also from the quadratic residuosity assumption. Hence, each of these two assumptions is stronger than the factoring assumption.

Discussions

For all assumptions described above, we do not consider a single fixed key $i \in I_k$: The success probability of adversary A is also taken over the random choice of the

key. Hence, the meaning of the probability statement is: Choosing both the key i with security parameter k and an instance randomly, the probability that adversary A correctly computes is small. The statement is not related to a particular key i. In practice, however, a public key i is chosen and then fixed for a long time, and it is known to the adversary. Thus, we are interested in the conditional probability of success, assuming a fixed public key i. Even if the security parameter k is very large, there may be public keys such that adversary A correctly breaks the system with a significant probability. However, as we will see in the following lemma, the number of such keys is negligibly small compared to the number of all keys with security parameter k. Hence, choosing i at random and uniformly from I_k, the probability of obtaining one for which adversary A has a significant chance of success is negligibly small.

Lemma 2.2.1. *Let $I = (I_k)_{k\in\mathbb{N}}$ be a key set with security parameter k. Let A_j be randomized algorithms with input x_j and output y_j, $1 \leq j \leq n$. Let B be a boolean function. Assume that A_n is the adversary A. Then the following statements are equivalent:*

1. *For every positive polynomial P and all sufficiently large k*

$$Pr[B(y_n) = 1 : i \leftarrow I_k, \{y_j \leftarrow A_j(x_j)\}_{1\leq j\leq n}] < \frac{1}{P(k)}.$$

2. *For all positive polynomials Q and R, and all sufficiently large k*

$$Pr\left[\left\{i \in I_k \,\middle|\, Pr[B(y_n) = 1 : \{y_j \leftarrow A_j(x_j)\}_{1\leq j\leq n}] > \frac{1}{Q(k)}\right\}\right] < \frac{1}{R(k)}.$$

Proof. Let

$$p_i = \Pr[B(y_n) = 1 : \{y_j \leftarrow A_j(x_j)\}_{1\leq j\leq n}]$$

be the conditional probability of success of A assuming a fixed i. We first prove that statement 2 implies statement 1. Let P be a positive polynomial. By statement 2, for sufficiently large k,

$$\Pr\left[\left\{i \in I_k | p_i > \frac{1}{2P(k)}\right\}\right] < \frac{1}{2P(k)}.$$

Hence

$$\begin{aligned}
&\Pr[B(y_n) = 1 : i \leftarrow I_k, \{y_j \leftarrow A_j(x_j)\}_{1 \le i \le n}] \\
&\quad = \sum_{i \in I_k} \Pr[i] \cdot p_i \\
&\quad = \sum_{p_i \le 1/(2P(k))} \Pr[i] \cdot p_i + \sum_{p_i > 1/(2P(k))} \Pr[i] \cdot p_i \\
&\quad < \sum_{p_i \le 1/(2P(k))} \Pr[i] \cdot \frac{1}{2P(k)} + \sum_{p_i > 1/(2P(k))} \Pr[i] \cdot 1 \\
&\quad = \Pr\left[\left\{ i \in I_k | p_i \le \frac{1}{2P(k)} \right\}\right] \cdot \frac{1}{2P(k)} + \Pr\left[\left\{ i \in I_k | p_i > \frac{1}{2P(k)} \right\}\right] \\
&\quad < \frac{1}{2P(k)} + \frac{1}{2P(k)} = \frac{1}{P(k)}.
\end{aligned}$$

Conversely, assume that statement 1 holds. Let Q an R be positive polynomials. Then for sufficiently large k

$$\begin{aligned}
\frac{1}{Q(k)R(k)} &> \Pr[B(y_n) = 1 : i \leftarrow I_k, \{y_j \leftarrow A_j(x_j)\}_{1 \le i \le n}] \\
&= \sum_{i \in I_k} \Pr[i] \cdot p_i \\
&\ge \sum_{p_i > 1/Q(k)} \Pr[i] \cdot p_i \\
&> \frac{1}{Q(k)} \cdot \Pr\left[\left\{ i \in I_k | p_i > \frac{1}{Q(k)} \right\}\right].
\end{aligned}$$

This inequality implies statement 2. □

2.3 Hash Functions

Hash functions can be used to control the integrity of a message. In signature schemes, hash functions are also applied to reduce messages of arbitrary lengths to message digests that can be signed in place of the original messages. Furthermore, hash functions can be employed as a substitution for the honest verifier in proofs of knowledge and thus turn them into signature schemes [90].

Definition 2.3.1. *A hash function is a function mapping a binary string of arbitrary*

finite length to a binary string of fixed length ℓ:

$$h : \{0,1\}^* \to \{0,1\}^\ell.$$

Naturally, we require that a hash function be efficiently computable. Additional security requirements of hash functions are motivated by the cryptographic purposes of applications. We now list three potential security properties.

- Preimage resistance (or one way): For a given y, it is computationally infeasible to find an x such that $h(x) = y$.
- Second preimage resistance (or weak collision resistance): For a given x, it is computationally infeasible to find an $x' \neq x$ such that $h(x) = h(x')$ [167].
- Collision resistance (or strong collision resistance): It is computationally infeasible to find a pair (x, x') with $x \neq x'$ such that $h(x) = h(x')$ if h is chosen at random from a family of hash functions [71].

It can easily be seen that the property of collision resistance implies the property of second preimage resistance. However, preimage resistance is not necessarily implied by collision resistance. For example, let $h' : \{0,1\}^k \to \{0,1\}^k$ be collision-resistant. We define $h(bx) = 1x$ if $b = 1$ and $= 0h'(x)$ if $b = 0$. We can see that h is collision-resistant but not one-way because a half of h^{-1} are polynomial-time computable. Damagård provides conditions under which collision resistance implies preimage resistance [64]; see also Gibson's comment in [102].

We give the precise definitions of one-way functions and collision-resistant hash functions as follows.

Definition 2.3.2 (One-Way Functions). *Let $I = (I_k)_{k\in\mathbb{N}}$ be a key set with security parameter k. Let K be a probabilistic polynomial sampling algorithm for I, which on input 1^k outputs $i \in I_k$.*

A family

$$f = (f_i : D_i \to R_i)_{i\in I}$$

of functions between finite sets D_i and R_i is a family of one-way functions (or, for short, a one-way function) with key generator K if and only if

1. *There is a uniform sampling algorithm S for $D = (D_i)_{i \in I}$, which on input $i \in I$ outputs $x \in D_i$.*

2. *f can be computed by a Monte Carlo algorithm $F(i, x)$. That is, there is a probabilistic polynomial-time algorithm $F(i, x)$ with*

$$Pr[F(i, x) = f_i(x)] \geq 1 - 2^{-k}, \ i \in I_k.$$

3. *f is hard to invert if the keys are generated by K. More precisely, for every probabilistic polynomial-time algorithm $A(i, y)$ ($i \in I, y \in R_i$), every positive polynomial P, and all sufficiently large k*

$$Pr[f_i(A(i, f_i(x))) = f_i(x) : i \leftarrow K(1^k), x \xleftarrow{u} D_i] < \frac{1}{P(k)}.$$

If K is a uniform sampling algorithm for I, then we call f a family of one-way function (or a one-way function) without explicitly referring to a key generator.

Definition 2.3.3 (Collision-Resistant Hash Functions). *Let $I = (I_k)_{k \in \mathbb{N}}$ be a key set with security parameter k. Let K be a probabilistic polynomial sampling algorithm for I, which on input 1^k outputs $i \in I_k$. Let $k(i)$ be the security parameter of i (i.e., $k(i) = k$ for $i \in I_k$) and $g : \mathbb{N} \to \mathbb{N}$ be a polynomial function.*

A family

$$h = (h_i : \{0,1\}^* \to \{0,1\}^{g(k(i))})_{i \in I}$$

of hash functions is called a family of collision-resistant hash functions (or, for short, a collision-resistant hash function) with key generator K if and only if

1. *The hash values $h_i(x)$ can be computed by a polynomial algorithm H with inputs $i \in I$ and $x \in \{0,1\}^*$.*

2. *It is computationally infeasible to find a collision. More precisely, for every probabilistic polynomial-time algorithm A, which on input $i \in I$ outputs messages $m_0, m_1 \in \{0,1\}^*$, $m_0 \neq m_1$, every positive polynomial P, and all sufficiently large k*

$$Pr[h_i(m_0) = h_i(m_1) : i \leftarrow K(1^k), \{m_0, m_1\} \leftarrow A(i)] < \frac{1}{P(k)}.$$

The following fact illustrates that collision-resistant hash functions can be made one-way.

Fact 2.3.1. *Let $I = (I_k)_{k\in\mathbb{N}}$ be a key set with security parameter k, and let a family $H = (h_i : \{0,1\}^* \to \{0,1\}^{g(k(i))})_{i\in I}$ of functions be collision-resistant. Let Q be a positive polynomial with $Q(k) \geq g(k) + 1$. Then the family*

$$(h_i : \{0,1\}^{Q(k(i))} \to \{0,1\}^{g(k(i))})_{i\in I}$$

of functions are one-way with respect to H's key generator.

We remark that one-way hash functions (OWHF) and collision-resistant hash functions (CRHF) are sometimes referred to as the following definitions [157]. A one-way hash function is a hash function satisfying preimage resistance and second preimage resistance. If the image elements have unique preimages, then second preimage resistance holds vacuously. A collision-resistant hash function is a hash function satisfying preimage resistance and collision resistance. A collision-resistant hash function is also called a collision-free hash function [63, 64, 67], or a collision-intractable hash function [222].

For many cryptographic applications, it is desirable that the only efficient way to determine the hash value $h(x)$ for a given x is to actually evaluate the function h at the value x. This should remain true even if many other values $h(x_1), h(x_2), \ldots$ have already been obtained. The random oracle model, which is introduce by Bellare and Rogaway [9], provides a mathematical model of such an ideal hash function. In this model, a hash function h is chosen randomly from a family of hash functions, and we are only permitted oracle access to the function h to obtain a hash value. This means that we are not given a formula or an algorithm to compute the value of the function h. Hence, the only way to compute a value $h(x)$ is to query the oracle. This can be thought of as looking up the value $h(x)$ for x in an enormous table of random numbers such that, for each possible x, there is a completely random value $h(x)$. Note that if the same input x is put to the oracle, identical output $h(x)$ is obtained.

Assuming that the best algorithm to find a collision of a hash function is brute-force search, the security against collisions depends mostly on the number of output-bits. To estimate the number of necessary output-bits, the so-called birthday attack

has to be considered: To find a collision with probability ϵ about

$$\sqrt{2 \cdot 2^{\ell} \ln(\frac{1}{1-\epsilon})}$$

random hashes must be evaluated. If we take $\epsilon = 0.5$, then the hashing just over $2^{\ell/2}$ random elements yields a collision. Hence, the birthday attack impose a lower bound on the sizes of secure message digests. A 40-bit message digest would be very insecure because a collision could be found with probability 1/2 with just over 2^{20} random hashes. Today, an output size of 160 bits (or larger) seems to have a reasonable security.

Many cryptographic hash functions are proposed so far in the literature. They are mainly divided into three types by constructions:

1. Hash functions based on block ciphers: A designer aims to implement a well-trusted hash function which is based on the security of a well-trusted block cipher such as DES.

2. Hash functions based on modular arithmetic: A designer aims to save on implementation costs. A hash function is generally used in conjunction with a digital signature algorithm which itself utilizes modular arithmetic.

3. Customized hash functions: Such hash functions, with their so-called "customized" design, tend to be fast and achieve a considerable advantage over the other types of hash functions.

Examples of the first type include DM [72, 152, 220] (as cited per Quisquater and Girault [187]), DES-based hash functions [159, 220, 221], FEAL-based N-hash [166], Snefru [158], Tandem DM, Abreast DM [136], MDC-2, and MDC-4 [24, 151, 160, 192]. The discrete logarithm-based hash functions [47, 103] and MASH-1 [122] are examples of the second type. Examples of the third type are MD4 [193], MD5 [194], SHA-1 [91], SHA-2 (namely SHA-256, SHA-384, and SHA-512) [92], HAVAL [182], RIPEMD [192], and RIPEMD-160 [21]. Several hash functions are analyzed and found to be insecure in the literature. For example, collisions for MD4 are found by Dobbertin [79] and a first partial attack on MD5 are published by den Boer and Bossalaers [74]. In particular, using techniques related to his attack on MD4, Dobbertin [80]

finds MD5 collisions in 10 hours on a personal computers (about 2^{34} hash function computation). For comprehensive surveys on hash functions, see Preneel [185, 186].

2.4 Indistinguishability of Probability Ensembles

Distribution indistinguishability is one of the most important concepts in defining the security of cryptosystems. In this section, we present several basic definitions relative to indistinguishability.

Definition 2.4.1 (Negligible Functions). *We call a function $f : \mathbb{N} \to \mathbb{R}$ negligible if for every positive polynomial $P(\cdot)$, there exits an n_0 such that for all $n > n_0$,*

$$f(n) < \frac{1}{P(n)}.$$

In the future we shall use the shorter phrase "for all sufficiently large n" to represent the phrase "there exists an n_0 such that for all $n > n_0$."

Definition 2.4.2. *Let $I = \bigcup_{k\in\mathbb{N}} I_k$ be an infinite index set which is partitioned into finite disjoint subsets I_k. Assume that the indexes are encoded as binary numbers. We denote $|i|$ the binary length of i. I is called an index set with security parameter k if the security parameter k of $i \in I$ can be derived from i by a deterministic polynomial algorithm and there is a polynomial P such that $|i| = P(k)$. We usually write $I = (I_k)_{k\in\mathbb{N}}$ instead of $I = \bigcup_{k\in\mathbb{N}} I_k$.*

Definition 2.4.3 (Probability Ensembles). *Let I be a countable index set. An ensemble indexed by I is a sequence of random variables indexed by I. Namely, any $X = \{X_i\}_{i\in I}$, where each X_i is a random variable, is an ensemble indexed by I.*

In general either $\mathbb{N}$ or a subset of $\{0,1\}^*$ is used as the index set. An ensemble of the form $X = \{X_n\}_{n\in\mathbb{N}}$ has each X_n range over strings of length poly(n), whereas an ensemble of the form $X = \{X_w\}_{w\in\{0,1\}^*}$ will have each X_w range over strings of length poly$(|w|)$. Note that poly$(\cdot)$ stands for some unspecified fixed polynomial.

Let A be a probabilistic algorithm and X_k a random variable. The probability that A accepts X_k is defined by

$$\Pr[A(X_k) = 1] = \sum_x \Pr[A(x) = 1] \cdot \Pr[X_k = x].$$

Let I be an index set with security parameter k. Let $X = \{X_i\}_{i\in I}$ and $Y = \{Y_i\}_{i\in I}$ be two ensembles of random variables indexed by string $i \in I$. Intuitively, if we cannot distinguish X_i and Y_i, we should not be able to tell which source (either X_i or Y_i with equal probability) a random sample α is taken from. This concept can be extended to probability ensembles X and Y. In the following, we give the definitions of indistinguishability of probability ensembles [110, 111].

Definition 2.4.4 (Computational Indistinguishability). *The probability ensembles X and Y are computationally indistinguishable if for every probabilistic polynomial-time algorithm A, every positive polynomial P, and all sufficiently long $k \in I$, it holds that*

$$\mid Pr[A(X_k) = 1] - Pr[A(Y_k) = 1] \mid < \frac{1}{P(k)}.$$

Alternatively, we have the following definition.

Definition 2.4.5. *Let Z_0 and Z_1 be probability ensembles. Z_0 and Z_1 are computationally indistinguishable if for every probabilistic polynomial-time algorithm A, for every positive polynomial P, and all sufficiently long $k \in I$, it holds that*

$$Pr[b' = b : b \leftarrow \{0,1\}, a \leftarrow Z_b, b' \leftarrow A(a)] < \frac{1}{2} + \frac{1}{P(k)}$$

Definition 2.4.6 (Statistical Indistinguishability, Statistical Closeness). *The probability ensembles X and Y are statistically indistinguishable (or statistically close) if for every polynomial P and for all sufficiently long $k \in I$, it holds that*

$$\sum_{\alpha\in\{0,1\}^*} \mid Pr[X_k = \alpha] - Pr[Y_k = \alpha] \mid < \frac{1}{P(k)}.$$

The statistical difference (or statistical distance) between two ensembles, X and Y, is defined by

$$\triangle(X,Y) = \frac{1}{2} \sum_{\alpha\in\{0,1\}^*} \mid Pr[X_w = \alpha] - Pr[Y_w = \alpha] \mid .$$

That is, two ensembles X and Y are statistically indistinguishable if $\triangle(X_k, Y_k)$ is a negligible function of $|k|$.

Definition 2.4.7 (Perfect Indistinguishability). *The probability ensembles X and Y are perfectly indistinguishable if for all $i \in I$ the random variables X_i and Y_i are identically distributed. That is, $\triangle(X_i, Y_i) = 0$ for all $i \in I$.*

2.5 Interactive Protocols and Proof Systems

Definition 2.5.1 (Interactive Protocols). *An interactive protocol is a pair of algorithms (P, V) for two communicating parties Peggy and Vic. It is common to call Peggy the prover and Vic the verifier. The parties send messages back and forth and perform some computation as prescribed by the specification of the protocol. Let x denote the common input and w and z denote the respective private inputs of P and V. The output of V is denoted by $\langle P(w), V(z)\rangle(x)$. Vic's view of a protocol with Peggy consists of the entire list of parameters Vic "sees" during the execution of the protocol and is denoted $view_V^P(x)$. This includes all communicated values, Vic's inputs and outputs, as well as all computations and random choices made by Vic.*

More formally, the communicating parties are assumed to be two Turing machines P and V. Both machines have their own working tape, input tape, and random tape; and both are able to read the same input from a read-only tape. The exchange of messages takes place in two communication tapes. One tape is a write-only tape for P and a read-only one for V, while the other is a write-only tape for V and a read-only one for P.

In an interactive protocol, the two parties alternately perform rounds that consist of:

1. Receive a message from the opposite party.
2. Perform some computation.
3. Send a message to the opposite party.

We remark that an interactive protocol may specify a sequence of rounds, which is then repeated a specified number of times.

If the prover and the verifier follow the behavior prescribed in the protocol, they are called an honest prover and an honest verifier. A prover that does not know the prover's secret but tries to convince the verifier otherwise is called a dishonest prover. A verifier that does not follow the behavior specified in the protocol is called a dishonest verifier. Dishonest parties are not restricted to storing only their specified output, but are assumed to store their entire view. Note that each party, whether honest or

not, complies with the syntax of the communication interface because not following the syntax is immediately detected. In other words, she may only be dishonest in her private computations and the resulting data that she transmits. Moreover, whenever the protocol specifies that a party must verify a condition, if the verification fails, then the protocol is stopped and all parties are notified.

We introduce some notations that will later be used when analyzing the properties of a protocol. Let (P, V) be an interactive protocol. We denote by P the algorithm that an honest prover executes, by P^* the algorithm that a dishonest prover executes, by V the algorithm of an honest verifier, and by V^* the algorithm of a general (possibly dishonest) verifier. To analyze the properties of the prover that manifests the knowledge of a secret, a third party, called a knowledge extractor, is often used. This knowledge extractor interacts with the prover, but, in contrast to a possibly dishonest verifier, has the ability to reset and restart the prover at will. Let $K^{P^*(x)}$ denote the output of the knowledge extractor K that is given oracle access to P^*, i.e., can reset and restart P^* on input x. We remark that in complexity theory $M^{(\cdot)}(\cdot)$ often denotes an oracle machine M that makes oracle queries. The running time of an oracle machine is the number of steps made during its computation, and the oracle's reply on each query is obtained in a single step.

Definition 2.5.2 (Interactive Proof Systems). *An interactive proof system (P, V) is a protocol between a prover P and a verifier V. The prover P is computationally unbounded, while the verifier V runs a probabilistic polynomial-time algorithm. The input to the protocol is a string x, known to both parties. The two exchange a sequence of messages $m_1, m_2, \ldots, m_{2|x|^k}$, where the prover sends the odd-numbered ones, and the verifier the even ones. The transcript, denoted by $tr_{P,V}(x)$, is defined as all exchanged messages $(m_1, m_2, \ldots, m_{2|x|^k})$. All messages are polynomial in length: $|m_i| \leq |x|^k$. The prover goes first, say.*

The messages are defined as follows: $m_1 = P(x)$—that is, the first message is produced by the prover based on input x alone. Subsequently, and for all $i \leq |x|^k, m_{2i} = V(x, m_1, \ldots, m_{2i-1}, r_i)$, and $m_{2i-1} = P(x, m_1, \ldots, m_{2i-2})$. Here r_i is the polynomial long random string used by the verifier at the ith exchange. Notice that the prover does not know r_i. Each even-numbered message is computed by the verifier based on the input, r_i and all previous messages, while each odd-numbered one is computed by

the prover based on the input and all previous messages. Finally, if the last message is $m_{2|x|^k} \in \{1, 0\}$ *the verifier accepts or rejects the common input. Usually* $m_{2|x|^k} = 1$ *is interpreted as "accept", whereas* $m_{2|x|^k} = 0$ *is interpreted as "reject."*

(P, V) *is an interactive proof system for a language* L *if it satisfies the following two conditions:*

1. *Completeness: For every* $x \in L$,

$$Pr[\langle P, V\rangle(x) = 1] \geq \frac{2}{3}.$$

2. *Soundness: For every* $x \notin L$ *and every prover,*

$$Pr[\langle P^*, V\rangle(x) = 1] \leq \frac{1}{3}.$$

Note that for soundness condition, V must be able to resist any attempt of cheating from any prover.

Definition 2.5.3 (The Class IP). *The complexity class* IP *is the set of languages that have interactive proof systems.*

When a protocol is designed for use in practice, a prover usually cannot be more computationally powerful than probabilistic polynomial-time. Therefore, it makes sense to consider the situation where a prover's computational powers are also limited to probabilistic polynomial time, and the prover's power comes from the fact that he takes some auxiliary input called a "witness." More formally:

Definition 2.5.4 (Computationally Sound Proof Systems, Interactive Arguments). *An interactive protocol* (P, V) *is a computationally sound proof System (or an interactive argument) for a language* L *if both algorithms are probabilistic polynomial-time with auxiliary inputs and the following two conditions hold:*

1. *Completeness: For every* $x \in L$,

$$Pr[\langle P(w), V(z)\rangle(x) = 1] \geq \frac{2}{3}.$$

2. *Soundness: For every* $x \notin L$ *and every prover,*

$$Pr[\langle P^*(w), V(z)\rangle(x) = 1] \leq \frac{1}{3}.$$

Interactive proofs of knowledge allow the prover to prove to the verifier that he knows the "witness," not merely its existence. The concept of proofs of knowledge is first mentioned as a remark in [110]. Formal definitions are first given by Feige, Fiat, and Shamir [87] and by Tompa and Woll [215], and further refined by Feige and Shamir [88] and by Bellare and Goldreich [8]. Here we give a definition of a proof of knowledge similar to that presented in [8, 88].

Definition 2.5.5 (Interactive Proof of Knowledge). *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a polynomially bounded binary relation and let L_R be the language defined by R. An interactive proof of knowledge for the relation R is a protocol (P, V) that satisfies the following two conditions:*

1. *Completeness: If $(x, w) \in R$ then $Pr[\langle P(w), V\rangle(x) = 1] = 1$.*
2. *Validity: There exits a probabilistic expected polynomial-time algorithm K (knowledge extractor) such that for every P^*, every polynomial Q and all sufficiently large $x \in L_R$,*

$$Pr[(x, K^{P^*(x)}) \in R] \geq Pr[\langle P^*, V\rangle(x) = 1] - \frac{1}{Q(|x|)}.$$

The probabilities are taken over all random choices of V, P, P^*, and K, respectively.

We note that the above definition makes no requirements for the case $x \notin L_R$ because a proof of knowledge for R does not necessarily give an interactive proof for language membership in L_R. Hence, soundness (i.e., a bound on the prover's ability to lead the verifier to accept $x \notin L_R$) is not required. If soundness is necessary or if the protocol is to run on input $x \notin L_R$, then the following additional property should be satisfied.

Soundness: For every P^*, $\forall x \notin L_R$,

$$Pr[\langle P^*, V\rangle(x) = 1] < \frac{1}{2}$$

holds. The probabilities are taken over all random choices of P^* and V.

If a protocol satisfies this soundness property, the probability of accepting if $x \notin L_R$ can be made arbitrarily small by repeating it sequentially sufficiently many times.

2.6 Zero-Knowledge Proof Systems

Definition 2.6.1 (Perfect/Statistical/Computational Zero Knowledge). *An interactive protocol (P, V) is said to be perfect/statistical/computational zero-knowledge if for every probabilistic polynomial-time verifier V^* there exists a probabilistic (expected) polynomial-time simulator S_{V^*} so that the two ensembles*

- $\{\langle P, V^* \rangle(x)\}_{x \in L}$
- $\{S_{V^*}(x)\}_{x \in L}$

are perfectly/statistically/computationally indistinguishable. Moreover, a protocol is simply said to be zero-knowledge if it is computational zero-knowledge.

An alternative, but equivalent, definition is to require the simulator $S(\cdot)$ to output V^*'s view $view^P_{V^*}(x)$ rather than V^*'s output. The view $view^P_{V^*}(x)$ consists of the entire sequence of the local configurations of the verifier during an interaction execution with the prover. That is, $view^P_{V^*}(x) = (x, r, tr_{P,V^*}(x))$, where r is random bits of V^*. It suffices to consider only the content of the random bits of V^* and the sequence of messages that V^* has received from the prover during the execution because the entire sequence of local configurations and the final output are determined by those objects.

To prove that a protocol is zero-knowledge according to Definition 2.6.1, one would have to construct a simulator for every possible verifier. This is often done by constructing a universal simulator that works for all verifiers. In particular, the universal simulator S_{V^*} uses any verifier V^* as a black box such that V^* outputs $V^*(q, r)$ for an input q (S_{V^*}'s query) and random bits r. S_{V^*} does not try to dissect V^* to see how V^* works, but S_{V^*} can rewind V^* to some previous execution state. If an interactive proof system is proved zero-knowledge by treating V^* as a black box in simulation, we call it black-box zero-knowledge. By black-box simulation, one often needs to allow probabilistic expected polynomial-time simulators (i.e., a Las Vegas algorithm) in order to have constant-round zero-knowledge arguments. Most of the known zero-knowledge protocols make use of the black-box techniques in their simulation. However, there are several negative results about the power of black-box simulators. Goldreich and Krawczyk show that obtaining black-box 3-round

zero knowledge proofs and constant-round Arthur-Merline zero-knowledge proofs is impossible [109]. Canetti, Kilian, Petrank, and Rosen show that no constant-round protocol is bounded concurrent zero-knowledge with a black-box simulator [41]. We remark that by the definition of zero-knowledge, the black-box simulation is not the only way to show the zero-knowledge property. Barak presents the first constructions of non-black-box simulators [7]. Using the new non-black-box techniques, he obtain several results that are previously proved to be impossible to obtain using the black-box simulators. See [7] for more details.

In many applications, the verifier interacting with the prover may have some additional a priori information z that may assist it in its attempts to extract knowledge from the prover. To model this concept, auxiliary-input zero knowledge is defined as follows [107]. The conditions are all the same as those for zero-knowledge except that the verifier and the simulator are allowed an extra input z the size of which is polynomially bounded in the size of x. The protocol is denoted by $\langle P, V(z)\rangle(x)$. If the basic protocol is auxiliary-input zero-knowledge, then sequential compositions can be shown to be zero-knowledge. However, a parallel composition of zero-knowledge protocols is in general no longer zero-knowledge. Furthermore, if a protocol is proved zero-knowledge with the black-box simulation techniques, it is also auxiliary-input zero-knowledge.

A slight weaker requirement than zero knowledge is honest-verifier zero-knowledge. A verifier is honest if its messages to P are exactly its random bits. In complexity terms, we say that the verifier tosses public coins. Honest-verifier zero knowledge requires simulatability of the view of only the honest verifier, rather than simulatability of the view of any possible probability polynomial-time verifier.

Definition 2.6.2 (Honest-Verifier Zero Knowledge). *An interactive protocol (P, V) is said to be perfect/statistical/computational honest-verifier zero-knowledge, if there exists a probabilistic (expected) polynomial-time simulator S_V so that the two ensembles*

$$\{\langle P, V\rangle(x)\}_{x\in L} \text{ and } \{S_V(x)\}_{x\in L}$$

are perfectly/statistically/computationally indistinguishable.

In many interactive protocols, it is essential that the transcript of the interaction does not yield any evidence of the interaction. Such protocols (e.g. undeniable

signatures [51] and deniable authentication [81, 83]) are said to be deniable. The standard definition of zero-knowledge in the plain model certainly satisfies deniability. However, this is no longer the case with the definitions of zero knowledge in the random oracle model [9]. This results from the fact that in the real world the public information in the random oracle model is fixed once and for all at start-up. However, when proving security, the simulator in the random oracle model is allowed to choose this public information (e.g. the random functions) in any way it pleases as long as it "looks ok." Thus even though there exists a simulator for a protocol, there is no guarantee that one can actually simulate a transcript using a certain predefined public information. For several settings in which zero knowledge and deniability are the goals, the standard definitions of zero knowledge in the random oracle model are not sufficient because they do not guarantee deniability. In the following we recall the definition of deniable zero knowledge in the random oracle model [178].

Definition 2.6.3 (Deniable Zero Knowledge). *An interactive protocol (P, V) is said to be deniable zero-knowledge in the random oracle (RO) model if for every probabilistic polynomial-time verifier V^* there exists a probabilistic (expected) polynomial-time simulator S_{V^*} so that the following two ensembles are computationally indistinguishable:*

- $\{RO, \langle P^{RO}, V^{*RO}\rangle(x)\}_{x\in L}$
- $\{RO, S^{RO}_{V^*}(x)\}_{x\in L}$,

where $RO : \{0,1\}^{poly(|x|)} \to \{0,1\}^{poly(|x|)}$ is a uniformly distributed random variable.

That is, for every probabilistic algorithm D running in time polynomial in the length of its first input, every polynomial P, all sufficiently long $x \in L$, it holds that

$$|Pr[D^{RO}(x, \langle P^{RO}, V^{*RO}\rangle(x)) = 1] - Pr[D^{RO}(x, S^{RO}_{V^*}(x)) = 1]| < \frac{1}{P(|x|)},$$

where $RO : \{0,1\}^{poly(|x|)} \to \{0,1\}^{poly(|x|)}$ is a uniformly distributed random variable.

We note that when proving security according to the standard zero-knowledge definition in the random oracle model, the simulator has two advantages over a plain model simulator, namely

1. The simulator can see what values parties query the oracle on.

2. The simulator can answer these queries in whatever way it chooses as long as the answers "look ok."

An easy way of seeing this is by noting that non-interactive zero-knowledge proofs [15] are possible in the random oracle model. A player receiving a non-interactive proof of an assertion can definitely do something new: it can simply send the same proof to someone else. This seems to contradict the zero-knowledge property. However, we note that the simulator for the non-interactive zero knowledge is much stronger in the random oracle model than in the plain model. In fact, the zero-knowledge property in the random oracle model only guarantees that the verifier will not be able to do anything new without referring to the random oracle.

The definition of deniable zero knowledge in the random oracle model restricts the power of the simulator and only allows it to see what values parties query the oracle on. This is due to the fact that in the definition of deniable zero knowledge in the random oracle model, the distinguisher is given access to the random oracle and can thus verify whether the simulator has answered the oracle queries in compliance with the predefined oracle.

2.7 Witness Indistinguishability and Hiding

Feige and Shamir propose an alternative privacy criteria: witness indistinguishability and witness hiding [88]. Both notions seem weaker than zero-knowledge, yet they suffice for several practical applications. Moreover, they have the advantage over zero knowledge in that they are preserved under parallel compositions, provided that the prover is probabilistic polynomial-time.

A variation of witness indistinguishability from Goldreich is witness independence [108]. Roughly speaking, an interactive proof of knowledge for an NP relation is witness-indistinguishable (resp. witness-independent) if the verifier's view of the interaction with the prover is computationally independent (resp. statistically independent) of the private input of the prover. Intuitively, this implies that the verifier cannot tell which witness the prover is using even if the verifier knows all witnesses.

Definition 2.7.1 (Witness Indistinguishability/Independence). *Let (P, V) be a proof of knowledge for an NP relation R. We say that (P, V) is witness-indistinguishable*

for R if for every probabilistic polynomial-time verifier V^*, all sufficiently long $x \in L_R$, any two sequences $W = \{w_x\}_{x \in L_R}$ and $W' = \{w'_x\}_{x \in L_R}$ such that $w_x, w'_x \in R(x)$, and all auxiliary inputs $z \in \{0,1\}^*$, the following two ensembles are computationally indistinguishable:

- $\{\langle P(w_x), V^*(z)\rangle(x)\}_{x \in L_R}$.
- $\{\langle P(w'_x), V^*(z)\rangle(x)\}_{x \in L_R}$.

We say that (P, V) is witness-independent for R if the random variables

$$\langle P(w_x), V^*(z)\rangle(x) \text{ and } \langle P(w'_x), V^*(z)\rangle(x)$$

are identically distributed.

An alternative, but equivalent, definition is to require two V^*'s views rather than two V^*'s outputs to be computationally indistinguishable (or identically distributed).

Feige and Shamir prove that witness indistinguishability is preserved under polynomial composition of protocols [88]. Furthermore, any zero-knowledge (resp. perfect zero-knowledge) protocol is witness-indistinguishable (resp. witness-independent). On the other hand, witness indistinguishability does not imply zero knowledge. In particular, any proof system for an instance having only a single witness is trivially witness-indistinguishable, but may not be zero-knowledge.

Intuitively, a proof of knowledge for an NP relation is witness-hiding if interacting with the prover does not help a (dishonest) verifier to find a witness for the common input which he does not know at the beginning of the protocol. Because each NP language has instances for which witness-finding is easy, we must consider the task of witness-finding for specially selected hard instance. Before defining what a witness-hiding proof is, we need the definition of distribution of hard instances.

Definition 2.7.2 (Distribution of Hard Instances). *Let R be an NP relation and L_R be the language defined by R. Let $X = \{X_n\}_{n \in \mathbb{N}}$ be a probabilistic ensemble such that X_n ranges over $L_R \cap \{0,1\}^n$. We say that X is hard for R if for every probabilistic polynomial-time (witness-finding) algorithm F, every polynomial Q, all sufficiently large n, and all $z \in \{0,1\}^{\text{poly}(n)}$,*

$$Pr[F(X_n, z) \in R(X_n)] < \frac{1}{Q(n)}.$$

Definition 2.7.3 (Witness Hiding). *Let (P, V) be a proof of knowledge for a NP relation R. Let $X = \{X_n\}_{n\in\mathbb{N}}$ be a hard-instance ensemble for R. We say that (P, V) is witness-hiding for the relation R under the instance ensemble X if for every probabilistic polynomial-time algorithm V^*, every polynomial Q, all sufficiently large n's, and all $z \in \{0,1\}^*$,*

$$Pr[\langle P(W_n), V^*(z)\rangle(X_n) \in R(X_n)] < \frac{1}{Q(n)},$$

where W_n is arbitrary distributed over $R(X_n)$.

Witness hiding guarantees only that witnesses are not disclosed completely. In contrast to zero knowledge, partial information may be leaked. For example, a digital signature cannot be zero-knowledge (or deniable zero-knowledge in the random oracle model), but they can be witness-hiding (e.g., fail-stop signatures [119, 217]).

A witness indistinguishability proof is not necessarily witness-hiding. For example, any proof system for an instance having only a single witness is trivially witness-indistinguishable, but may not be witness-hiding. However, as shown in [88], if each instance has at least two computationally independent witnesses, then a witness-indistinguishable proof of knowledge is also witness-hiding.

Chapter 3

Elementary Cryptographic Tools

This chapter presents several elementary cryptographic tools that can be used as building blocks for complex cryptographic applications. These tools include encryption schemes, signature schemes, and basic cryptographic protocols. There are numerous books devoted to theory and practice of cryptography. We refer the reader to [33, 73, 108, 157, 202, 210, 214].

3.1 Public-Key Encryption Schemes

Encryption schemes allow one party to send messages to another party securely. To deliver data confidentially, the sender applies an encryption function to a message (called plaintext) to obtain ciphertext, which is then sent. Only the intended receiver is able to retrieve the original plaintext from the ciphertext through the corresponding decryption function.

Encryption schemes can be divided into two types: secret-key (or symmetric) schemes and public-key (or asymmetric) schemes. In a secret-key scheme, the same key is used for encryption and decryption. Hence, when two parties want to securely communicate with symmetric encryption scheme, they need to exchange a secret key in advance. In a public-key scheme, different keys are used for encryption and decryption. The public key used for encryption can be published, while the secret key used for decryption must be kept secret.

Work on public-key encryption schemes begins with Diffie and Hellman's paper

"New Directions in Cryptography" [78] in 1976. They invent public-key cryptography and propose a concrete scheme for obtaining a common secret key over an insecure channel. The notion of a trapdoor one-way function is put forth. This is a function that is easy to compute but hard to invert—unless a trapdoor is known. In 1978, Rivest, shamir, and Adleman propose the RSA scheme that depends on the intractability of factoring large integers [195]. They are the first to present a concrete realization of a trapdoor one-way function. In 1984, ElGamal discovers another scheme that is based on the trapdoor Diffie-Hellman problem [84, 85].

A public-key encryption scheme is a triple of algorithms (gen, enc, dec).

- Key generation algorithm gen: This is a probabilistic polynomial-time algorithm $\text{gen}(1^k) = (sk, pk)$, where 1^k is a secure parameter, sk and pk are a pair of secret and public keys, each of size $O(k^a)$ for $a \in \mathbb{N}$ a constant.

- Encryption algorithm enc: This is often a probabilistic algorithm $\text{enc}(pk, m) = c$, where m is a message in the message space M, and c is the corresponding ciphertext in the ciphertext space C.

- Decryption algorithm dec: This is a deterministic algorithm $\text{dec}(sk, c) = m$ (i.e., $\text{dec}(sk, \text{enc}(pk, m)) = m$) for every $m \in M$, where sk and pk are a pair of secret and public keys.

If the algorithm enc is probabilistic, the encryption scheme is called probabilistic.

Often we would like to tell how secure an encryption scheme is. A way to define secure encryption is by considering separately the various goals of encryptions and the possible attack models of adversaries. Then, a particular goal and a attack model are combined to obtain the desired definition.

In the literature two different goals have been considered: indistinguishability of encryptions (IND) [111] and non-malleability (NM) [81]. IND is also called polynomial security. IND requires that it be infeasible for an adversary to distinguish between the ciphertexts of any two messages, even if the original messages are given. In terms of protecting the data that is encrypted, the most basic is privacy, which requires that an adversary should not be able to learn any useful information about the plaintext from the ciphertext. Semantical security captures in the most direct way the notions of privacy: Whatever can be efficiently computed about a message

given the ciphertext can be computed without the ciphertext [111]. Assume that an adversary A consists of two sub-algorithms A_1 and A_2. In addition A_1 can output some useful state information s that will be passed to A_2. We make polynomial security and semantical security precise in the following definitions.

Definition 3.1.1 (Polynomial Security). *A public-key encryption scheme (gen, enc, dec) is polynomially secure if for every probabilistic polynomial-time adversary A, every positive polynomial P, and all sufficiently large k, it holds that*

$$Pr[b' = b : (sk, pk) \leftarrow gen(1^k), (s, m_0, m_1) \leftarrow A_1(pk) \text{ where } |m_0| = |m_1|, b \leftarrow \{0,1\},$$
$$c \leftarrow enc(pk, m_b), b' \leftarrow A_2(s, m_0, m_1, pk, c)] < \frac{1}{P(k)}.$$

Definition 3.1.2 (Semantical Security). *Let M be a message space and let R be any polynomially-bounded relation that is recognizable in probabilistic polynomial time. A public-key encryption scheme (gen, enc, dec) is semantically secure if for every probabilistic polynomial-time adversary A, every positive polynomial P, and all sufficiently large k, there is a probabilistic polynomial-time simulator S such that*

$$|p_0(k) - p_1(k)| < \frac{1}{P(k)},$$

where

$$p_0(k) = Pr[R(m, z) : (sk, pk) \leftarrow gen(1^k), (s, M, R) \leftarrow A_1(pk) \text{ where all}$$
$$m_0, m_1 \in M, |m_0| = |m_1|, m \leftarrow M, c \leftarrow enc(pk, m),$$
$$z \leftarrow A_2(s, M, R, pk, c)]$$

$$p_1(k) = Pr[R(m, z) : (sk, pk) \leftarrow gen(1^k), (s, M, R) \leftarrow A_1(pk) \text{ where all}$$
$$m_0, m_1 \in M, |m_0| = |m_1|, m \leftarrow M, z \leftarrow S(s, M, R, pk)].$$

It can be proved that a cryptosystem (gen, enc, dec) is semantically secure if and only if it is polynomially secure [111, 162]. Because semantic security is subtle and is not easy to use, IND is often used when one analyzes the security of encryption schemes.

A second goal NM requires that an adversary given a challenge ciphertext be unable to obtain a different ciphertext such that the plaintexts underlying these two ciphertexts are "meaningfully related." This is made precise as follows.

Definition 3.1.3 (Non-Malleability). *Let M be a message space and let R be any polynomially-bounded relation that is recognizable in probabilistic polynomial time. A public-key encryption scheme (gen, enc, dec) is non-malleable if for every probabilistic polynomial-time adversary A, there is a probabilistic polynomial-time simulator S such that for every positive polynomial P, and all sufficiently large k, it holds that*

$$|p_0(k) - p_1(k)| < \frac{1}{P(k)},$$

where

$$\begin{aligned} p_0(k) = Pr[R(m, dec(z)) : (sk, pk) \leftarrow gen(1^k), (s, M, R) \leftarrow A_1(pk) \text{ where all} \\ m_0, m_1 \in M, |m_0| = |m_1|, m \leftarrow M, c \leftarrow enc(pk, m), \\ z \leftarrow A_2(s, M, R, pk, c)] \end{aligned}$$

$$\begin{aligned} p_1(k) = Pr[R(m, dec(z)) : (sk, pk) \leftarrow gen(1^k), (s, M, R) \leftarrow A_1(pk) \text{ where all} \\ m_0, m_1 \in M, |m_0| = |m_1|, m \leftarrow M, z \leftarrow S(s, M, R, pk)], \end{aligned}$$

The adversaries may be passive or active. A passive adversary is a probabilistic polynomial-time algorithm that has pairs of plaintext and ciphertext. An active adversary is a probabilistic polynomial-time algorithm that accesses the encryption algorithm or even the decryption oracle in an adaptive way. Both these two goals (IND and NM) can be considered under three different active attacks: chosen-plaintext attack (CPA), non-adaptive chosen-ciphertext attack (CCA1) [171], and adaptive chosen-ciphertext attack (CCA2) [191]. Under CPA the adversary can obtain ciphertext of any plaintext. Public-key encryption schemes have to be safe against the attack. Under CCA1 the adversary can get access to an oracle for the decryption function only for the period of time preceding his being given the challenge ciphertext. In other words, adversary's queries to the decryption oracle cannot depend on the challenge ciphertext. However, under CCA2 the adversary can continue getting access to an oracle for the decryption function even after obtaining the challenge ciphertext. The only restriction is that the adversary cannot make the decryption oracle decrypt the challenge ciphertext. A public-key encryption scheme is more secure if it can withstand the attack from more capable adversaries.

One can combine the goals with the attack models to get six basic notions of security: IND-CPA [111], IND-CCA1 [171], IND-CCA2 [191], NM-CPA, NM-CCA1, and NM-CCA2 [81, 82].

3.1.1 The Diffie-Hellman Key Agreement

Suppose Alice and Bob wish to use a symmetric encryption system to keep their communication over an insecure channel secret. Initially, Alice and Bob must agree on a secret key. The Diffie-Hellman key-agreement system [78] enables Alice and Bob to use their insecure channel for this key agreement. The protocol is a milestone in public-key cryptography.

The Diffie-Hellman protocol works as follows. Alice and Bob wish to agree on a common secret key. First, they agree on a large prime number p and a primitive root g modulo p with $2 \leq g \leq p-2$. The prime p and the primitive root g can be publicly known. Now Alice chooses an integer $a \in \mathbb{Z}_{p-1}$ randomly. She computes $A = g^a \bmod p$ and sends the result A to Bob. She keeps the exponent a secret. Bob chooses an integer $b \in \mathbb{Z}_{p-1}$ randomly. He computes $B = g^b \bmod p$ and sends the result to Alice. He keeps his exponent b secret. To obtain the common secret key, Alice computes $B^a \bmod p = g^{ab} \bmod p$ and Bob computes $A^b \bmod p = g^{ab} \bmod p$. The common key is $g^{ab} \bmod p$. Hence, Alice and Bob can use an insecure communication channel for this agreement.

The security of the scheme is based on the Diffie-Hellman assumption. It is this scheme that gives the assumption its name. In addition, a secure and efficient Diffie-Hellman key-agreement protocol can be implemented in all cyclic groups in which the Diffie-Hellman problem is difficult to solve and for which the group operations can be efficiently implemented.

3.1.2 The RSA Encryption Scheme

The encryption scheme is proposed by Rivest, Shamir, and Adleman [195]. It is the first concrete realization of a trapdoor one-way function as introduced by Diffie-Hellman [78].

This encryption scheme works as follows. Assume Alice wants to send a message

$0 \leq m < n$ to Bob.

- **Key generation:** $\text{gen}(1^k) = ((n, d), (n, e))$.
  Bob picks randomly and independently two large primes p and q, and computes $n = pq$. He also chooses an integer e such that $1 < e < (p-1)(q-1)$ and $\gcd(e, (p-1)(q-1)) = 1$. Then Bob computes an integer d such that $1 < d < (p-1)(q-1)$ and $de \equiv 1 \pmod{(p-1)(q-1)}$. Because $\gcd(e, (p-1)(q-1)) = 1$, such a number d exits. Bob's public key is (n, e) and secret key is (n, d).

- **Encryption:** $\text{enc}(m, (n, e)) = c$.
  Alice encrypts the plaintext m by computing $c = m^e \bmod n$. The ciphertext is c.

- **Decryption:** $\text{dec}(c, (n, d)) = m$.
  Bob can reconstruct the plaintext as $m = c^d \bmod n$.

The security of RSA is related to the intractability of factoring integers; however, it is not known if breaking RSA is as difficult as factoring integers. But it has been shown that computing d from the public key (n, e) is as difficult as finding the prime factors p and q of n. If $e = 2$, the scheme is called the Rabin encryption scheme [189]. In contrast with RSA, it can be shown that breaking the Rabin encryption scheme efficiently is equivalent to efficiently factoring integers.

3.1.3 The ElGamal Encryption Scheme

The encryption scheme is proposed by ElGamal [84, 85]. It can be seen as a special application of the Diffie-Hellman key-agreement protocol.

The encryption scheme works as follows. Assume Alice wants to send a message $m \in \{0, 1, \ldots, p-1\}$ to Bob.

- **Key generation:** $\text{gen}(1^k) = ((g, p, \alpha), (g, p, \beta))$.
  Bob picks a prime p and a primitive root g modulo p. He also chooses a random exponent $\alpha \in \{0, \ldots, p-2\}$ and computes $\beta = g^\alpha \bmod p$. Bob's public key is (g, p, β) and secret key is (g, p, α).

- **Encryption:** $\text{enc}(m, (g, p, \beta)) = (c_1, c_2)$.
  Alice chooses a random exponent $r \in \{1, \ldots, p-2\}$, and computes $c_1 = g^r \bmod p$ and $c_2 = \beta^r m \bmod p$. The ciphertext is (c_1, c_2)

- **Decryption:** $\text{dec}((c_1, c_2), (g, p, \alpha)) = m$.
  Bob can reconstructed the plaintext as $m = c_1^{-\alpha} c_2 \bmod p$

The ElGamal encryption scheme is a probabilistic encryption scheme [111]. In addition, it can be proved that the semantical security of the ElGamal encryption is equivalent to the decision Diffie-Hellman problem [216].

3.2 Commitment Schemes

A commitment is a string c sent by a committer Peggy to a receiver Vic to commit to a message m. A commitment scheme enables Peggy to commit to m while keeping it secret. Later on, Peggy can open c by providing an additional information. It is guaranteed that after committing to m, the value m cannot be changed.

There are many applications for commitment schemes. Sealed-bid auctions are one obvious example: the committed value represents the committer's bid. Commitment schemes are useful for identification schemes [90, 203], multiparty protocols [104], and are an essential component of many zero-knowledge schemes [27, 64, 105]. Goldreich, Micali, and Wigderson use them to construct zero-knowledge proofs for all languages in NP [105]. Ben-Or et al. extends this result to the larger class of all languages in IP [10].

In a commitment scheme, there are two participants, the committer and the receiver. Overall, a commitment scheme consists of two phases.

- Commit: The committer sends the message m he wants to commit to, in encrypted form c, to the receiver. Let $f : \{0,1\}^{|m|} \times X \to Y$ be a function, where X and Y are finite sets. Often the commitment c is any value $f(m, w), w \in X$.

- Reveal (or open): The committer opens the commitment c by sending an opening string (m, w) to the receiver.

A commitment scheme must satisfy the following properties.

- Correctness: If both the committer and the receiver follow the protocol, the receiver will always recover the message m.
- Hiding: No matter what the receiver does, he cannot learn anything about the message m. This is very similar to what we have seen before in encryption schemes.
- Binding: The committer cannot open c to different messages after the commit phase, that is, the committer cannot find legal opening strings (m, w) and (m', w').

If the hiding or the binding property depends on any assumption about computational complexity, we refer to computational hiding or computational binding. If the hiding or the binding property does not depend on any assumption about computational complexity, we refer to unconditional hiding or unconditional binding.

Commitment schemes that are unconditionally hiding and computationally binding have been proposed by many researchers, including Blum [14], Brassard, Chaum, and Crépeau [27], Brassard, Crépeau, and Yung [30], Halevi and Micali [115], and Halevi [114]. Brassard and Yung use "one-way group actions" to develop a very general framework and theory for all bit commitments having unconditional hiding [31]. Damgård, Pedersen, and Pfitzmann show that the existence of statistically hiding bit commitment schemes (which provide nearly perfect unconditional hiding) is equivalent to the existence of fail-stop signature schemes [66]. Ostrovsky, Venkatesan, and Yung investigate the feasibility of bit commitment when one of the committer/receiver is computationally unbounded, and in particular show that the existence of unconditionally hiding bit commitment is equivalent to the existence of oblivious transfer between a computationally bounded and a computationally unbounded party [176]. This implies that bit commitment that is unconditionally hiding and computationally binding is "as hard" as any other protocol because oblivious transfer is complete [129]. We remark that oblivious transfer (OT) is a two-party protocol introduced by Rabin [188]. Rabin's oblivious transfer assumes that the sender Alice possesses a value x, after the transfer the receiver Bob gets x with probability $1/2$ and Bob knows whether or not he got x. Alice does not know whether Bob gets x. A similar notion of 1-2-OT is introduced by [86]. In 1-2-OT, Alice has two bits b_0 and b_1 and Bob has a selection

bit i. After the transfer, Bob obtains only b_i, while Alice does not know the value of i. Equivalently, Bob may obtain a random bit in $\{b_0, b_1\}$, or the protocol can be played on strings rather than bits. Further, there are many other flavors of oblivious transfer [29, 60, 61, 86, 129] all of which are shown to be information-theoretically equivalent. That is, given any one of these protocols, one can implement the other ones. Thus, by "oblivious transfer" we can denote any one of them.

On the other hand, commitment schemes that are computationally hiding and unconditionally binding have also been studied by many authors, such as Brassard, Chaum, and Crépeau [27], Naor [168], Ohta, Okamoto, and Fujioka [173], and Ostrovsky, Venkatesan, and Yung [176]. Naor [168] presents a bit commitment protocol that is computationally hiding and unconditionally binding, using any one-way function; when both parties are computationally bounded, this is the best possible because such a protocol implies a one-way function [121]. Ostrovsky, Venkatesan, and Yung show that when the committer is computationally unbounded, a commitment scheme may be based on any hard-on-average problem in PSPACE [176].

It may be desirable that a commitment scheme be unconditionally hiding and unconditionally binding. However, this is impossible as the following discussion shows [68]. Suppose that $f : \{0,1\}^{|m|} \times X \to Y$ defines a scheme with both unconditional hiding and unconditional binding. Then when Peggy sends a commitment $c = f(m, w)$ to Vic, there must exist a w' such that $c = f(m', w')$. Otherwise, computationally unbounded Vic could compute (m, w), contradicting the unconditionally hiding property. However, if Peggy is also computationally unbounded, then she can also find (m', w') and open the commitment as m', thus contradicting the unconditionally binding property. We remark that this reasoning follows from the basic reason that the normal commitment scenario (two-party with noiseless channel) ensures each party sees everything the other party sends. There are several scenarios, however, in which the reasoning does not apply. In a distributed scenario with many parties, or in a two-party case where communication is noisy, it is no longer true that each party sees exactly what the other party sends. In such cases, unconditional hiding and binding can in fact be obtained simultaneously. For commitment schemes in such scenarios, see e.g. [11, 42, 61, 62, 65]. In addition, some researchers have explored bit commitment in models of quantum computation. Brassard et al. [28] propose a

quantum bit commitment scheme, but a subtle flaw is discovered; Mayers [156] proves secure quantum bit commitment to be impossible, as do Lo and Chau [142]. We note that despite the fact that the reasoning does not apply to quantum communication either, bit commitment with unconditional security is not possible with quantum communication alone. Salvail shows that under certain restricted assumptions about the committer's ability to make measurements, quantum bit commitment is still possible [198]. More discussions about quantum cryptography for two-party computation can be found in [199].

3.2.1 A Bit Commitment Scheme

A bit commitment scheme allows Peggy to commit to a single bit to Vic. In the scheme described below, the hiding property depends on the quadratic residuosity assumption, while the binding property is unconditional. The present scheme can be found in [27]. As usual, let $\mathrm{QR}_n = \{a \in \mathbb{Z}_n^* | a \text{ is a quadratic residue modulo } n\}$, $\mathrm{QNR}_n = \{a \in \mathbb{Z}_n^* | a \text{ is a quadratic nonresidue modulo } n\}$, and $J_n^{+1} = \{a \in \mathbb{Z}_n^* | (\frac{a}{n}) = 1\}$ where $(\frac{a}{n})$ is the Jacobi symbol of a mod n. Suppose Peggy would like to commit to a bit b.

- System setup: Peggy chooses $n = pq$, where p and q are primes, and $u \in J_n^{+1} \cap \mathrm{QNR}_n$. The integer n and u are public, and the factorization $n = pq$ is known only to Peggy.
- Commit: Peggy chooses $r \in \mathbb{Z}_n^*$ at random, computes $c = r^2 u^b \bmod n$, and sends c to Vic.
- Reveal: Peggy sends b and r to Vic. Then Vic can verify whether $c = r^2 u^b \bmod n$.

Let us think about the computational hiding property. The commitment is a Goldwasser-Micali probabilistic encryption [111] of 0 or of 1, and it reveals no information about the plaintext value b by the quadratic residuosity assumption. Consider the unconditional binding property. Let us suppose the property does not hold. Then $r_1^2 u \equiv r_2^2 \pmod{n}$ for some $r_1, r_2 \in \mathbb{Z}_n^*$. But then

$$u \equiv (r_2 r_1^{-1})^2 \pmod{n},$$

which is a contradiction because $u \in J_n^{+1} \cap \text{QNR}_n$.

3.2.2 A Number Commitment Scheme

We now give a commitment scheme that enable Peggy to commit to a message $m \in \mathbb{Z}_q$. The scheme described below is similar to those in [69, 95]. In the scheme, the hiding property is unconditional, while the binding property depends on the discrete logarithm assumption. Suppose that Peggy would like to commit to $m \in \mathbb{Z}_q$ to Vic.

- System setup: Vic picks large primes p, q such that $q \mid (p-1)$. Then Vic randomly chooses two numbers $g_1, g_2 \in \mathbb{Z}_p^*$ of order q. Vic sends p, q, g_1 and g_2 to Peggy.
- Commit: Peggy checks that p and q are primes, that q divides $p-1$, and that $g_1, g_2 \in \mathbb{Z}_p^*$ are elements of order q. Then she chooses a random exponent $r \in \{0, \ldots, q-1\}$, computes $c = g_1^r g_2^m \bmod p$, and sends c to Vic.
- Reveal: Peggy sends m and r to Vic. Vic verifies that $c = g_1^r g_2^m \bmod p$.

To obtain an element $a \in \mathbb{Z}_p^*$ of order q, Vic selects elements $b \in \mathbb{Z}_p^*$ at random until he obtains $a = b^{(p-1)/q} \bmod p \neq 1$. Then a has order q. Let G_q be the subgroup of order q in $\mathbb{Z}_p^*$. By Fact 2.2.10(3), there is a unique subgroup G_q of order q in $\mathbb{Z}_p^*$, G_q is cyclic, and each element $a \in \mathbb{Z}_p^*$ of order q is a generator of G_q. So g_1 and g_2 are generators of G_q.

Let us think about the unconditional hiding property. As $r \in_R \{0, \ldots, q-1\}$, g_1^r is a uniformly chosen random element in G_q, perfectly hiding g_2^m and thus m in $g_1^r g_2^m$. Hence, the unconditional hiding property holds. Consider the computational binding property. Suppose that it does not hold. Then $g_1^r g_2^m \equiv g_1^{r'} g_2^{m'} \pmod{n}$ for some $m, r, m', r' \in \mathbb{Z}_q$, where $m \neq m'$. So $\log_{g_1}(g_2) = (r - r')/(m' - m)$. Thus Peggy can compute $\log_{g_1}(g_2)$ of a randomly chosen element $g_2 \in G_q$, contradicting the discrete logarithm assumption. Note that Vic has no advantage if he knows $\log_{g_1}(g_2)$.

3.3 Identification Protocols

An identification protocol allows a prover Peggy to convince a verifier Vic of her identity. A goal of an identification scheme is that someone listening in as Peggy identifies herself to Vic should not subsequently be able to misrepresent herself as Peggy. Furthermore, it is desirable to guard against the possibility that Vic himself tries to impersonate Peggy after Peggy has identified herself. We will see that efficient implementations of such schemes exist.

Zero-knowledge identification protocols provide a mechanism to achieve the desirable properties. In general, Peggy has a secret key only known to her and a public key known to Vic. Thus, if some person can prove to Bob that he knows Peggy's secret key corresponding to Peggy's public key, then Bob can conclude that this person must be Peggy.

Informally, an identification protocol consists of a key generation algorithm gen and an interactive protocol (P, V) for a prover Peggy and a verifier Vic.

- Key generation algorithm gen: This is a probabilistic polynomial-time algorithm $\text{gen}(1^k) = (sk, pk)$. It takes as input a security parameter 1^k and outputs a pair (sk, pk) of secret and public keys for the prover, each of size $O(k^a)$ for $a \in \mathbb{N}$ a constant.

- Interactive protocol (P, V): The protocol (P, V) is an interactive proof of knowledge for sk corresponding to pk. We requires that it be complete, valid, and zero-knowledge (or witness-indistinguishable).

3.3.1 The Schnorr Identification Protocol

The Schnorr identification protocol [203] is one of the most attractive practical identification scheme. It is a typical three-round (commit-challenge-response) interactive proof of knowledge. P first commits to a value. V then challenges one of two things: either the commitment is of right form or P knows the witness. P then responds to such challenge, but reveals no information about the witness. Finally, V verifies whether the response is correct. We now describe the Schnorr identification protocol. Assume that Peggy wants to convince Vic of her identity.

- **Key generation:** $\text{gen}(1^k) = (\alpha, (g, p, q, \beta))$.
  Peggy picks large primes p, q such that $q \mid (p-1)$ and a number $g \in \mathbb{Z}_p^*$ of order q. She also chooses a random exponent $\alpha \in \{0, \ldots, q-1\}$ and computes $\beta = g^\alpha \bmod p$. Peggy's secret key is α and public key is (g, p, q, β).

- **Interactive protocol:** $\langle P(\alpha), V\rangle(g, p, q, \beta) = accept/reject$.

  1. Peggy chooses a random number $r \in \{1, 2, \ldots, q-1\}$. She computes $t = g^r \bmod p$, and sends t to Vic.
  2. Vic chooses a random number $c \in \{0,1\}^k$, where $1 \leq c \leq 2^k < q$. He sends c to Peggy.
  3. Peggy checks $1 \leq c \leq 2^k$ and sends $s = r - c\alpha \bmod q$ to Vic.
  4. Vic accepts Peggy's identity if $t = g^s \beta^c \bmod p$.

The Schnorr protocol allows Peggy to prove that she knows the discrete logarithm of her public key. The security of the protocol is based on the assumed intractability of the discrete logarithm problem. It can be shown that the protocol is honest-verifier zero-knowledge proof of knowledge. We discuss the protocol's properties in detail in the next subsection.

3.3.2 Analysis of the Schnorr Identification Protocol

The Schnorr protocol is based on an initial idea of Chaum et al. [43]. Chaum et al. have shown that the variant with $k = 1$ and $q = p - 1$ is a zero-knowledge proof of knowledge when sequentially repeated $\log_2(p)$ times. Nevertheless, in [203] it is shown that the Schnorr identification scheme is a proof of knowledge, but not zero knowledge. We will analyze the Schnorr identification protocol with respect to several properties, including proof of knowledge, zero knowledge, witness-indistinguishability, and witness-hiding.

Let $\mathcal{G}$ be a family of groups such that computing discrete logarithms in them is infeasible. The binary relation R underlying the protocol is the set $\{((p, g, q, \beta), \alpha) | p, q$ primes, $q \mid (p-1)$, $g \in \mathbb{Z}_p^*$ of order q, $\beta = g^\alpha \bmod p$ with $0 \leq \alpha < q, \langle g\rangle \in \mathcal{G}\}$.

Lemma 3.3.1. *The Schnorr identification protocol is a proof of knowledge for* $k = \Theta(\text{poly}(\ell))$ *where* ℓ *denotes the length of input* $(\approx 4\log_2(p))$.

Proof. Completeness. It can easily be seen that P can always convince V.

Validity. We construct a knowledge extractor K. Let

$$\delta = Pr[\langle P^*(\alpha), V\rangle(p, g, q, \beta) = accept].$$

Consider the following algorithm for a knowledge extractor K that has oracle access to the (dishonest) prover P^*.

1. Run P^* to obtain t and s using a randomly chosen $c \in \{0,1\}^k$. Proceed if the triple (t, c, s) is accepting, otherwise output $\perp$ and stop.

2. Reset and run P^* repeatedly with a randomly chosen $\tilde{c} \in \{0,1\}^k$ until an accepting triple $(t, \tilde{c}, \tilde{s})$ is found. If $\tilde{c} \neq c$ proceed to Step 3, otherwise output $\perp$ and stop.

3. Output $\alpha = \frac{\tilde{s}-s}{c-\tilde{c}} \bmod q$.

Let p_t denote the probability that P^* outputs a commitment t and let δ_t denote the probability that then, on input of a random $c \in_R \{0,1\}^k$, P^* outputs an s such that (t, c, s) is an accepting triple. Then we have $\delta = \sum_t p_t \delta_t$.

First we show that the expected running time of K is polynomial in the length of the input. Recall that verifying whether a triple is accepting requires $O(\ell^3)$ steps (modular exponentiation computations). Consider a particular t. The probability that K stops in the first step is $1 - \delta_t$ and the running time is $O(\ell^3)$ (note that a call to the oracle P^* counts as one step). In Steps 2 and 3, we have an expected running time of $(1/\delta_t)O(\ell^3)$. Because Step 2 is only entered with probability δ_t, the total expected running time given a particular t is

$$(1 - \delta_t)O(\ell^3) + \delta_t \frac{1}{\delta_t} O(\ell^3) = (2 - \delta_t)O(\ell^3).$$

Because a particular t gets chosen with probability p_t, the expected running time of the knowledge extractor K is

$$\sum_t p_t(2 - \delta_t)O(\ell^3) = (2 - \delta)O(\ell^3),$$

which is polynomial in the length of the input as required.

What remains to show is that for every positive polynomial Q and all sufficiently large ℓ,

$$Pr[((p,g,q,\beta),K^{P^*(p,g,q,\beta)}) \in R] \geq Pr[\langle P^*(\alpha),V\rangle(p,g,q,\beta) = accept] - \frac{1}{Q(\ell)},$$

where $\Pr[((p,g,q,\beta),K^{P^*(p,g,q,y)}) \in R]$ is the probability that the extractor K will output a witness and not the special symbol $\perp$, and $\Pr[\langle P^*(\alpha),V\rangle(p,g,q,\beta) = accept] = \delta$.

Consider the probability that in the second step, a triple with $\tilde{c} \neq c$ is found is

$$\frac{2^k\delta_t - 1}{2^k\delta_t} = 1 - \frac{1}{2^k\delta_t},$$

because we have $\delta_t 2^k > 1$ accepting triples in Step 2 and one of which we cannot use. Again, the probability that Step 2 is entered is δ_t, and thus the total success probability is

$$Pr[((p,g,q,\beta),K^{P^*(p,g,q,\beta)}) \in R] = \sum_t p_t\delta_t(1 - \frac{1}{2^k\delta_t}) = \delta - \frac{1}{2^k}.$$

For every positive polynomial Q and all sufficiently large ℓ, the probability is at least $\delta - 1/Q(\ell)$ if $k = \Theta(\text{poly}(\ell))$ holds. Hence we have constructed a knowledge extractor. □

As stated in [203] it is not zero-knowledge when k is selected as in Lemma 3.3.1. This is because k is too large such that the success probability of the simulator is negligible. If a smaller k is chosen, namely $k = O(\log_2(\ell))$, it is zero-knowledge. The following algorithm is a simulator for the output of any verifier V^*.

1. Choose $\tilde{c} \in_R \{0,1\}^k$.
2. choose $\tilde{s} \in_R \mathbb{Z}_q$.
3. Compute $\tilde{t} = g^{\tilde{s}}\beta^{\tilde{c}}$.
4. Run V^* using the computed $\tilde{t}$ and receive a c.
5. If c equals $\tilde{c}$, send $\tilde{s}$ to V^* and output V^*'s output and stop. Otherwise, continue with Step 1.

By construction, the output of the simulator is identically distributed to the output of the verifier. To conclude that the protocol is zero-knowledge, we must show that the expected running time is polynomial in ℓ. Because for the simulator all possible choices of $\tilde{c}$ are equally likely, the probability that in Step 5 the variable c will equal $\tilde{c}$ (and thus the simulator will stop) is 2^{-k}. Hence the expected running time of the simulator is $O(2^k)$. Therefore, we have to choose $k = O(\log(\ell))$ so that the simulator's expected running time is polynomial in ℓ. However, for such a choice the Schnorr identification scheme is no longer a proof of knowledge because this would require $k = \Theta(\text{poly}(\ell))$.

The following lemma says that we can obtain both properties (proof of knowledge and zero knowledge) at the same time by repeating the protocol in a sequential manner.

Lemma 3.3.2. *The Schnorr identification protocol is a proof of knowledge and is perfect zero-knowledge for $k = O(\log_2(\ell))$ when sequentially repeated $\Theta(\text{poly}(\ell))$ times.*

The Schnorr identification protocol is not known to be witness-hiding but is trivially witness-indistinguishable because there exists only one witness for each instance. In [174], Okamoto presents a variant which uses two different bases and thus two secret exponents for a public key. Then there exist q different witnesses for each instance. This variant is witness-indistinguishable and witness-hiding.

3.4 Digital Signature Schemes

The concept of a digital signature is put forth by Diffie and Hellman [78]. Analogous to a handwritten signature, a digital signature is a binary string that relates a message to the signer's identity. In general, each user holds a secret signing key x and publishes a corresponding verification key y so that only the user with x can sign a message m as σ and every one with y can verify whether σ is the signature of m by the user.

A digital signature scheme is a triple of algorithms (gen, sig, ver).

- Key generation algorithm gen: This is a probabilistic polynomial-time algorithm $\text{gen}(1^k) = (sk, pk)$. It takes as input a security parameter 1^k and outputs a pair (sk, pk) of signing and verification keys, each of size $O(k^a)$ for $a \in \mathbb{N}$ a constant.

- Signing algorithm sig: This is often a probabilistic polynomial-time algorithm $\text{sig}(m, sk) = \sigma$. It takes as input a signing key sk and a message m and outputs a signature σ of m.

- Verification algorithm ver: This a deterministic polynomial-time algorithm

$$\text{ver}(m, \sigma, pk) = accept/reject.$$

  It take as input a message m, a signature σ, and the verification key pk, and outputs "accept" if and only if σ is the signature of m using the signing key sk, i.e.,

$$\text{ver}(m, \text{sig}(m, sk), pk) = accept.$$

We may define the level of security of a signature scheme by the level of success of an adversary performing a certain type of attack. The level of success of an adversary may be described in increasing order as follows [112].

- Existential forgery: An adversary is able to compute a signature of at least one arbitrary message.

- Selective forgery: An adversary is able to compute a signature of a particular message chosen a priori. The adversary might have little or no control over the message whose signature is obtained.

- Total break: An adversary is able to compute the secret key of the signer or find an efficient signing algorithm functionally equivalent to the valid signing algorithm.

There are different types of attacks on signature schemes [112].

- Key-only attack: An adversary knows only the signer's verification key.

- Known-message attack: An adversary has seen signatures for a set of messages that are not chosen by him.

- Non-adaptive chosen-message attack: An adversary obtains valid signatures from a chosen list of messages before attempting to break the signature scheme. This attack is non-adaptive in the sense that messages are chosen before any signatures are seen.

- Adaptive chosen-message attack: An adversary is allowed to use the signer as an oracle at any time. The adversary can adaptively choose messages (by depending on the signer's public key and previously obtained signatures or messages) and obtain the corresponding signatures from the signer.

We say that a digital signature scheme is existentially unforgeable against the adaptively chosen-message attack if an adversary, when given oracle access to the signing algorithm, cannot forge valid signatures for messages that the adversary did not yet ask the oracle. We make this precise as follows.

Definition 3.4.1. *A digital signature scheme (gen, sig, ver) is existentially unforgeable against the adaptively chosen-message attack if for any probabilistic polynomial-time algorithm A, every positive polynomial P, and all sufficiently large k, it holds that*

$$Pr[ver(m, \sigma, pk) = accept \wedge m \notin Q : (sk, pk) \leftarrow gen(1^k), \\ (m, \sigma) \leftarrow A^{sig}(pk)] < \frac{1}{P(k)},$$

where Q is the set of messages that the adversary has queried to the signing algorithm.

This is the strongest and reasonable security requirement. Often a signature scheme is said to be secure if it is existentially unforgeable, even under the adaptively chosen-message attack. Goldwasser, Micali and Rivest are the first to propose a signature scheme that is secure against the adaptively chosen-message attack, assuming the existence of claw-free trapdoor permutations [112].

There are two types of signature schemes. One is custom designed, such as the RSA and the ElGaml signature schemes. The other is derived from an identification protocol, such as the Feige-Fiat-Shamir and the Schnorr signature schemes.

To convert an identification protocol into a signature scheme, we use a hash function to replace the verifier in the identification scheme so that the interaction is not necessary. This technique is introduced in [90] and has been formalized by Bellare and Rogaway [9] as the so-called random oracle model. In this model, the hash function is replaced by an oracle (random function). For such constructions, it is often argued that a signature scheme is secure if the underlying identification protocol is an honest-verifier zero-knowledge proof of knowledge in the random oracle model.

Pointcheval and Stern show that in the random oracle model the Schnorr signature scheme and a variant of the ElGamal signature scheme are secure against the adaptively chosen-message attack [183, 184]. They also state that every signature scheme obtained from an honest-verifier zero-knowledge proof of knowledge is secure against existential forgery under the adaptively chosen-message attack.

3.4.1 The RSA signature scheme

The RSA signature scheme is directly derived from the RSA encryption scheme by reversing the roles of encryption and decryption [195].

This signature scheme works as follows. Assume Alice wants to sign a message m to Bob. Let h be a publicly known collision-resistant hash function that maps $\{0,1\}^*$ to $\mathbb{Z}_n$.

- **Key generation:** $\text{gen}(1^k) = ((d,n),(e,n))$.
  Alice picks randomly and independently two large primes p and q, and compute $n = pq$. She also chooses an integer e such that $1 < e < (p-1)(q-1)$ and $\gcd(e,(p-1)(q-1)) = 1$. Then Alice computes an integer d such that $1 < d < (p-1)(q-1)$ and $de \equiv 1 \pmod{(p-1)(q-1)}$. Because $\gcd(e,(p-1)(q-1)) = 1$, such a number d exits. Alice's public key is (e,n) and secret key is (d,n).

- **Signing:** $\text{sig}(m,(d,n)) = \sigma$.
  Alice signs the message m by computing $\sigma = h(m)^d \bmod n$. The signature is σ.

- **Verification:** $\text{ver}(m,\sigma,(e,n)) = accept/reject$.
  Bob wants to verify the signature σ. He checks if $h(m) \stackrel{?}{=} \sigma^e \bmod n$. If this equation holds, then he accepts the signature; otherwise, he rejects it.

3.4.2 The ElGamal Signature Scheme

The signature scheme is proposed by ElGamal together with his public-key encryption scheme [84, 85].

This signature scheme works as follows. Assume Alice wants to sign a message m to Bob. Let h be a publicly known collision-resistant hash function that maps $\{0,1\}^*$ to $\mathbb{Z}_p$.

- **Key generation:** $\text{gen}(1^k) = ((g, p, \alpha), (g, p, \beta))$.
  Alice picks a prime p and a primitive root g modulo p. She also chooses a random exponent $\alpha \in \{0, \ldots, p-2\}$ and computes $\beta = g^\alpha \bmod p$. Alice's public key is (g, p, β) and secret key is (g, p, α).

- **Signing:** $\text{sig}(m, (g, p, \alpha)) = (s_1, s_2)$.
  Alice chooses a random number $r \in \{1, 2, \ldots, p-2\}$ that is prime to $p-1$. She computes $s_1 = g^r \bmod p$ and $s_2 = (h(m) - \alpha s_1) r^{-1} \bmod (p-1)$ where r^{-1} is the inverse of r modulo $p-1$. The signature is (s_1, s_2).

- **Verification:** $\text{ver}(m, (s_1, s_2), (g, p, \beta)) = accept/reject$.
  For $s_1 \in \mathbb{Z}_p^*$ and $s_2 \in \mathbb{Z}_{p-1}$, Bob checks if $\beta^{s_1} s_1^{s_2} \stackrel{?}{\equiv} g^{h(m)} \pmod p$. If this congruence holds, then he accepts the signature; otherwise, he rejects it.

The signature scheme is a randomized signature mechanism. Its security is based on the discrete logarithm assumption. Moreover, the random value r should be chosen differently each time a message is signed; otherwise, the secret key could be computed from two such signatures. For a more detailed analysis of the security we refer the reader to [3, 13, 85, 141, 157, 183].

A variant of the ElGamal signature scheme, called digital signature algorithm (DSA) [93], is proposed as a standard by the U.S. National Institute of Standards and Technology (NIST). The digital signature standard (DSS) is the first digital signature scheme recognized by any government.

3.4.3 The Schnorr Signature Scheme

The Schnorr signature scheme is an example of the construction of a signature scheme from an identification protocol [203]. Compared with the ElGamal signature scheme, the Schnorr scheme provides shorter signatures for the same level of security.

This signature scheme works as follows. Assume Alice wants to sign a message m to Bob. Let h be a publicly known collision-resistant hash function that maps $\{0,1\}^*$ to $\mathbb{Z}_q$ and let $\|$ denote the concatenation of two strings.

- **Key generation:** $\text{gen}(1^k) = ((g, p, q, \alpha), (g, p, q, \beta))$.
  Alice picks large primes p, q such that $q \mid (p-1)$ and a number $g \in \mathbb{Z}_p^*$ of

order q. She also chooses a random exponent $\alpha \in \{0, \ldots, q-1\}$ and computes $\beta = g^\alpha \bmod p$. Alice's public key is (g, p, q, β) and secret key is (g, p, q, α).

- **Signing:** $\mathrm{sig}(m, (g, p, q, \alpha)) = (c, s)$.
  Alice chooses a random number $r \in \{1, 2, \ldots, q-1\}$. She computes $c = h(m \parallel g^r)$ and $s = (r - c\alpha) \bmod q$. The signature σ is (c, s).

- **Verification:** $\mathrm{ver}((c, s), (g, p, q, \beta)) = accept/reject$.
  For $c, s \in \mathbb{Z}_{q-1}$, Bob checks if $c \stackrel{?}{=} h(m \parallel g^s \beta^c \bmod p)$. If the equation holds, then he accepts the signature; otherwise, he rejects it.

3.4.4 Signatures of Knowledge

Using the techniques introduced in [87, 90], every three-round proof of knowledge that is honest-verifier zero-knowledge can be turned into a signature scheme by replacing the verifier with a hash function. We call the schemes "signatures based on proofs of knowledge", or "signatures of knowledge" for short. A signature of knowledge allows a signer to prove the knowledge of a secret with respect to some public information noninteractively. The signer can also tie his knowledge of a secret to a message being signed. In [184], it is proved that in the random oracle model all such signatures are simulatable and secure against existential forgery under adaptively chosen-message attacks. Simulatability means that the distribution of the strings that can be efficiently generated without knowledge of the secret signing key are indistinguishable from the distribution of the actual signatures. Note that though such signatures are simulatable, they are not deniable zero-knowledge [178].

All signatures of knowledge presented in this section have the corresponding three-round protocols similar to Schnorr's identification protocol. These protocols can be shown to be honest-verifier zero-knowledge proofs of knowledge. We first describe the algebraic setting used here. Let G be a finite cyclic group of prime order q, k an integer, and let $g, h, g_1, g_2, \ldots, g_k \in G$ be generators of G such that computing discrete logarithms of any group element with respect to any one of the generators is infeasible. Furthermore, the generators are chosen in a random manner, such that none of the discrete logarithms of any generator with respect to another is known.

Then the computation of a representation of a group element with respect to multiple generators is as hard as the discrete logarithm problem.

We introduce our notation about signatures of knowledge. Suppose the signer (prover) chooses $\alpha, \beta \in_R \mathbb{Z}_q^*$ as secret keys and compute her public keys $y_1 = g^\alpha$ and $y_2 = g^\beta h^\alpha$. Let $\wedge$ denote the logical conjunction. An expression such as

$$sok[(\alpha, \beta) : y_1 = g^\alpha \wedge y_2 = g^\beta h^\alpha](m)$$

denotes a signature based on a proof of knowledge of secret keys α and β such that the statement to the right of the colon is true. This is equal to proving the knowledge of the discrete logarithm of y_1 to the base g and a representation of y_2 to the bases g and h and, in addition, that the h-part of this representation of y_2 equals the discrete logarithm of y_1 to the base g. The Greek letters denote the knowledge of the signer. The variable *sok* can be thought of as reference to the definition of a particular signature of knowledge. If the message is the null message, then the term (m) after the '] ' is omitted. In addition, we denote by $\|$ the concatenation of two strings and let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\ell$ be a collision-resistant hash function.

Signatures of Knowledge of a Discrete Logarithm

The signature proves the knowledge of the discrete logarithm of a public key y to the base g.

Definition 3.4.2. *A pair* $(c, s) \in \{0,1\}^\ell \times \mathbb{Z}_q$ *satisfying*

$$c = \mathcal{H}(m \parallel g \parallel y \parallel g^s y^c)$$

is a signature of knowledge of the discrete logarithm of a group element y *to the base* g *of the message* $m \in \{0,1\}^*$ *and is denoted by* $SKDL[\,\alpha : y = g^\alpha\,](m)$.

Basically, $SKDL[\,\alpha : y = g^\alpha\,](m)$ is the Schnorr signature with a slightly different argument to the hash function. If the value $\alpha = \log_g(y)$ is known, such a signature can be computed by choosing a random integer $r \in \mathbb{Z}_q$ and computing $t = g^r$ and then c and s according to

$$c = \mathcal{H}(m \parallel g \parallel y \parallel g^r),$$
$$s = r - c\alpha \bmod q.$$

Anyone can verify (c, s) by checking $c \stackrel{?}{=} \mathcal{H}(m \parallel g \parallel y \parallel g^s y^c)$.

Although the signature of knowledge is not interactive, it is reasonable to call t the commitment, c the challenge, and s the response.

Signatures of Knowledge of a Representation

The signature proves the knowledge of a representation of a public key. The corresponding proof systems are first introduced in [43].

Definition 3.4.3. *A $(k+1)$-tuple $(c, s_1, \ldots, s_k) \in \{0,1\}^\ell \times (\mathbb{Z}_q)^k$ satisfying*

$$c = \mathcal{H}(m \parallel g_1 \parallel \ldots \parallel g_k \parallel y \parallel y^c \prod_{i=1}^{k} g_i^{s_i})$$

is a signature of knowledge of a representation of a element y with respect to the bases $g_1, \ldots, g_k$ of the message $m \in \{0,1\}^*$. It is denoted by

$$SKAREP[(\alpha_1, \ldots, \alpha_k) : y = \prod_{i=1}^{k} g_i^{\alpha_i}](m).$$

If values $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}_q$ are known such that $y = \prod_{i=1}^{k} g_i^{\alpha_i}$ holds, such a signature can be computed as follows. The signer chooses the integers $r_1, \ldots, r_k$ at random from $\mathbb{Z}_q$ and computes

$$c = \mathcal{H}(m \parallel g_1 \parallel \cdots \parallel g_k \parallel y \parallel y^c \prod_{i=1}^{k} g_i^{r_i}),$$
$$s_i = r_i - c\alpha_i \bmod q, \text{ for } i = 1, \ldots, k.$$

Anyone can verify the signature by checking $c \stackrel{?}{=} \mathcal{H}(m \parallel g_1 \parallel \ldots \parallel g_k \parallel y \parallel y^c \prod_{i=1}^{k} g_i^{s_i})$.

Because a public key y_i can be formed using only a subset of the generators $g_1, \ldots, g_k$, later we use a set $\mathcal{J}_i \subseteq \{1, \ldots, k\}$ such that $y_i = \prod_{j \in \mathcal{J}_i} g_j^{\alpha_{i,j}}$ holds, where $\alpha_{i,j}$ are secret keys with respect to the public key y_i. Note that the secret keys $\alpha_{i,j}$ are in general not numbered consecutively but take a tuple (i, j), where i is the index of the public key y_i and j is the index of the respective generator g_j. In particular, the secret keys $\alpha_{i,j}$ for $j \notin \mathcal{J}_i$ are not defined.

Signatures of Knowledge of Equality of the Discrete Logarithms

The signature proves that the discrete logarithms of two public keys with respect to two different bases are equal. Such a scheme is introduced in [45].

Definition 3.4.4. *A pair* $(c, s) \in \{0,1\}^{\ell} \times \mathbb{Z}_q$ *satisfying*

$$c = \mathcal{H}(m \parallel g \parallel h \parallel y \parallel z \parallel g^s y^c \parallel h^s z^c)$$

is a signature of the message $m \in \{0,1\}^*$ *based on a proof of knowledge and of equality of the discrete logarithm of* z *with respect to the base* h *and of the discrete logarithm of* y *with respect to the base* g*. It is denoted by*

$$SKEDL[\alpha : y = g^{\alpha} \wedge z = h^{\alpha}](m).$$

If $\alpha = \log_g(y)$ is known and if $\log_g(y) = \log_h(z)$ holds, such a signature (c, s) can be computed as follows. One chooses a random integer r from $\mathbb{Z}_q$ and computes $t_1 = g^r$ and $t_2 = h^r$. Then, c and s are calculated according to

$$c = \mathcal{H}(m \parallel g \parallel h \parallel y \parallel z \parallel t_1 \parallel t_2)$$

and

$$s = r - c\alpha \bmod q.$$

Anyone can verify the signature by checking $c \stackrel{?}{=} \mathcal{H}(m \parallel g \parallel h \parallel y \parallel z \parallel g^s y^c \parallel h^s z^c)$.

$SKEDL[\alpha : y = g^{\alpha} \wedge z = h^{\alpha}](m)$ can be seen as two parallel signatures $SKDL[\alpha : y = g^{\alpha}]$ and $SKDL[\alpha : z = h^{\alpha}]$, where the exponent for the commitments, the challenges, and the responses are the same. This technique can be generalized to signatures of knowledge of representations of several public keys: Whenever two elements of the representations are equal, the respective responses are the same by choosing the same exponent for the commitments and the same challenge. Signatures of knowledge of representations are described in the next subsection.

Signatures of Knowledge of Representations

The signature proves the knowledge of representations of several, say w, public keys at the same time. Of course, this can be done by computing w separate signatures

$SKAREP[(\alpha_{i,j})_{j\in\mathcal{J}_i} : y_i = \prod_{j\in\mathcal{J}_i} g_j^{\alpha_{i,j}}](m)$. However, it is possible to merge these signatures by using the same challenge for all of them and choosing the same exponent for two commitments whenever the respective elements of the representations are equal. Thus we can make the resulting signature shorter.

Definition 3.4.5. *A $(u+1)$-tuple $(c, s_1, \ldots, s_u) \in \{0,1\}^\ell \times (\mathbb{Z}_q)^u$ satisfying*

$$c = \mathcal{H}(m \parallel g_1 \parallel \ldots \parallel g_k \parallel y_1 \parallel \ldots \parallel y_w \parallel \mathcal{J}_1 \parallel \ldots \parallel \mathcal{J}_w \parallel \{e_{ij}\}_{i=1,\ldots,w;j\in\mathcal{J}_i} \parallel y_1^c \prod_{j\in\mathcal{J}_1} g_j^{s_{e_{1,j}}} \parallel \ldots \parallel y_w^c \prod_{j\in\mathcal{J}_w} g_j^{s_{e_{w,j}}})$$

is a signature of the message $m \in \{0,1\}^*$ based on a proof of knowledge of representations of $y_1, \ldots, y_w$ with respect to some of the bases $g_1, \ldots, g_k$ and that , additionally, some of the elements of the representations are equal. It is denoted by

$$SKREP\left[(\alpha_1, \ldots, \alpha_u) : (y_1 = \prod_{j\in\mathcal{J}_1} g_j^{\alpha_{e_{1j}}}) \wedge \cdots \wedge (y_w = \prod_{j\in\mathcal{J}_w} g_j^{\alpha_{e_{wj}}})\right](m),$$

where $e_{ij} \in \{1, \ldots, u\}$ index the elements $\alpha_1, \ldots, \alpha_u \in \mathbb{Z}_q$ and the elements of $\mathcal{J}_i$ index the base elements $g_1, \ldots, g_k$.

If a u-tuple $(\alpha_1, \ldots, \alpha_u)$ is known that satisfies the considered statement, such a signature can be computed as follows. One first chooses $r_i \in_R \mathbb{Z}_q$ for $i = 1, \ldots, u$, computes c as

$$c = \mathcal{H}(m \parallel g_1 \parallel \ldots \parallel g_k \parallel y_1 \parallel \ldots \parallel y_w \parallel \mathcal{J}_1 \parallel \ldots \parallel \mathcal{J}_w \parallel \{e_{ij}\}_{i=1,\ldots,w;j\in\mathcal{J}_i} \parallel \prod_{j\in\mathcal{J}_1} g_j^{r_{e_{1j}}} \parallel \ldots \parallel \prod_{j\in\mathcal{J}_w} g_j^{r_{e_{wj}}}), \tag{3.1}$$

and calculates

$$s_i = r_i - c\alpha_i \bmod q.$$

Anyone can verify the signature by checking

$$c \stackrel{?}{=} \mathcal{H}(m \parallel g_1 \parallel \ldots \parallel g_k \parallel y_1 \parallel \ldots \parallel y_w \parallel \mathcal{J}_1 \parallel \ldots \parallel \mathcal{J}_w \parallel \{e_{ij}\}_{i=1,\ldots,w;j\in\mathcal{J}_i} \parallel y_1^c \prod_{j\in\mathcal{J}_1} g_j^{s_{e_{1,j}}} \parallel \ldots \parallel y_w^c \prod_{j\in\mathcal{J}_w} g_j^{s_{e_{w,j}}}).$$

Example 3.4.1. Suppose we want to obtain a signature of knowledge that we know α_1, α_2, and α_3 such that, given y_1, y_2, y_3 and g_1, g_2, g_3, the following holds:

$$y_1 = g_1^{\alpha_1} g_3^{\alpha_2},$$
$$y_2 = g_2^{\alpha_1} g_3^{\alpha_3},$$
$$y_3 = g_1^{\alpha_1} g_2^{\alpha_3} g_3^{\alpha_2}.$$

The signature

$$SKREP\left[(\alpha_1, \alpha_2, \alpha_3) : y_1 = g_1^{\alpha_1} g_3^{\alpha_2} \wedge y_2 = g_2^{\alpha_1} g_3^{\alpha_3} \wedge y_3 = g_1^{\alpha_1} g_2^{\alpha_3} g_3^{\alpha_2}\right](m)$$

is a 4-tuple (c, s_1, s_2, s_3), where $e_{11} = 1, e_{12} = 2, e_{21} = 1, e_{22} = 3, e_{31} = 1, e_{32} = 3, e_{33} = 2$, $\mathcal{J}_1 = \{1, 3\}$, $\mathcal{J}_2 = \{2, 3\}$, and $\mathcal{J}_3 = \{1, 2, 3\}$ in Eq. (3.1).

Chapter 4

A Fully Public-Key Traitor-Tracing Scheme

A fully public-key traitor-tracing scheme is a public-key traitor-tracing scheme that allows a subscriber to choose his own private decryption key without others learning the key. In this chapter we propose such a scheme and discuss its efficiency and security.

4.1 Introduction

In an open broadcast network, a distributor transmits digital contents to a large number of users in such a way that only subscribers are authorized to extract the contents. Applications include such fee-based services as pay-per-view television and Web financial information channel. Clearly, anyone connected to the open network is able to pick up the data that flow through the broadcast channel, whether authorized or not. A straightforward solution to this problem is for the distributor to separately encrypt the contents with each subscriber's key before broadcasting the ciphertext. Now, only the subscribers have the corresponding private keys to decrypt the ciphertext. This issue of secure broadcasting is first addressed in [56]; however, the proposed method carries out n-times encryptions for one copy of data, where n is the number of subscribers. Later, broadcast encryption is proposed [12, 89, 127]. A broadcast-encryption scheme prevents nonsubscribers from extracting the contents. By properly

generating keys, the distributor can encrypt the contents with the encryption key derived from all subscribers' secret keys, and the subscribers have the decryption key to decrypt the ciphertext. The tradeoff between the bandwidth requirement and the keys' storage space is studied in [16, 17, 143, 212].

A broadcast-encryption scheme remains prone to the collusion attack. Some subscribers may collude to create new decryption keys, and the resulting pirate decoder allows nonsubscribers to extract the contents. To discourage subscribers to reveal their private keys, traitor tracing is initiated in [57, 58] and studied further in [96, 181, 211, 213]. The idea is an algorithm that uses the confiscated pirate decoder to track down at least one colluder without wrongly accusing noncolluders with high probability. Most of these schemes are so-called black-box traceable. This means that the pirate decoder can be queried on different inputs as an oracle but cannot be opened to reveal its private key. Most of the traitor-tracing schemes are secret-key systems. Although they can be founded on public keys, complex protocols may result [181]. More recent work in [19] proposes a public-key traitor-tracing scheme; furthermore, as long as the number of colluders is below some threshold, the tracing algorithm catches all and only traitors. The scheme has two disadvantages: It is only partially black-box traceable, and the secret keys of the subscribers are generated by a trusted center (the system is hence not fully public-key). We will present a traitor-tracing scheme with these following strong features: The tracing algorithm is black-box traceable, it tracks down all the colluders regardless of their size, and the subscribers generate their own secret keys (it is thus fully public-key).

Key management is a critical issue. Subscribers' keys are affected for at least two reasons. First, a key must be discarded if it is found to be pirated or if its user leaves the system. But then the remaining subscribers' keys may be subject to changes when even one user's key is discarded. Second, when a new subscriber joins the system, the existing subscribers' keys may need to be changed to prevent the new subscriber from decrypting the ciphertext received before the new user joins the system. Both scenarios become problematic if pirating is frequent or if the subscriber base is fluid. In order for the subscribers to easily manage their own keys, the system should minimize the need to regenerate subscribers' secret keys. Such a scheme is said to be long-lived. This attribute is studied in [97]. In that proposed scheme, some

subscribers may need to be rekeyed when a sufficient number of keys are discarded. In contrast, our scheme does not require the regeneration of the secret keys in the above two scenarios. It is therefore perfectly long-lived.

Anonymity is another critical issue for any traitor-tracing schemes because the promise of anonymity usually promotes subscription [127]. We list two cases which may compromise anonymity. When new subscribers join the service, their identities may be revealed because of their interaction with the distributor. Second, the broadcast contents themselves may disclose the subscribers' identities, which makes eavesdropping threaten the privacy of the subscribers. Our scheme solves both problems: Registering with the service is noninteractive, and analyzing the transmissions does not reveal the subscribers' identities. Thus, the subscribers' identities are not known to anyone except the distributor.

We now summarize the features of our traitor-tracing scheme. The traitor-tracing scheme is perfectly long-lived and achieves anonymity. It is a fully public-key system, without relying on a trusted center to generate the keys. The scheme is based on the following ideas. Each subscriber randomly selects a secret key to compute a number which is sent to the distributor. After the distributor receives the numbers from all the subscribers, it combines them to create a single encryption key. Using the ElGamal encryption scheme, digital contents are encrypted with the encryption key. Henceforth, each subscriber uses his or her own secret key to decrypt the ciphertext.

This chapter is organized as follows. In Section 4.2, basic terms are defined. Then in Section 4.3, useful facts in number theory are presented. Section 4.4 describes our scheme and discusses its security. Important attributes of our scheme are analyzed in Section 4.5. Conclusions are given in Section 4.6.

4.2 Key Terms

Broadcast encryption consists of three components: key generation, encryption, and decryption. It specifies the way to encrypt and decrypt the digital contents with the generated keys in a broadcast network. The important task facing broadcast encryption is to prevent nonsubscribers from decrypting the encrypted content. Like the point-to-point encryption scheme, a broadcast-encryption scheme can be secret-key

or public-key. Traitors are subscribers who allow nonsubscribers to extract the contents. When a broadcast-encryption scheme has the capability to track down traitors, it is said to be traitor-tracing. Traitor tracing involves a tracing algorithm, which uses the confiscated pirate decoder to track down traitors. The tracing algorithm is k-traceable if at least one of the traitors can be identified given any pirate decoder created by at most k traitors. The tracing algorithm is said to be black-box if the pirate decoder can only be queried as an oracle but not opened to reveal its internal key. The perfect long-livedness attribute means that rekeying the subscribers is unnecessary in the following two situations:

1. An existing key is discarded to prevent its future use.

2. New subscribers join the system without being able to decrypt earlier ciphertext.

The anonymity attribute means that the subscribers' identities are not known to anyone except the distributor. Our proposed traitor-tracing scheme will be perfectly long-lived and anonymous.

A public-key traitor-tracing scheme consists of four components [19]:

- **Key generation:** Given a security parameter 1^k and a number n, the key-generation algorithm (performed by the distributor or a trusted center) outputs a public encryption key e and n private decryption keys $d_1, d_2, \ldots, d_n$. Any decryption key d_i can be used to decrypt a ciphertext created with the public encryption key e.

- **Encryption:** Given a public key e and a message x, the encryption algorithm outputs a ciphertext C.

- **Decryption:** Given a ciphertext C and any decryption key d_i, the decryption algorithm outputs the message x. The algorithm itself is public. Only the decryption keys are secret as in the point-to-point cryptosystem.

- **Tracing:** Given a pirate decryption box $\mathcal{D}$, the tracing algorithm outputs at least one traitor if at most k of the decryption keys are involved in creating the box. The tracing algorithm may be black-box or otherwise.

A fully public-key traitor-tracing scheme strengthens the key-generation part as follows:

- **Key generation:** Given a security parameter 1^k, every subscriber generates his private decryption key d_i and sends the corresponding public information e_i to the distributor. Then the distributor computes the public encryption key e using all subscribers' e_i. Any decryption key d_i can be used to decrypt a ciphertext created with the public encryption key e.

The secret keys are now known only to their subscriber owners. This is clearly desirable.

4.3 Number-Theoretic Preliminaries

The present scheme is based on several number-theoretic results. In addition to those stated in Section 2.2, we prove the following lemma that is useful for the proposed scheme. As usual, let $\phi(n)$ denote Euler's phi function, which gives the number of positive integers $m \in \{1, 2, \ldots, n-1\}$ such that $\gcd(m, n) = 1$.

Lemma 4.3.1. *If q and $p = 2q + 1$ are both primes and a is a positive integer with $1 < a < p - 1$, then $-a^2$ is a quadratic nonresidue and a primitive root modulo p.*

Proof. There are $\phi(2q) = q - 1$ primitive roots of p, q quadratic residues of p, and q quadratic nonresidues of p. None of the q quadratic residues of p can be primitive roots. Furthermore, -1, a quadratic nonresidue, cannot be a primitive root either. Thus, the remaining $q - 1$ quadratic nonresidues of p must be all the primitive roots. The number $-a^2$ is a quadratic nonresidue of p because

$$\left(\frac{-a^2}{p}\right) = \left(\frac{-1}{p}\right)\left(\frac{a^2}{p}\right) = (-1) \times 1 = -1.$$

Hence $-a^2$ is a primitive root modulo p. □

Lemma 4.3.1 says that if the primes are chosen in the form $2q + 1$, where q is also a prime, then we can easily obtain primitive roots and quadratic nonresidues.

4.4 Proposed Scheme

Let k be a security parameter and n be the number of subscribers. Choose k-bit primes $p_i = 2q_i + 1$, where q_i are odd primes. For convenience, assume $p_1 < p_2 < \cdots < p_n$. The primes will be such that $q_1^{1/2}$ is large enough. Let $M = \prod_{i=1}^{n} p_i$. Assume the contents (plaintext) are elements of $\mathbb{Z}_{p_1}$. Let g be a common primitive root modulo each of the p_i's. By Lemma 4.3.1 such a g exists because any $-a^2$ is a valid candidate when $1 < a < p_1 - 1$. We now describe the four components of our scheme.

- **Key generation:** Each subscriber i chooses a private decryption key $d_i \in \mathbb{Z}_{p_i}$ randomly and sends (β_i, p_i) to the distributor, where $\beta_i = g^{d_i} \bmod p_i$. Next, the distributor computes $\beta = \sum_{i=1}^{n} \beta_i M_i y_i \bmod M$, where $M_i = M/p_i$ and $M_i y_i \equiv 1 \pmod{p_i}$, for $1 \le i \le n$. The triple (g, β, M) is the public encryption key. Note that
$$\beta \equiv \beta_i \pmod{p_i}.$$

- **Encryption:** Let the plaintext be $x \in \mathbb{Z}_{p_1}$. The distributor picks a random element r from $\{0, 1, \ldots, p_1 - 1\}$ and computes
$$z_1 = g^r \bmod M,$$
$$z_2 = x\beta^r \bmod M.$$
The ciphertext is $C = (z_1, z_2)$. Note that the digital content is encrypted only once not n times.

- **Decryption:** Given a ciphertext $C = (z_1, z_2)$, the decryption algorithm computes $z_2(z_1^{d_i})^{-1} \bmod p_i$. This step correctly yields the plaintext x because
$$\begin{aligned} z_2(z_1^{d_i})^{-1} &\equiv x\beta^r (g^{rd_i})^{-1} \\ &\equiv x\beta_i^r (g^{rd_i})^{-1} \\ &\equiv x(g^{d_i})^r (g^{rd_i})^{-1} \\ &\equiv x \pmod{p_i}. \end{aligned}$$

- **Tracing:** The tracing algorithm is described in Section 4.5.

The encryption and decryption algorithms are similar to those in the ElGamal cryptosystem. The content is encrypted once, and each subscriber can decrypt using his or her own decryption key. There is no need of a key generation center to generate the decryption keys. Instead, subscribers generate their own keys at random. The distributor creates the encryption key without knowing any decryption key. The decryption keys are therefore private to their respective owners. This is identical to the standard point-to-point public-key cryptosystem setup. Our scheme is hence fully public-key.

Consider this straightforward approach to secure broadcasting in a full public-key setting [56]: Before broadcasting, the plaintext is encrypted n times with all subscribers' public keys, and then the Chinese remainder algorithm is applied. If the straightforward approach uses the ElGamal probabilistic encryption, the number of modular multiplications and squarings in performing encryptions is n-times ours because we encrypt the plaintext only once. The fast modular multiplication algorithm devised in [32] requires $t + 7$ clock pulses, where t is the length of the modulus. Assume our encryption needs a total of ℓ modular multiplications and squarings. For each plaintext, our encryption needs time $\ell(nk+7)$, but the straightforward approach needs time $\ell n(k + 7)$ plus $O_B(M(nk)\log(n)) + O_B(nM(k)\log(k))$ for performing the Chinese remainder algorithm. Hence, our scheme is more efficient.

4.4.1 Security Analysis

We will show that our scheme is plaintext-secure against a passive generic adversary if $q_1^{1/2}$ is large enough. We now explain the terms. A passive generic adversary is a generic algorithm and can only eavesdrop the network. A generic algorithm does not exploit any special properties of the encodings of group elements except that each group element is encoded as a unique bit string [208]. An encryption system is plaintext-secure if the full plaintext about a content cannot be derived from its encryption form.

Let M' be a product of t primes p_i, say,

$$M' = p_{i_1} p_{i_2} \cdots p_{i_t}.$$

Let $\beta' = \beta \bmod M'$. As before, g is a common primitive root modulo p_i, $i =$

$1, 2, \ldots, n$. The Diffie-Hellman problem (DH) in a group with generator g is to compute $g^{r_1 r_2}$ from g^{r_1} and g^{r_2}. Shoup shows that any generic algorithm must perform $\Omega(q_{\max}^{1/2})$ group operations for the problem, where $q_{\max}$ is the largest prime dividing the order of the group [208]. Because g is a common primitive root modulo each of the p_is, the largest possible order of g modulo M' is $\lambda(M') = 2q_{i_1}q_{i_2}\cdots q_{i_t}$ by equation (2.1). So the subgroup H of $Z_{M'}^*$ generated by g has order $\lambda(M')$. The order of H surely contains a prime factor which is not smaller than q_1. As we choose a large $q_1^{1/2}$, the DH problem in H is intractable for a generic adversary. Based on this intractability, we next prove that our encryption scheme is plaintext-secure against a passive generic adversary. But first we need the following lemma.

Lemma 4.4.1. *If the ElGamal cryptosystem in $Z_{M'}^*$ can be broken, then the Diffie-Hellman problem in the subgroup H of $Z_{M'}^*$ generated by g can be solved efficiently.*

Proof. Suppose that there is an algorithm $\mathcal{A}$ that breaks the ElGamal cryptosystem in $Z_{M'}^*$. Given g, M', β', z_1 and z_2, algorithm $\mathcal{A}$ computes the plaintext

$$x = z_2(\beta'^{\log_g z_1})^{-1} \bmod M'.$$

Assume that $\beta', \gamma \in H$. When given inputs g, M', β' and γ for the Diffie-Hellman problem, $\mathcal{A}$ can be invoked to solve this DH problem by

$$\begin{aligned}\mathcal{A}(g, M', \beta', \gamma, 1)^{-1} &= ((\beta'^{\log_g \gamma})^{-1})^{-1} \bmod M' \\ &= g^{\log_g \beta' \log_g \gamma} \bmod M'.\end{aligned}$$

□

Theorem 4.4.1. *Our encryption scheme is plaintext-secure against a passive generic adversary.*

Proof. A passive adversary can only eavesdrop to get $C = (z_1, z_2)$. For such an adversary, to derive plaintext is equivalent to breaking the ElGamal encryption scheme in $Z_{M'}^*$. Nevertheless, by Lemma 4.4.1, breaking the ElGamal scheme in $Z_{M'}^*$ implies solving the DH problem in the subgroup H of $Z_{M'}^*$ generated by g. For a generic algorithm, solving the DH problem for the subgroup H needs at least $\Omega(q_1^{1/2})$ time. Because $q_1^{1/2}$ is chosen to be large, the theorem is proved. □

4.4.2 Semantic Security

The security of our scheme is based on the ElGamal encryption scheme in $Z^*_{M'}$. To enhance the security, we show how to choose the generator of subgroups and limit the message so that our scheme is semantically secure.

Because g is the common generator of the $Z^*_{p_i}$'s, g^2 has order q_i and generates all the q_i quadratic residues in $Z^*_{p_i}$ for each $i = 1, 2, \ldots, n$. Similarly, g^2 has order $\lambda(M')/2 = q_{i_1} q_{i_2} \cdots q_{i_k}$ and generates all the quadratic residues in $Z^*_{M'}$. So the cyclic subgroup of $Z^*_{M'}$ generated by g^2 has order $q_{i_1} q_{i_2} \cdots q_{i_k}$, each of whose prime factors is at least q_1. The decision Diffie-Hellman problem (DDH) in group G generated by h with a large order is to efficiently distinguish the two distributions: (h^{r_1}, h^{r_2}, h^z) where r_1, r_2, z are random and $(h^{r_1}, h^{r_2}, h^{r_1 r_2})$ where r_1, r_2 are random. Any generic algorithm must perform $\Omega(q_{\min}^{1/2})$ group operations for the DDH problem, where $q_{\min}$ is the smallest prime dividing the order of the group [208]. Modify the parameters in our scheme: Let the common generator be g^2 and assume the plaintext (contents) in Z_{p_1} must be a common quadratic residue of all the p_i's. Because $q_1^{1/2}$ is chosen to be large, the DDH problem for the subgroup generated by g^2 is intractable for a generic adversary. Based on this intractability, our scheme can be shown to be semantically secure by using a similar result in [216] after replacing the modulus p there with our M'.

The parameters were modified so that each plaintext $x \in Z_{p_1}$ is a common quadratic residue of all the p_i's. That is very inconvenient. Here is an alternative. Encrypt x^2 instead of x. After decryption, subscriber i obtains $x' = x^2 \bmod p_i$. As $p_i \equiv 3 \pmod 4$, the solutions to $x^2 \equiv x' \pmod{p_i}$ are $x \equiv \pm x'^{(p_i+1)/4} \pmod{p_i}$. Note that one of the solutions is odd and the other even. If we always pad one bit in the least significant bit of x to make it, say, odd, the plaintext can be uniquely decided. So we have the following theorem.

Theorem 4.4.2. *Our scheme modified as described above is semantically secure against a passive generic adversity.*

4.4.3 Forgery of Decryption Keys

Recall that $\beta, M, \beta_i, p_i, g$, and d_i, for $i = 1, 2, \ldots, n$, are all the keys in the system, among which only d_i are not public. The ciphertext (z_1, z_2) is also public. Because our encryption scheme is plaintext-secure, it is impossible to fake subscribers' keys or create new decryption keys from the public information. This implies that combining d_i's is the only way to create decryption keys. Suppose t of the d_i's are used to create a new decryption key d_H, say $d_1, d_2, \ldots, d_t$. Because $d_1, d_2, \ldots, d_t$ are involved in creating d_H, we solve d_H from

$$g^{d_H} \equiv g^{d_i} \pmod{p_i} \tag{4.1}$$

for $i = 1, \ldots, t$. Let $M'_H = p_1 p_2 \cdots p_t$. The following lemma proves that this d_H works.

Lemma 4.4.2. *Suppose that $d_1, d_2, \ldots, d_t$ are used to create a new decryption key d_H. Then d_H exists and equals $\sum_{i=1}^{t} d_i M_{H_i} y_i \bmod M_H$ if and only if $\gcd(p_i - 1, p_j - 1)$ divides $d_i - d_j$, where $M_H = \operatorname{lcm}(p_1 - 1, p_2 - 1, \ldots, p_t - 1)$, $M_{H_i} = M_H/(p_i - 1)$, and $M_{H_i} y_i \equiv 1 \pmod{p_i - 1}$ for $i = 1, 2, \ldots, t$.*

Proof. By equation (4.1), d_H satisfies $g^{d_H} \equiv \beta \pmod{M'_H}$. Hence d_H and M'_H can be used to decrypt the ciphertext. By Fact 2.2.16, $g^{d_H} \equiv g^{d_i} \pmod{p_i}$ implies $d_H \equiv d_i \pmod{p_i - 1}$. The rest of the lemma follows from Fact 2.2.12. □

The next theorem is immediate.

Theorem 4.4.3. *For a passive generic adversary, the only way to create a new decryption key d_H in our scheme is to combine the d_i's in the way mentioned in Lemma 4.4.2.*

4.5 Traceability, Long-Livedness, Anonymity

Traceability

The tracing algorithm shall utilize the pirate decoder to apprehend the traitors. First, assume that the pirate decryption key d_H and M'_H are revealed by opening the decoder

box to simplify the analysis. The tracing algorithm uses M'_H to detect all traitors as follows: If $M'_H \equiv 0 \pmod{p_i}$, then subscriber i is a traitor; otherwise, he is innocent. Thus all traitors will be captured, and innocent subscribers will not be accused. Because p_i are public, the tracing algorithm does not need the private keys of subscribers to succeed.

Now suppose that the decoder cannot be opened (it is a black box). Then the black-box tracing algorithm performs the following steps for each $i = 1, 2, \ldots, n$ to trace all traitors.

Step 1: Compute $\alpha_i = \beta \bmod M_i$, where $M_i = M/p_i$, for $M = p_1 p_2 \cdots p_n$.

Step 2: Choose a plaintext x. Compute the ciphertext C with the public key (α_i, g, M_i).

Step 3: Feed C to the black-box decoder. If the output is not equal to x, then subscriber i is a traitor.

The tracing algorithm flags the subscribers whose moduli are the prime factors of M'_H. Thus the tracing algorithm does track down all and only traitors. Of course, its performance suffers from not being able to open the box.

Perfectly long-lived keys

When an existing decryption key is discarded or when a new subscriber joins the system, the system shall not require the other subscribers to perform any interaction with the distributor to change their secret keys. The public encryption key will be modified to achieve this goal. Assume that (β, g, M) is the original public encryption key.

Suppose that subscriber i's key is disabled because he is a traitor or he wants to leave the system. Then the new public encryption key is

$$(\hat{\beta}, g, \hat{M}) = (\beta \bmod (M/p_i), g, M/p_i).$$

No rekeying is required for the remaining subscribers. By disabling a traitor's key, the pirate decoder using that key also becomes useless.

Now suppose that a new subscriber $n+1$ joins the system. Then the new public encryption key is

$$\begin{aligned}&(\hat{\beta}, g, \hat{M})\\ =&(\beta p_{n+1} v + \beta_{n+1} M w \bmod M p_{n+1}, g, M p_{n+1}),\end{aligned}$$

where p_{n+1} is a new prime distinct from $p_1, p_2, \ldots, p_n$, $p_{n+1} v \equiv 1 \pmod{M}$, and $Mw \equiv 1 \pmod{p_{n+1}}$. Existing subscribers do not need to update their secret keys, and the new subscriber cannot decrypt the contents received before he joins. After the public encryption key has been changed, the traitor-tracing scheme still satisfies the same properties. The following theorem demonstrates that our method obtains the correct encryption key.

Theorem 4.5.1. *Let $p_1, p_2, \ldots, p_{n+1}$ be distinct primes. Suppose that for any integers $\beta_1, \beta_2, \ldots, \beta_n$, the system of congruences*

$$x \equiv \beta_i \pmod{p_i}, \quad i = 1, 2, \ldots, n,$$

has a unique solution β modulo $M = p_1 p_2 \cdots p_n$. Then $\beta' = \beta \bmod M_n$, where $M_n = M/p_n$, is the unique solution modulo M_n to the system of congruences

$$x \equiv \beta_i \pmod{p_i}, \quad i = 1, 2, \ldots, n-1. \tag{4.2}$$

Furthermore, $\beta'' = \beta p_{n+1} v + \beta_{n+1} M w \bmod M''$ is the unique solution modulo M'' to the system of congruences

$$x \equiv \beta_i \pmod{p_i}, \quad i = 1, 2, \ldots, n+1,$$

where $M'' = M p_{n+1}$, $p_{n+1} v \equiv 1 \pmod{M}$, and $Mw \equiv 1 \pmod{p_{n+1}}$.

Proof. Both β and β' are solutions to congruences (4.2). As the solutions are congruent modulo M_n, we have $\beta' = \beta \bmod M_n$. That β'' is the desired solution follows from Fact 2.2.12. □

Anonymity

In our scheme, a subscriber registers by sending his or her own public information to the distributor. This part is noninteractive. Because all the subscribers receive the

same ciphertext, the broadcast message need no addressing. Even when one obtains the plaintext and the public key of another subscriber, the subscriber's identity remains hidden. As the encryption scheme is probabilistic, encrypting a plaintext with anyone's public keys and then comparing the resulting ciphertext with any historical ciphertext for clues is wasted efforts. Our scheme is hence anonymous.

4.6 Conclusions

In order to prevent others from learning the secret keys, we propose a fully public-key traitor-tracing scheme. Perfect long-livedness and anonymity are achieved. Furthermore, it is a simple task to recompute the encryption key if needed. By the choice of parameters, our scheme can be plaintext-secure or semantically secure against a passive generic adversary. The tracing algorithm is n-traceable and captures all and only traitors. This holds even if the pirate decoder is a black box.

Chapter 5

Group Undeniable Signatures with Convertibility

Group undeniable signatures are like group signatures except that verifying signatures requires the cooperation of the group manager. A convertible group undeniable signature scheme allows the group manager to turn select group undeniable signatures into universally verifiable group signatures. In this chapter we propose such a scheme and discuss its efficiency and security.

5.1 Introduction

Digital signatures are used to verify whether one message really comes from the alleged signer. In general, the signer keeps a secret value to generate his signature and publicizes the corresponding public information for verification purpose. Like handwritten signatures, standard digital signatures are nonrepudiatable and universally verifiable. Nonrepudiation guarantees that a signer cannot deny his signature at a later time. Universal verifiability allows everybody to verify a signature with the signer's public information. However, universal verifiability might not suit the situation, for example, where signatures are generated for sensitive, nonpublic data. Consider two parties striking a confidential deal. Each party wants the other to sign the contract but does not want the contents of the contract released with signatures that can be universally verified. What they need is a signature scheme whose ver-

ification requires interaction with the signers, who then have some control over the sensitive data.

In 1989, Chaum and van Antwerpen [51] introduce undeniable signatures by which anyone must interact with the signer to verify a valid signature through a confirmation protocol and the signer must be able to disavow an invalid signature through a denial protocol. That is, undeniable signatures require that signature verification must be done with the signer's participation. Subsequent works on undeniable signatures include [22, 48, 49, 46, 70, 75, 94, 98, 124, 125, 126, 163, 164, 175, 179]

Convertible undeniable signatures offer an additional flexibility on signature verification. By releasing appropriate verification keys, the signer can convert all or select undeniable signatures into standard digital signatures without compromising the security of the secret key used to generate the signatures. Furthermore, the signer can also give the verification keys to trusted parties so that they can help handle the verification task. As an example that convertible undeniable signatures are preferable to undeniable signatures, consider the problem of keeping digital archives of confidential political or diplomatic documents. Authenticating such records with standard digital signatures is hardly acceptable: If the data are leaked to the press, anyone can verify the signatures and thus the authenticity of the records. Undeniable signatures are clearly more suitable here. However, such records usually become publicly accessible after some years by freedom-of-information laws and should therefore become publicly verifiable. However, at this point the signers who generate the original undeniable signatures may no longer be alive, or physically fit to handle the vast amount of verification requests. This can be solved with convertibility: The signers could make the verification keys public, or give them to trusted parties, who would assume the job of verifying the signatures.

In 1990, Boyar et al. [22] introduce the concept of convertible undeniable signatures. The convertible schemes in [22, 70] consider converting valid undeniable signatures to universally verifiable ones. Michels and Stadler [164] present a convertible undeniable signature scheme in which the signer can convert not only valid undeniable signatures into standard digital signatures but also invalid undeniable signatures into universally verifiable statements about the fact that signatures are invalid.

In contrast to individual signatures, a group signature scheme allows a group

member to sign messages anonymously on behalf of a group. Analogous to standard digital signatures, group signatures are both nonrepudiatable and universally verifiable. In case of later disputes, a designated group manager can use the group signature to trace the actual identity of the signer. But, no one—including the group manager—can attribute a valid signature to a nonsigner. Group signatures have many practical applications such as authenticating price quotes and digital contracts. The concept of group signatures is introduced by Chaum and van Heyst [44]. Camenisch and Stadler [38] present the first scheme in which the sizes of the public key and signatures are independent of the group size. More works on group signatures include [4, 5, 6, 34, 35, 39, 55, 54, 180]

However, as mentioned above, if group signatures are for sensitive and nonpublic data, the group manager may hope that no one can verify the signatures without his participation. For example, when an employee signs a digital contract about a confidential business deal, it is desirable that no one can authenticate the contract without the help of the manager even if the contract is leaked to competitors. The signature is regarded simply as evidence that the contract has been signed by some group member and the signature can not be denied later. We will call these signatures group undeniable signatures. Lyuu and Wu [145, 147] are the first to introduce group undeniable signatures satisfying the following requirements: (1) only group members can anonymously sign on behalf of the group (anonymity); (2) a verifier must interact with the group manager to verify the signature (nontransferability); (3) the group manager can identify the signer of a valid signature (traceability). In addition, we will propose the first convertible group undeniable signature scheme such that the group manager can convert all or select group undeniable signatures into universally verifiable ones without compromising the security of the secret signing key. The proposed scheme also allows the group manager to delegate the ability to confirm and deny signatures to trusted parties without providing them the capability of generating signatures.

The convertibility feature is very useful in practice. Consider the situation of a software company. The software engineer signs off a software product on behalf of the company. When asked, a company manager can prove to the buyer that the product is authentic by verifying the signature's validity. For nonbuyers, in contrast,

the manager may refuse to prove to them the authenticity of the software. But once the company goes bankrupt, authenticity is no longer provable. Now suppose the group undeniable signatures are convertible. Then the manager can convert select signatures into group signatures when the company is still in business; thus buyers can continue to validate the software if necessary. Even after conversion, no one is able to derive the secret key used to generate signatures; thus the original company's signatures cannot be forged. In addition, the manager can delegate the capability of verifying signatures to trusted parties to share the load or to step in if the company fails.

In this chapter, we use signatures of knowledge [38] and undeniable signatures [48] to construct a convertible group undeniable signature scheme. Under reasonable cryptographic assumptions, our signature scheme will be proved to be anonymous, nontransferable, traceable, unforgeable, exculpable, and unlinkable. Moreover, any colluding subset of group members cannot generate valid signatures that cannot be traced. The signature confirmation and denial protocols can be further made zero-knowledge. Finally, the sizes of the public key and signatures are independent of the group size. This desirable feature means the system remains efficient as the group grows.

This chapter is organized as follows. In Section 5.2, useful facts and cryptographic assumptions in number theory are described. In addition, signatures of knowledge that are used as building blocks of our scheme are reviewed. In Section 5.3, the convertible group undeniable signature model and the proposed scheme are presented. Section 5.4 discusses its security. Conclusions are given in Section 5.5.

5.2 Preliminaries

5.2.1 Number-Theoretic Facts and Assumptions

The security of our scheme is based on several number-theoretic results and assumptions. In addition to the results stated in Section 2.2, we establish several new lemmas in this subsection.

As usual, the following notation is used. For positive integer n, $\mathbb{Z}_n$ denotes the ring of integers modulo n, and $\mathbb{Z}_n^*$ denotes the multiplicative group modulo n. Expression

$x \in_R I$ means that x is chosen randomly from set I. Let $\phi(n)$ denote Euler's phi function, which gives the number of positive integers in $\{1, 2, \ldots, n-1\}$ that are relatively prime to n. Let $\langle g \rangle$ denote the cyclic group generated by g. The least positive integer d such that $g^d \equiv 1 \pmod{n}$ is called the order of g modulo n and is denoted by $\mathrm{ord}_n g$ or simply $\mathrm{ord}(g)$ if n is understood.

Let $n = p_1 p_2$, where $p_1 \neq p_2, p_1 = 2q_1 + 1, p_2 = 2q_2 + 1$, and p_1, p_2, q_1, q_2 are all prime numbers. Also let $g \in \mathbb{Z}_n^*$ with $\mathrm{ord}_n g = q_1 q_2$. The following lemmas are useful for the proposed scheme.

Lemma 5.2.1. *Among three consecutive positive integers, there is at least one integer relatively prime to $q_1 q_2$.*

Proof. Note that q_1 and q_2 are odd. Let i be an arbitrary positive integer. We demonstrate that among $i, i+1, i+2$, there is at least one integer relatively prime to $q_1 q_2$. If $\gcd(i, q_1 q_2) = 1$, the lemma is correct. So, without loss of generality, assume $\gcd(i, q_1 q_2) = q_1$. Let $i = a_1 q_1$ for some a_1. Then $i + 1 = a_1 q_1 + 1$ is relatively prime to q_1. If $i+1$ is also relatively prime to q_2, then $\gcd(i+1, q_1 q_2) = 1$ and we are done. So assume $i+1 = a_2 q_2$ for some a_2. Now we have $i+2 = a_1 q_1 + 2$ and $i+2 = a_2 q_2 + 1$. Hence $\gcd(i+2, q_1 q_2) = 1$. □

Lemma 5.2.2. *Let $y_i \in_R \mathbb{Z}_n$ and $z_i = g^{y_i} \bmod n$. Suppose $\gcd(y_i, q_1 q_2) = 1$. An integer $c_i \in \mathbb{Z}_{q_1 q_2}^*$ satisfies both $(z_i g^{c_i})^{q_1} \not\equiv 1 \pmod{n}$ and $(z_i g^{c_i})^{q_2} \not\equiv 1 \pmod{n}$ if and only if $\gcd(y_i + c_i, q_1 q_2) = 1$.*

Proof. Suppose that c_i satisfies $(z_i g^{c_i})^{q_1} \not\equiv 1 \pmod{n}$ and $(z_i g^{c_i})^{q_2} \not\equiv 1 \pmod{n}$. Because $(g^{y_i + c_i})^{q_1} \equiv (z_i g^{c_i})^{q_1} \not\equiv 1 \pmod{n}$ and $\mathrm{ord}_n g = q_1 q_2$, we have $\gcd(y_i + c_i, q_2) = 1$. By the same argument, $\gcd(y_i + c_i, q_1) = 1$. Hence, $\gcd(y_i + c_i, q_1 q_2) = 1$. For the opposite direction, if $\gcd(y_i + c_i, q_1 q_2) = 1$, then $(z_i g^{c_i})^{q_1} \equiv (g^{y_i + c_i})^{q_1} \not\equiv 1 \pmod{n}$ and $(z_i g^{c_i})^{q_2} \equiv (g^{y_i + c_i})^{q_2} \not\equiv 1 \pmod{n}$ because $\mathrm{ord}_n g = q_1 q_2$. □

Lemma 5.2.3. *Let $y_i \in_R \mathbb{Z}_n$ and $z_i = g^{y_i} \bmod n$. Suppose $\gcd(y_i, q_1 q_2) = 1$. Integer $c_i \in \mathbb{Z}_{q_1 q_2}^*$ satisfying $(z_i g^{c_i})^{q_1} \not\equiv 1 \pmod{n}$ and $(z_i g^{c_i})^{q_2} \not\equiv 1 \pmod{n}$ can be obtained by testing at most three consecutive positive integers against the two inequalities.*

Proof. By Lemma 5.2.1, among three consecutive positive integers, there is at least one integer c_i satisfying $\gcd(y_i + c_i, q_1 q_2) = 1$ and thus satisfying $(z_i g^{c_i})^{q_1} \not\equiv 1 \pmod{n}$ and $(z_i g^{c_i})^{q_2} \not\equiv 1 \pmod{n}$ by Lemma 5.2.2. □

Lemma 5.2.4. *Among any nine consecutive positive integers, there is at least three consecutive integers relatively prime to q_1q_2 as long as $q_1, q_2 > 9$.*

Proof. Let $a_1, a_2, \ldots, a_9$ be nine consecutive positive integers. Suppose $a_1 < a_2 < \cdots < a_9$. Because $q_1, q_2 > 9$, among $a_1, a_2, \ldots, a_9$, there are at most two integers a_i and a_j not relatively prime to q_1q_2. Without loss of generality, let $i < j$. Let sets $I_1 = \{a_1, \ldots, a_{i-1}\}$, $I_2 = \{a_{i+1}, \ldots, a_{j-1}\}$, and $I_3 = \{a_{j+1}, \ldots, a_9\}$. There must be at least one set I_i whose cardinality is at least $\lceil 7/3 \rceil = 3$ by the pigeon-hole principle. □

Lemma 5.2.5. *Integer $c_i \in \mathbb{Z}^*_{q_1q_2}$ satisfies* $\gcd((g^{c_i} \bmod n)+1, n) = 1$ *and* $\gcd((g^{c_i} \bmod n) - 1, n) = 1$ *if and only if* $\gcd(c_i, q_1q_2) = 1$.

Proof. By Fact 2.2.24(1), c_i satisfies $\gcd((g^{c_i} \bmod n)+1, n) = 1$ and $\gcd((g^{c_i} \bmod n)-1, n) = 1$ if and only if $\mathrm{ord}_n g^{c_i} = q_1q_2$ or $\mathrm{ord}_n g^{c_i} = 2q_1q_2$. Because $\mathrm{ord}_n g = q_1q_2$, it must hold that $\mathrm{ord}_n g^{c_i} = q_1q_2$. Thus $\gcd(c_i, q_1q_2) = q_1q_2/\mathrm{ord}_n g^{c_i} = 1$ by Fact 2.2.10(1). □

Lemma 5.2.6. *Let $S_b = g^b$ for $b \in \mathbb{Z}^*_{q_1q_2}$. Assume $q_1, q_2 > 9$. Among any nine consecutive positive integers, there must exist three consecutive integers k_1, k_2, k_3 satisfying* $\gcd((g^{k_i}/S_b \bmod n) + 1, n) = 1$ *and* $\gcd((g^{k_i}/S_b \bmod n) - 1, n) = 1$ *for $i = 1, 2, 3$.*

Proof. Among any nine consecutive positive integers, there are at least three consecutive integers k_i satisfying $\gcd(k_i - b, q_1q_2) = 1$ by Lemma 5.2.4. Note that $g^{k_i}/S_b = g^{k_i - b} \pmod n$. Hence $\gcd((g^{k_i}/S_b \bmod n) + 1, n) = 1$ and $\gcd((g^{k_i}/S_b \bmod n) - 1, n) = 1$ hold for $i = 1, 2, 3$ by Lemma 5.2.5. □

Assume G is a cyclic group with order M, where M is the product of two large primes. In the following we simply recall the well-known cryptographic assumptions used in our scheme.

1. The factoring assumption.
   Assume $n = p_1p_2$ where p_1 and p_2 are two large primes. Given n, it is computationally infeasible to find p_1 and p_2.

2. The Diffie-Hellman (DH) problem assumption.
   Assume the order of $g \in G$ is M. Given g^{x_1} and g^{x_2}, it is computationally infeasible to find $g^{x_1x_2}$ even if M is known.

3. The discrete logarithm (DL) assumption.
   Assume the order of $g \in G$ is M. Given $y \in G$ and g, it is computationally infeasible to find a discrete logarithm x satisfying $y = g^x$ even if M or M's factorization is known.

4. The representation problem assumption.
   Assume the orders of $g_1, \ldots, g_k \in G$ are M. Given $y \in G, g_1, \ldots, g_k$, it is computationally infeasible to find a representation $(x_1, x_2, \ldots, x_k)$ of y satisfying $y = g_1^{x_1} g_2^{x_2} \cdots g_k^{x_k}$ even if M is known.

5. The assumption of equality of discrete logarithms (EDL) ([22, 51, 70]).
   Assume $f, g \in G$ have order M. Given $y_1, y_2 \in G$, f, and g, it is computationally infeasible to determine the equality of $\log_f y_1$ and $\log_g y_2$ over $\mathbb{Z}_M$ even if M is known.

The security of our scheme is based on the random oracle model ([9, 90]). An oracle is essentially a black box which answers every query. The random oracle model refers to a setting in which all parties (including the adversary) have oracle access to a truly random function that takes as input a value and outputs a truly random value. In practice, the random oracle is replaced with a good cryptographic hash function such as MD4 [193], MD5 [194], and SHA-1 [91]. To achieve nonrepudiation of signatures, a minimum requirement on the hash function is that it is infeasible for the signer to find two different messages providing the same hash value. This property is called collision resistance. In the random oracle model, the collision-resistant hash function can be seen as an oracle that produces a truly random value for every new query. Of course, if the same input is put to the oracle, identical output is obtained. In this chapter we will regard a coalition-resistant hash function $\mathcal{H}$ as a random oracle.

5.2.2 Building Blocks

Our group undeniable signature scheme employs signatures of knowledge as building blocks. Based on DL and representation problem assumptions, all these signatures of knowledge used can be showed to be simulatable and existentially unforgeable against adaptively chosen message attacks in the random oracle model [38, 184]. Simulatability means that the distribution of the strings that can be efficiently generated without

knowledge of the secret signing key are indistinguishable from the distribution of the actual signatures. Existential unforgeability against adaptively chosen message attacks means that an adversary cannot obtain a new message-signature pair even if he can obtain signatures on chosen messages.

We now introduce some notations used in this chapter. Assume $G = \langle g \rangle$ is a cyclic group with order M, where M is the product of two large primes. The parameters M, G, and g are chosen such that computing discrete logarithms in G to the base g is infeasible. In addition, computing roots in $\mathbb{Z}_M^*$ is also infeasible without knowing the factorization of M. We denote by Greek letters the elements whose knowledge is to be proved and by all other letters the elements that are publicly known. Denote by $\|$ the concatenation of two strings and by $\wedge$ the logical conjunction. Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell (\ell \approx 160)$ be a coalition-resistant hash function throughout the chapter. A signature of knowledge is said to be correct if it passes the associated verification procedure. In the following we describe the signatures of knowledge used in our scheme.

Signatures of Knowledge of Discrete Logarithms and Representations

Note the algebraic setting in the current chapter is slightly different from that in Section 3.4.4. In particular, the order M of G is the product of two large primes and the signer does not know the value M. We present how the two signatures of knowledge SKDL and SKREP described in Section 3.4.4 must be adapted in order to remain secure in the random oracle model, i.e., in order that the corresponding interactive protocols remain honest-verifier zero-knowledge proofs of knowledge.

First consider the case $M = q_1 q_2$ where q_1 and q_2 are two large primes. Because the order M is not prime, the challenge c must be smaller than both prime-factors of M, i.e., an upper bound 2^ℓ on the challenge c is needed. This is to prevent the difference of two random challenges from being congruent to 0 modulo one of the prime factors of M; otherwise, the knowledge extractor would fail to find a witness. An example of an upper bound is $\ell = 0.4 \log_2(M)$ assuming that q_1 and q_2 are $\approx 0.5 \log_2(M)$. On the other hand c must not be too small, i.e., $|c|$ must be polynomial in the input length. Otherwise, the success probability of the knowledge extractor would be too small.

The other case to consider is that the order M is unknown. If the challenge c is chosen from $\{0, \ldots, 2^{\ell}\}$, then the r's must be chosen from $\{0, \ldots, 2^{(|M|+\ell)\Im} - 1\}$, where $\Im$ is a constant > 1. Let $a = 2^{(|M|+\ell)\Im} - 1$ be public. Then,

1. Signature $SKDL[\,\alpha : y = g^{\alpha}\,](m)$ equaling (c, s) is computed as follows. The signer chooses a random integer $r \in_R \{0, \ldots, a\}$ and then computes c and s according to

$$c = \mathcal{H}(m \parallel g \parallel y \parallel g^r),$$
$$s = r - c\alpha.$$

   Anyone can verify (c, s) by checking $c \stackrel{?}{=} \mathcal{H}(m \parallel g \parallel y \parallel g^s y^c)$.

2. Signature

$$SKREP\left[(\alpha_1, \ldots, \alpha_u) : (y_1 = \prod_{j \in \mathcal{J}_1} g_j^{\alpha_{e_{1j}}}) \wedge \cdots \wedge (y_w = \prod_{j \in \mathcal{J}_w} g_j^{\alpha_{e_{wj}}})\right](m),$$

   equaling $(c, s_1, \ldots, s_u)$ is computed as follows. The signer first chooses $r_i \in_R \{0, \ldots, a\}$ for $i = 1, \ldots, u$, computes c as

$$c = \mathcal{H}(m \parallel g_1 \parallel \ldots \parallel g_k \parallel y_1 \parallel \ldots \parallel y_w \parallel \mathcal{J}_1 \parallel \ldots \parallel \mathcal{J}_w \parallel \{e_{ij}\}_{i=1,\ldots,k; j \in \mathcal{J}_i}$$
$$\parallel \prod_{j \in \mathcal{J}_1} g_j^{r_{e_{1j}}} \parallel \ldots \parallel \prod_{j \in \mathcal{J}_w} g_j^{r_{e_{wj}}}),$$

   and calculates

$$s_i = r_i - c\alpha_i.$$

   Anyone can verify the signature by checking

$$c = \mathcal{H}(m \parallel g_1 \parallel \ldots \parallel g_k \parallel y_1 \parallel \ldots \parallel y_w \parallel \mathcal{J}_1 \parallel \ldots \parallel \mathcal{J}_w \parallel \{e_{ij}\}_{i=1,\ldots,k; j \in \mathcal{J}_i}$$
$$\parallel y_1^c \prod_{j \in \mathcal{J}_1} g_j^{s_{e_{1,j}}} \parallel \ldots \parallel y_w^c \prod_{j \in \mathcal{J}_w} g_j^{s_{e_{w,j}}}).$$

Signatures of Knowledge of a Root of a Discrete Logarithm

Such signatures can prove the knowledge of an e-th root of a discrete logarithm of a public key. An e-th root of the discrete logarithm of $y \in G$ to the base g is an integer x satisfying

$$g^{(x)^e} = y,$$

if such an x exists. When the order M of G is known, the signature described in the following definition works for all exponents e but is not very efficient. For small exponents e, even if M is unknown, one can construct more efficient signatures that are presented later. Let $c[i]$ denote the i-th bit of a string counting from the right-hand end.

Definition 5.2.1. *Let $k \leq \ell$ be a security parameter. A $(k+1)$ tuple $(c, s_1, \ldots, s_k) \in \{0,1\}^\ell \times (\mathbb{Z}_M^*)^k$ satisfying*

$$c = \mathcal{H}(m \parallel g \parallel y \parallel e \parallel t_1 \parallel \ldots \parallel t_k)$$

where

$$t_i = \begin{cases} g^{s_i^e} & \text{if } c[i] = 0, \\ y^{s_i^e} & \text{if } c[i] = 1. \end{cases}$$

is a signature of the message $m \in \{0,1\}^*$ based on a proof of an e-th root of the discrete logarithm of y to the base g, and is denoted by

$$SKRDL_M[\alpha : y = g^{\alpha^e}](m).$$

Such a signature can be computed if values M and α satisfying $y = g^{\alpha^e}$ are known. One first chooses $r_i \in_R \mathbb{Z}_M^*$ and then computes

$$c = \mathcal{H}(m \parallel y \parallel g \parallel e \parallel g^{r_1^e} \parallel \cdots g^{r_k^e}),$$
$$s_i = \begin{cases} r_i & \text{if } c[i] = 0, \\ \frac{r_i}{\alpha} \bmod M & \text{otherwise.} \end{cases}$$

One can verify the signature by checking whether

$$c \stackrel{?}{=} \mathcal{H}(m \parallel y \parallel g \parallel e \parallel t_1 \parallel \cdots \parallel t_k)$$
$$\text{with } t_i = \begin{cases} g^{s_i^e} & \text{if } c[i] = 0, \\ y^{s_i^e} & \text{if } c[i] = 1. \end{cases}$$

Lemma 5.2.7. *The identification protocol corresponding to the $SKRDL_M$ is honest-verifier zero-knowledge and a proof of knowledge of an e-th root of the discrete logarithm of y with respect to the base g.*

Proof. (sketch) Proof of knowledge: The proof is analogous to that of Schnorr's identification protocol. We only show how an α with $y = g^{\alpha^e}$ can be computed from two different views having the same commitments. Without loss of generality we assume that the j-th bits of c and $\tilde{c}$ differ and that $c[j] = 0$. Then we have

$$t_j = g^{s_j^e} = y^{\tilde{s}_j^e} = g^{\alpha^e \tilde{s}_j^e}$$

and thus

$$\alpha = \frac{s_j}{\tilde{s}_j} \bmod M$$

because all s_j's and $\tilde{s}_j$'s are relatively prime to M.

Honest-verifier zero-knowledge: The simulator can be constructed clearly. □

For small exponents e, one can construct a more efficient signature of knowledge of an e-th root of a discrete logarithm even though the order M of G is unknown. Assume that $h \in G$ is another generator of G, the discrete logarithm to the base g of which is unknown. We first introduce the following signature.

Definition 5.2.2. *An $(e-1)$-tuple $(v_1, \ldots, v_{e-1}) \in G^{e-1}$ and a signature*

$$SKREP[\,(\gamma_1, \gamma_2 \ldots, \gamma_e, \delta) : v_1 = h^{\gamma_1} g^{\delta} \wedge v_2 = h^{\gamma_2} v_1^{\delta} \wedge \cdots \wedge v_{e-1} = h^{\gamma_{e-1}} v_{e-2}^{\delta} \wedge v = h^{\gamma_e} v_{e-1}^{\delta}\,](m)$$

is a signature of the message $m \in \{0,1\}^*$ based on a proof of knowledge of an e-th root of the g-part of a representation of v to the bases h and g. It is denoted by

$$SKRREP[\,(\alpha, \beta) : v = h^{\alpha} g^{\beta^e}\,](m)$$

Such a signature can be computed efficiently if values α, β satisfying $v = h^{\alpha} g^{\beta^e}$ are known. Let $b = 2^{|M|} - 1$ be public. One first computes the values $v_i = h^{r_i} g^{\beta^i}$ for $i = 1, \ldots, e-1$ with randomly chosen $r_i \in_R \{0, \ldots, b\}$. As $r_i \in_R \{0, \ldots, 2^{|M|} - 1\}$, numbers v_i are truly random element in G. Furthermore, because of equations $v_i = h^{r_i} g^{\beta^i}$ and $v = h^{\alpha} g^{\beta^e}$, we let $\gamma_1 = r_1, \gamma_i = r_i - \beta r_{i-1}$ for $i = 2, \ldots, e-1$, $\gamma_e = \alpha - \beta r_{e-1}$, and $\delta = \beta$. Thus $SKREP$ can be derived.

Using $SKRREP$, we now construct a more efficient signature of knowledge of an e-root of a discrete logarithm for small exponent e.

Definition 5.2.3. *A signature*

$$SKRREP[\,(\alpha,\beta) : y = h^{\alpha} g^{\beta^e}\,](m)$$

and a signature

$$SKDL[\,\gamma : y = g^{\gamma}](m)$$

is a signature of the message $m \in \{0,1\}^*$ *based on a proof of knowledge of the* e*-th root of the discrete logarithm of* y *to the base* g*. It is denoted by*

$$SKRDL[\,\alpha : y = g^{\alpha^e}\,](m).$$

With the secret x, the signer knows a representation $(0, x^e)$ of $y = h^0 g^{x^e}$ to bases h and g. This implies that $\alpha = 0, \beta = x$, and $\gamma = x^e$, and the two underlying signatures can be calculated. To verify $SKRDL$, one checks the correctness of the two components.

5.3 Proposed Scheme

5.3.1 The System Model

The system contains a group manager, a set of group members, nonmember persons, and verifiers. The group manager produces system parameters, provides membership certificates for the joining persons, and manages signature verification. Only group members can sign messages on behalf of the group. A verifier performs signature verification.

A group undeniable signature scheme must satisfy the following requirements.

- Anonymity: No one except the group manager can identify the signer.
- Nontransferability: Only the group manager can prove the validity or invalidity of signatures.
- Traceability: The group manager can identify the signer of a valid signature.

A convertible group undeniable signature scheme allows the group manager to turn all or select group undeniable signatures into group signatures. To present our scheme clearly, we divide the operations of our scheme into the following.

- System setup: The group manager generates the group's secret and public keys.
- Join: To become a group member, a nonmember person first generates his secret key and membership key. He then registers the membership key with the group manager. The group manager finally sends him the membership certificate.
- Sign: A group member signs messages using his secret key, his membership certificate, and the group public key.
- Signature confirmation protocol: To validate a signature requires interacting with the group manager.
- Signature denial protocol: The group manager can prove to anyone that an invalid signature is indeed invalid through a signature denial protocol.
- Open: The group manager can trace the identity of the member who signs a given message.
- Conversion: Conversion involves receipt generations and verification with the help of a receipt.

    1. Individual receipt generation: Given a message, an alleged signature, and the group secret key, the group manager can generate an individual receipt by which anyone can verify whether the alleged signature is valid. A single group undeniable signature can be converted into a group signature by releasing its individual receipt.
    2. Individual verification: Given a message, an alleged signature, an individual receipt, and the group public key, one can check whether the receipt is valid with respect to the alleged signature. If the receipt is valid, the alleged signature can be verified using the receipt.
    3. Universal receipt generation: Given the group secret key, the group manager can generate a universal receipt by which anyone can verify whether a signature from the group is valid. A group undeniable signature scheme can be converted into a group signature scheme by releasing the universal receipt.

4. Universal verification: Given a universal receipt and the group public key, one can check whether the receipt is valid. If the receipt is valid, anyone can verify any signature with the receipt.

In addition to anonymity, nontransferability, and traceability, a group undeniable signature scheme should also satisfy the following security properties.

- Unforgeability: Only group members can sign on behalf of the group.
- Exculpability: Neither the group manager nor a group member can sign on behalf of other group members.
- Unlinkability: No one except the group manager can tell whether two different signatures are generated by the same group member.
- Coalition resistance: Any colluding subset of group members cannot generate valid signatures that can not be traced by the group manager.
- Zero knowledge: The confirmation and denial protocols reveal no extra information beyond the validity or invalidity of signatures.

The efficiency of a group undeniable signature scheme involves the following parameters of interest.

- The size of the group undeniable signature.
- The size of the group public key.
- The complexity of System setup, Join and Open.
- The complexity of Sign and Verify (including the confirmation and deniable protocols).

It is desirable that these parameters are independent of the number of group members.

5.3.2 Realization of the Proposed Scheme

We now describe the details of the proposed scheme in the following.

System Setup

The group manager computes the following values to obtain the group secret key and the group public key.

- $n = p_1 p_2$, where both $p_i = 2q_i + 1$ and q_i are primes for $i = 1, 2$.
- An RSA public key $(q_1 q_2, e_R)$ and secret key d_R.
- Integer $g \in \mathbb{Z}_n^*$ with $\text{ord}_n g = q_1 q_2$.
- $f = g^{r_1}, u = g^{r_2}, t = u^a, S_b = g^b$, where $r_1, r_2, a, b \in_R \mathbb{Z}_{q_1 q_2}^*$, and all arithmetics are modulo n.
- $e, d \in \mathbb{Z}_{q_1 q_2}^*$ such that $ed \equiv 1 \pmod{q_1 q_2}$.
- $S_d = f^d \bmod n$.

It is important that n be chosen such that factoring n and solving the DL problem in $\mathbb{Z}_n^*$ are intractable. By Fact 2.2.24, we can obtain g with order $q_1 q_2$. By Fact 2.2.10(1), the orders of f, u, t, S_b, and S_d are all $q_1 q_2$. The group manager keeps $(a, b, d, e, d_R, p_1, p_2)$ as the group secret key and publishes $(g, f, u, t, e_R, S_d, S_b, n)$ as the group public key.

Join

When user i wants to join the group, he chooses a secret key $y_i \in_R \mathbb{Z}_n$ and computes his membership key $z_i = g^{y_i} \bmod n$. We assume that $\gcd(y_i, q_1 q_2) = 1$. User i sends z_i to the group manager and proves to the group manager that she knows the discrete logarithm of z_i without revealing y_i (see [43, 98] for the protocol). Next the group manager picks $c_i \in \mathbb{Z}_{q_1 q_2}^*$ such that $(z_i g^{c_i})^{q_1} \not\equiv 1 \pmod{n}$ and $(z_i g^{c_i})^{q_2} \not\equiv 1 \pmod{n}$. Lemma 5.2.2 shows that $\gcd(y_i + c_i, q_1 q_2) = 1$, and by Lemma 5.2.3 c_i can be obtained by testing at most three consecutive integers. Then the group manager computes user i's membership certificate as (x_i, v_i, w_i), where

$$\begin{aligned} x_i &= g^{c_i} \bmod n, \\ v_i &= (c_i + b)^{d_R} \bmod q_1 q_2, \\ w_i &= (z_i x_i)^d \bmod n, \end{aligned}$$

and sends (x_i, v_i, w_i) to user i. Note that

$$w_i = (z_i x_i)^d \bmod n = (g^{y_i+c_i})^d \bmod n. \tag{5.1}$$

The 4-tuple (y_i, x_i, v_i, w_i) is called a valid signing key. It is important to note that the group manager must choose distinct c_i's for different joining users and must not reveal c_i to anybody. Fact 2.2.10(1) implies $\mathrm{ord}(z_i) = \mathrm{ord}(x_i) = \mathrm{ord}(w_i) = q_1 q_2$.

Sign

Given a message m, user i can generate signature S by computing the following values:

- $\hat{g} = g^r \bmod n$ for $r \in_R \mathbb{Z}_n$ (assume $\gcd(r, q_1 q_2) = 1$).
- $Z_0 = S_b^r \bmod n$.
- $Z_1 = \hat{g}^{y_i} \bmod n$.
- $Z_2 = x_i^r \bmod n$.
- $A_1 = g^{y_i} u^r \bmod n$.
- $A_2 = t^r \bmod n$.
- $S_0 = SKREP[\,(\alpha, \beta) : \hat{g} = g^\beta \bmod n \wedge Z_0 = S_b^\beta \bmod n \wedge Z_1 = \hat{g}^\alpha \bmod n \wedge A_1 = g^\alpha u^\beta \bmod n \wedge A_2 = t^\beta \bmod n\,](m)$.
- $S_1 = SKRDL[\,\gamma : Z_2 Z_0 \equiv \hat{g}^{\gamma^{e_R}} \pmod n\,](m)$.
- $S_2 = w_i^r \bmod n$.

User i's group undeniable signature for m is

$$S = (\hat{g}, Z_0, Z_1, Z_2, A_1, A_2, S_0, S_1, S_2).$$

Note that $Z_2 Z_0 \equiv \hat{g}^{c_i} \hat{g}^b \equiv \hat{g}^{c_i+b} \pmod n$, $Z_1 Z_2 \equiv \hat{g}^{y_i+c_i} \pmod n$ and

$$S_2 \equiv w_i^r \equiv ((g^{y_i+c_i})^d)^r \equiv ((\hat{g})^{y_i+c_i})^d \equiv (Z_1 Z_2)^d \pmod n. \tag{5.2}$$

Because $\gcd(y_i + c_i, q_1 q_2) = 1$ and $\gcd(d, q_1 q_2) = 1$, the orders of $Z_1 Z_2$ and S_2 are $q_1 q_2$ by Fact 2.2.10(1). In addition, the orders of $\hat{g}, Z_0, Z_1, Z_2, A_2$ are also $q_1 q_2$. We call

S a valid group undeniable signature if S_0 and S_1 are correct and $S_2 = (Z_1 Z_2)^d \bmod n$. Obviously, if S is generated using a valid signing key, then S is a valid group undeniable signature.

We briefly explain what roles some of the elements in S play. First, S_0 proves that the same random number r is used in the computation of $\hat{g}, Z_0, A_1$, and A_2, and that the same exponent $y \in Z$ is used in $Z_1 = \hat{g}^y \bmod n$ and $A_1 = g^y u^r \bmod n$. S_1 proves that user i knows the knowledge of an e_R-th root of the discrete logarithm of $Z_2 Z_0$ to base $\hat{g}$. Finally, the verifier must interact with the group manager to verify whether $S_2 = (Z_1 Z_2)^d \bmod n$ via the signature confirmation and denial protocols.

Signature Confirmation Protocol

A signature confirmation protocol is an interactive protocol between the group manager and the verifier whereby the group manager can convince the verifier that the signature is valid. However, the group manager cannot deceive the verifier into accepting an invalid signature as valid except with a very small probability. In the following, we denote by $\mathcal{P}$ the group manager and by $\mathcal{V}$ the verifier. In the signature confirmation protocol, common inputs to $\mathcal{P}$ and $\mathcal{V}$ include the message m, the group public key, and the alleged signature S. The secret input to $\mathcal{P}$ is the group secret key.

To be convinced that S is valid, $\mathcal{V}$ first verifies S_0 and S_1. If either is incorrect, then $\mathcal{V}$ recognizes that S is invalid. Otherwise, $\mathcal{P}$ and $\mathcal{V}$ perform the following steps:

Step 1: $\mathcal{V}$ chooses $e_1, e_2 \in_R \mathbb{Z}_n$ and computes $A = S_2^{e_1} S_d^{e_2} \bmod n$. Then $\mathcal{V}$ sends A to $\mathcal{P}$.

Step 2: $\mathcal{P}$ computes $B = A^e \bmod n$ and sends B to $\mathcal{V}$.

Step 3: $\mathcal{V}$ verifies whether $B = (Z_1 Z_2)^{e_1} f^{e_2} \bmod n$. If the equality holds, then $\mathcal{V}$ accepts S as a valid signature for m.

In the following, we first show that $\mathcal{V}$ accepts valid signatures. We then show that $\mathcal{P}$ cannot make $\mathcal{V}$ accept invalid signatures as valid except with a very small probability.

Theorem 5.3.1. *If S is a valid group undeniable signature, then the verifier will accept S as a valid signature.*

Proof. S_0 and S_1 must be correct. Because $S_2 = (Z_1 Z_2)^d \bmod n$, we have

$$B \equiv A^e \equiv ((S_2)^{e_1}(S_d)^{e_2})^e \equiv (Z_1 Z_2)^{e_1} f^{e_2} \pmod{n}.$$

□

Theorem 5.3.2. *If S is not a valid group undeniable signature, then the verifier will accept S as a valid signature with probability $\leq 1/(q_1 q_2)$.*

Proof. If either S_0 or S_1 is incorrect, the verifier recognizes S as invalid. Now suppose S_0 and S_1 are correct. Because S is invalid, $S_2 \neq (Z_1 Z_2)^d \bmod n$. $\mathcal{P}$ can make $\mathcal{V}$ accept the signature S only if $\mathcal{P}$ can find a B such that

$$B = (Z_1 Z_2)^{e_1} f^{e_2} \bmod n \tag{5.3}$$

$$A = S_2^{e_1} S_d^{e_2} \bmod n. \tag{5.4}$$

As the order of f is $q_1 q_2$, we let $A = f^i$, $B = f^j$, $S_2 = f^k$, and $Z_1 Z_2 = f^\ell$, where $0 \leq i, j, k, \ell < q_1 q_2$ and all arithmetics are modulo n. Recall $S_d = f^d \bmod n$. From Eqs. (5.3) and (5.4), we have

$$j \equiv \ell e_1 + e_2 \pmod{q_1 q_2},$$
$$i \equiv k e_1 + d e_2 \pmod{q_1 q_2}.$$

As $f^k \not\equiv f^{\ell d} \pmod{n}$, we have $k \not\equiv \ell d \pmod{q_1 q_2}$ and the linear equations have a unique solution for (e_1, e_2) for $0 \leq e_1, e_2 < q_1 q_2$.

Because the orders of S_2 and S_d are $q_1 q_2$, there are $q_1 q_2$ pairs (e_1, e_2) for $0 \leq e_, e_2 < q_1 q_2$ satisfying Eq. (5.4). $\mathcal{P}$ cannot identify which among them was used to compute A by $\mathcal{V}$. In addition, because the orders of $Z_1 Z_2$ and f are $q_1 q_2$, each of these possible $q_1 q_2$ pairs (e_1, e_2) corresponds to a different j and hence B. Consequently, the probability that $\mathcal{P}$ will give $\mathcal{V}$ the correct B is $1/(q_1 q_2)$. □

In order to state the protocol clearly, the above steps omit the zero-knowledge part. However, there are well-known techniques [27, 106, 108] to add the zero-knowledge property to the above protocol as follows. Instead of $\mathcal{P}$ sending B in Step 2, he sends

a commitment of B to $\mathcal{V}$ using a commitment scheme [27, 168], after which $\mathcal{V}$ reveals to $\mathcal{P}$ the values of e_1 and e_2. After checking that $B \equiv (Z_1 Z_2)^{e_1} f^{e_2} \pmod{n}$, $\mathcal{P}$ sends B to $\mathcal{V}$. $\mathcal{V}$ checks that B corresponds to the value committed by $\mathcal{P}$ and then performs the test of Step 3. In this way, if $\mathcal{V}$ knows e_1 and e_2, he can compute B. Hence, the zero-knowledge property is achieved through the following two characteristics of the commitment scheme: (1) It is infeasible for $\mathcal{V}$ to derive B from the commitment of B, and (2) $\mathcal{P}$ cannot find B' such that B' and B have the same commitment.

Signature Denial Protocol

A signature denial protocol allows $\mathcal{P}$ to convince $\mathcal{V}$ of the fact that an invalid signature is indeed invalid. However, $\mathcal{P}$ cannot make $\mathcal{V}$ believe that a valid signature is invalid except with a very small probability. In the denial protocol, the common inputs to $\mathcal{P}$ and $\mathcal{V}$ include two system global constants C_1 and C_2, the message m, the group public key, and the alleged signature S. The secret input to $\mathcal{P}$ is the group secret key.

We now present how $\mathcal{P}$ can make $\mathcal{V}$ accept an invalid signature S as invalid. $\mathcal{V}$ starts by checking S_0 and S_1. If either is incorrect, then $\mathcal{V}$ recognizes that S is invalid. Otherwise, $\mathcal{P}$ and $\mathcal{V}$ repeat the following steps C_2 times.

Step 1: $\mathcal{V}$ chooses $e_1 \in_R \mathbb{Z}_{C_1}, e_2 \in_R \mathbb{Z}_n$ and computes $D_1 = (Z_1 Z_2)^{e_1} f^{e_2} \bmod n$ and $D_2 = S_2^{e_1} S_d^{e_2} \bmod n$. Then $\mathcal{V}$ sends D_1 and D_2 to $\mathcal{P}$.

Step 2: $\mathcal{P}$ finds B such that $D_1/D_2^e \equiv (Z_1 Z_2/S_2^e)^B \pmod{n}$ by trying $B = 0, 1, \ldots, C_1 - 1$ and sends B to $\mathcal{V}$.

Step 3: $\mathcal{V}$ checks whether $B = e_1$. If the equality holds, then $\mathcal{V}$ is convinced that S is invalid.

If $\mathcal{V}$ is convinced of S's invalidity C_2 times, $\mathcal{V}$ will accept S as invalid. It is noteworthy that $\mathcal{P}$ performs $O(C_1 C_2)$ operations.

The protocol satisfies the following two properties. First, $\mathcal{P}$ can convince $\mathcal{V}$ of the fact that an invalid signature is indeed invalid. Second, $\mathcal{P}$ cannot fool $\mathcal{V}$ into accepting a valid signature as invalid except with a small probability.

Theorem 5.3.3. *If S is not a valid group undeniable signature, then the verifier will accept S as an invalid signature.*

Proof. If S_0 or S_1 is incorrect, the verifier will recognize S as an invalid signature. Suppose S_0 and S_1 are both correct. Because S is invalid, $S_2 \neq (Z_1 Z_2)^d \bmod n$ and therefore $S_2^e \not\equiv Z_1 Z_2 \pmod n$. As $D_1/D_2^e \equiv ((Z_1 Z_2)^{e_1} f^{e_2})/(S_2^{e_1} S_d^{e_2})^e \equiv (Z_1 Z_2/S_2^e)^{e_1} \pmod n$, $\mathcal{P}$ can always find the required e_1 as B. So $\mathcal{V}$ will accept S as an invalid signature. □

Theorem 5.3.4. *If S is a valid group undeniable signature, then the verifier will accept S as an invalid signature with probability $1/C_1^{C_2}$.*

Proof. Because S is valid, S_0 and S_1 are correct and $S_2 = (Z_1 Z_2)^d \bmod n$. Therefore $S_2^e \equiv Z_1 Z_2 \pmod n$. As $D_1/D_2^e \equiv ((Z_1 Z_2)^{e_1} f^{e_2})/(S_2^{e_1} S_d^{e_2})^e \equiv (Z_1 Z_2/S_2^e)^{e_1} \equiv (Z_1 Z_2/(Z_1 Z_2)^{de})^{e_1} \equiv 1 \pmod n$, $\mathcal{P}$ will guess e_1 correctly with probability $1/C_1$ in each round. So $\mathcal{V}$ will accept S as an invalid signature with probability $1/C_1^{C_2}$. □

For simplicity the above protocol omits the zero-knowledge part. We can add the zero-knowledge property to the above protocol as follows: Instead of $\mathcal{P}$ sending B in Step 2, he sends a commitment of B to $\mathcal{V}$ using a commitment scheme [27, 168], after which $\mathcal{V}$ reveals to $\mathcal{P}$ the value of e_1. After checking that $B = e_1$, $\mathcal{P}$ sends B to $\mathcal{V}$. $\mathcal{V}$ checks that B corresponds to the value committed by $\mathcal{P}$ and then performs the test of Step 3. Consequently, the zero-knowledge property is achieved as explained in the signature confirmation protocol.

Open

Given a valid signature S, the group manager can compute

$$A_1 A_2^{-(a^{-1} \bmod q_1 q_2)} \bmod n,$$

which equals

$$(g^{y_i} u^r)((u^a)^r)^{-(a^{-1} \bmod q_1 q_2)} \bmod n = g^{y_i} \bmod n.$$

The signer with the membership key $z_i = g^{y_i} \bmod n$ can be traced.

Conversion

In the phase, the group manager converts all or select group undeniable signatures into group signatures. Details of operations are described below.

1. Individual receipt generation.
   Let S be a signature for message m. The group manager chooses $r \in_R \mathbb{Z}^*_{q_1q_2}$ and computes S's individual receipt as $R = (\tilde{f}, R_1, R_2, R_3)$, where

$$\begin{aligned}
\hat{f} &= f^r \bmod n, \\
R_1 &= (Z_1Z_2)^r \bmod n, \\
H &= \mathcal{H}(m \parallel \hat{f} \parallel R_1) \text{ (assume } \gcd(H, q_1q_2) = 1), \\
R_2 &= SKREP[\,\alpha : R_1 = (Z_1Z_2)^\alpha \bmod n \wedge \hat{f} = f^\alpha \bmod n,\,](m), \\
R_3 &= (r - Hd) \bmod q_1q_2.
\end{aligned}$$

   Obviously it is infeasible to derive the secret key d from the individual receipt.

2. Individual verification.
   Note that $f^{R_3}S_d^H \equiv f^{R_3}f^{Hd} \pmod n$. Hence, $f^{R_3}S_d^H \equiv f^r \pmod n$ if and only if $R_3 \equiv (r - Hd) \pmod{q_1q_2}$. To validate R, the verifier checks the correctness of R_2 and tests whether $f^{R_3}S_d^H \equiv \hat{f} \pmod n$. If both succeed, then the receipt R is valid; otherwise, the receipt is invalid. If R is valid, then the alleged signature S can be verified by checking the correctness of S_0 and S_1 and testing whether $(Z_1Z_2)^{R_3}S_2^H \equiv R_1 \pmod n$ (see Lemma 5.3.1 below). Hence, with a valid individual receipt, the alleged signature can be verified.

   Lemma 5.3.1. *Assume R is valid. Then, $(Z_1Z_2)^{R_3}S_2^H \equiv R_1 \pmod n$ if and only if $S_2 = (Z_1Z_2)^d \bmod n$.*

   Proof. Because R is valid, we have $R_3 \equiv (r - Hd) \pmod{q_1q_2}$ and thus $(Z_1Z_2)^{R_3}S_2^H \equiv (Z_1Z_2)^{(r-Hd)}S_2^H \pmod n$. Suppose $(Z_1Z_2)^{R_3}S_2^H \equiv R_1 \pmod n$. Then $(Z_1Z_2)^{(r-Hd)}S_2^H \equiv (Z_1Z_2)^r \pmod n$. So $S_2^H \equiv (Z_1Z_2)^{Hd} \pmod n$. Thus $S_2 = (Z_1Z_2)^d \bmod n$. For the opposite direction, if $S_2 = (Z_1Z_2)^d \bmod n$, then $(Z_1Z_2)^{R_3}S_2^H \equiv (Z_1Z_2)^{(r-Hd)}((Z_1Z_2)^d)^H \equiv (Z_1Z_2)^r \pmod n$. □

3. Universal receipt generation.
   To make all signatures universally verifiable, the group manager releases secret e as the universal receipt. According to the basic assumption behind RSA, this does not compromise the security of the secret key d.

4. Universal verification.
   To validate e, the verifier tests whether $f = S_d^e \bmod n$. If the equality holds, then e is valid; otherwise, e is invalid. If e is valid, then all alleged signatures can be verified by checking the correctness of S_0 and S_1 and testing whether $S_2^e \equiv Z_1 Z_2 \pmod{n}$. This works because $S_2 = (Z_1 Z_2)^d \bmod n$ if and only if $S_2^e \equiv Z_1 Z_2 \pmod{n}$ because $ed \equiv 1 \pmod{q_1 q_2}$. Consequently, the group undeniable signature scheme can be converted into a group signature scheme by releasing the universal receipt e. In addition, our scheme allows the group manager to delegate the ability to confirm and deny signatures to trusted parties by issuing e to them only.

5.4 Security Analysis

Under the random oracle model, the security of our scheme is based on the standard cryptographic assumptions described in Section 5.2. In the following we show that the proposed scheme satisfies the security properties of group undeniable signatures.

5.4.1 Exculpability

Because the DL problem is intractable, neither the group manager nor a group member can derive the secret key of another group member. Thus it is infeasible to frame another member.

5.4.2 Unforgeability

A valid signature $(\hat{g}, Z_0, Z_1, Z_2, A_1, A_2, S_0, S_1, S_2)$ must contain correct S_0, S_1, and S_2. To be correct, it must hold that $S_2 = (Z_1 Z_2)^d \bmod n$ by Eq. (5.2). However, using adaptive chosen message attacks, the attacker can only obtain $Z^d \bmod n$ with random Z. Let $S_2 = Z^d \bmod n$. Then the attacker must obtain Z_1 and Z_2 such that $Z_1 Z_2 \equiv Z$

(mod n) and both S_0 and S_1 are correct. Note that $S_0 = SKREP[(\alpha, \beta) : \hat{g} = g^\beta \bmod n \wedge Z_0 = S_b^\beta \bmod n \wedge Z_1 = \hat{g}^\alpha \bmod n \wedge A_1 = g^\alpha u^\beta \bmod n \wedge A_2 = t^\beta \bmod n](m)$ and $S_1 = SKRDL[\gamma : Z_2 Z_0 \equiv \hat{g}^{\gamma^{e_R}} \pmod n](m)$. To obtain S_0, the attacker must choose α and β. Thus $Z_0 = S_b^\beta \bmod n$ and $Z_1 = \hat{g}^\alpha \bmod n$. Let c be the discrete logarithm of $Z_2 = Z Z_1^{-1} \bmod n$ to base $\hat{g}$. Because c, b, and d_R are unknown and the DL problem is intractable, it is infeasible to derive $\gamma \equiv (c+b)^{d_R} \pmod{q_1 q_2}$ so that $\hat{g}^{\gamma^{e_R}} \equiv Z_2 Z_0 \pmod n$. Therefore, it is infeasible to generate S_1. Simultaneously generating S_0, S_1, and S_2 is thus infeasible.

It is known that S_0 and S_1 are existentially unforgeable against adaptive chosen message attacks [38, 184]. Consequently, the proposed scheme is existentially unforgeable against adaptive chosen message attacks.

5.4.3 Anonymity, Nontransferability, and Unlinkability

If the signature is simulatable, then it itself reveals no information, and thus the three properties will hold. So it suffices to show that the signature can be simulated. Let S be a valid signature. Because the order of u is $q_1 q_2$, we can assume the signer's membership key z_i equals $u^{r_z} \bmod n$ for some r_z. So $A_1 \equiv g^{y_i} u^r \equiv z_i u^r \equiv u^{r_z + r} \pmod n$. To generate an $\tilde{S}$ indistinguishable from actual signatures, the simulator randomly chooses $\bar{r}$, $\tilde{r}$, $\tilde{y}$, $\tilde{c}$, $\tilde{d} \in \mathbb{Z}$ relatively prime to $q_1 q_2$ (also $\gcd(\tilde{y} + \tilde{c}, q_1 q_2) = 1$) and then computes $\tilde{g} = g^{\tilde{r}}, \tilde{Z}_0 = S_b^{\tilde{r}}, \tilde{Z}_1 = \tilde{g}^{\tilde{y}}, \tilde{Z}_2 = \tilde{g}^{\tilde{c}}, \tilde{A}_1 = u^{\bar{r}}, \tilde{A}_2 = t^{\tilde{r}}$, and $\tilde{S}_2 = (\tilde{Z}_1 \tilde{Z}_2)^{\tilde{d}}$, where all arithmetics are modulo n. Obviously, $\tilde{g}, \tilde{Z}_0, \tilde{A}_1$, and $\tilde{A}_2$ are indistinguishable from $\hat{g}, Z_0, A_1$, and A_2, respectively. Because the EDL problem is intractable, $\tilde{Z}_1, \tilde{Z}_2$ and $\tilde{S}_2$ are indistinguishable from Z_1, Z_2 and S_2, respectively [98]. In addition, S_0, and S_1 are simulatable in the random oracle model [184]. Furthermore, note that the simulator generates $\tilde{S}$ following the specification of the system. So $\tilde{S}$ is indistinguishable from the actual signature S. Hence, the signature is simulatable.

5.4.4 Coalition Resistance

First we show that three colluding members may together compute $g^d \bmod n$ efficiently. Any two colluding members i and j can compute

$$T_1 = y_i - y_j,$$
$$T_2 = v_i^{e_R} - v_j^{e_R}.$$

Note that $T_2 \equiv v_i^{e_R} - v_j^{e_R} \equiv (c_i + b) - (c_j + b) \equiv c_i - c_j \pmod{q_1 q_2}$. By Eq. (5.1), the colluding members i and j can compute

$$\begin{aligned} T_3 &= w_i / w_j \bmod n \\ &= g^{(y_i + c_i)d} / g^{(y_j + c_j)d} \bmod n \\ &= g^{[(y_i - y_j) + (c_i - c_j)]d} \bmod n \\ &= g^{(T_1 + T_2)d} \bmod n. \end{aligned}$$

Assume the number of the colluding members exceeds two. Two of the colluding members can obtain (T_1', T_2', T_3') and another two of the colluding members can obtain (T_1'', T_2'', T_3''). Suppose that $\gcd(T_1' + T_2', T_1'' + T_2'') = 1$, then by the extended Euclidean algorithm they can find E_1 and E_2 such that $E_1(T_1' + T_2') + E_2(T_1'' + T_2'') = 1$. Finally, they calculate $g^d \bmod n$ as $(T_3')^{E_1}(T_3'')^{E_2} \bmod n$ because $(T_3')^{E_1}(T_3'')^{E_2} \equiv (g^{(T_1' + T_2')d})^{E_1}(g^{(T_1'' + T_2'')d})^{E_2} \equiv g^d \pmod n$.

We now show that it is infeasible for colluding members to generate an untraceable signature $(\hat{g}, Z_0, Z_1, Z_2, A_1, A_2, S_0, S_1, S_2)$ such that S_0, S_1, and S_2 are correct even if $g^d \bmod n$ is available. To be correct, it must hold that $S_2 = (Z_1 Z_2)^d \bmod n$. In addition, the two values Z_1 and Z_2 need to make S_0 and S_1 correct. Here $S_0 = SKREP[\,(\alpha, \beta) : \hat{g} = g^\beta \bmod n \wedge Z_0 = S_b^\beta \bmod n \wedge Z_1 = \hat{g}^\alpha \bmod n \wedge A_1 = g^\alpha u^\beta \bmod n \wedge A_2 = t^\beta \bmod n\,](m)$ and $S_1 = SKRDL[\,\gamma : Z_2 Z_0 \equiv \hat{g}^{\gamma^{e_R}} \pmod n\,](m)$. To obtain an untraceable S_0, the colluding members choose α and β, where α differs from any of the colluding members' secrets y_i. To obtain S_1, the colluding members choose γ. Let c satisfy $\gamma^{e_R} \equiv (c + b) \pmod{q_1 q_2}$. Because solving b is infeasible (see Theorem 5.4.1 below), deriving such a c from γ^{e_R} is infeasible. However, such a c must be used for $S_2 = (g^\beta)^{d(\alpha + c)} \bmod n$ by Eq. (5.2). But the colluding members do not know which c to use. Furthermore, by the DH problem assumption, computing $(g^\beta)^{d(\alpha+c)} \bmod n$ is

infeasible if $(g^\beta)^d \bmod n$ and $(g^\beta)^{(\alpha+c)} \bmod n$ are known but d and $\alpha + c$ are unknown (here, we need $g^d \bmod n$). Accordingly, it is infeasible for the colluding members to obtain α, β, γ, and $(g^\beta)^{d(\alpha+c)} \bmod n$ simultaneously. Hence, the proposed signature scheme is coalition-resistant.

Theorem 5.4.1. *Under the DL assumption, solving for b is infeasible for a group member even if he has access to other members' secret keys and membership certificates.*

Proof. Suppose for contradiction that b can be solved. We next show that the discrete logarithm of $S_b = g^b \bmod n$ with known q_1 and q_2 can be solved, contradicting the DL assumption. We simulate members' secret keys and membership certificates as follows.

1. Choose $y_i \in_R \mathbb{Z}_n$ such that $\gcd(y_i, q_1 q_2) = 1$.
2. Choose c_i' satisfying
   (a) $\gcd((g^{c_i'}/S_b \bmod n) + 1, n) = 1$,
   (b) $\gcd((g^{c_i'}/S_b \bmod n) - 1, n) = 1$,
   (c) $(g^{y_i}(g^{c_i'}/S_b))^{q_1} \not\equiv 1 \pmod{n}$,
   (d) $(g^{y_i}(g^{c_i'}/S_b))^{q_2} \not\equiv 1 \pmod{n}$.

   Note that $g^{c_i'}/S_b = g^{c_i'-b} \pmod{n}$ and c_i' plays the role of $(c_i + b) \bmod q_1 q_2$ in the Join phase. Conditions (a) and (b) ensure that $\gcd(c_i, q_1 q_2) = 1$ by Lemma 5.2.5, and conditions (c) and (d) ensure that $\gcd(y_i + c_i, q_1 q_2) = 1$ by Lemma 5.2.2. We now show that c_i' can be obtained efficiently. By Lemma 5.2.6 three consecutive integers satisfying conditions (a) and (b) can be obtained by testing at most nine consecutive integers. Then by Lemma 5.2.3, c_i' can be obtained by testing the three consecutive integers for conditions (c) and (d). With c_i', members' membership certificates are simulated as (x_i, v_i, w_i), where

$$\begin{aligned} x_i &= g^{c_i'}/S_b \bmod n, \\ v_i &= (c_i')^{d_R} \bmod q_1 q_2,\ d_R \in_R \mathbb{Z}^*_{\phi(q_1 q_2)}, \\ w_i &= (g^{y_i} x_i)^d \bmod n,\ d \in_R \mathbb{Z}^*_{q_1 q_2}. \end{aligned}$$

Thus we can solve for b using the secret keys and membership certificates above. □

5.5 Conclusions

Group undeniable signatures are like group signatures except that verifying signature needs the participation of the group manager. In this chapter, we employ signatures of knowledge and undeniable signature concepts to construct the first convertible group undeniable signature scheme in which the group manager can turn all or select group undeniable signatures into group signatures without compromising the security of the secret key used to generate signatures. The proposed scheme also allows the group manager to delegate the ability to confirm and deny signatures to trusted parties without providing them the capability of generating signatures. Moreover, the sizes of the group public key and signatures are independent of the group size. This makes the system scalable. Under standard cryptographic assumptions and the random oracle model, the present scheme is proved to be anonymous, nontransferable, traceable, unforgeable, exculpable, and unlinkable. Furthermore, any colluding subset of group members cannot generate valid signatures that cannot be traced. Finally, the signature confirmation and denial protocols could be made zero-knowledge using the commitment techniques.

Chapter 6

Concluding Remarks

Cryptology is a very important technology in electronic security systems. At the earliest stage of computer system development, protecting individual privacy and authenticity may be sufficient for ensuring information security. However, this becomes insufficient after the advent of computer networks. Networks bring many new types of relationships to computers and to the society, as well as many new sources and types of risks and threats. To cope with these new risks and threats, new methods for information protection are developed. Thus many forms of confidential communication between two or more parties may be performed over an insecure computer network. In this thesis we present schemes for two new group-oriented applications: a fully public-key traitor-tracing scheme and a convertible group undeniable signature scheme. In addition, we study many basic cryptographic techniques that are essential when one constructs complex security systems.

For traitor-tracing applications, we propose a fully public-key traitor-tracing scheme in which every subscriber can prevent others, including the distributor, from learning his secret key. By the choice of parameters, our scheme can be plaintext-secure or semantically secure against a passive generic adversary. There are several desirable properties in our scheme.

1. Key longevity and subscribers' anonymity are achieved.

2. It is a simple task for the distributor to recompute the encryption key if needed.

3. The tracing algorithm can capture all and only traitors even if the pirate decoder

is a black box.

For group undeniable signatures, we are the first to introduce the concept of such signatures. They are more suitable than group signatures in applications where signatures are generated for sensitive, nonpublic data. The first convertible group undeniable signature is proposed in which the group manager can turn select group undeniable signatures into group signatures without compromising the security of the secret key used to generate signatures. There are several desirable properties in the proposed scheme.

1. The group manager can delegate the ability to confirm and deny signatures to trusted parties without providing them the capability of generating signatures.

2. The sizes of the group public key and signatures are independent of the group size.

3. Under standard cryptographic assumptions and the random oracle model, our scheme is proved to be anonymous, nontransferable, traceable, unforgeable, exculpable, and unlinkable.

4. Any colluding subset of group members cannot generate valid signatures that cannot be traced.

5. The signature confirmation and denial protocols could be made zero-knowledge using the commitment techniques.

Further we list several relative problems that deserve to be studied in the future.

1. Key revocation for group (undeniable) signatures is important. When misusing anonymity, a cheating member must be revoked by the group manager, making him unable to sign in the future but without sacrificing the security of past group (undeniable) signatures. It is desirable to design an efficient group (undeniable) signature scheme such that after key revocation, the private keys of the remaining members need not be changed.

2. In a group, a smart security policy is that a set of parties must cooperate in order to carry out some specific task. A typical example is the Shamir threshold

scheme [204]. Because different group may have different access strategies, it seems interesting to explore group-oriented encryption or signature schemes considered under various threshold situations.

3. In our fully public-key traitor-tracing scheme, the size of message transmission is dependent on the number of total subscribers. It is desirable to construct a scheme such that the size of message transmission is independent of the number of total subscribers.

Bibliography

[1] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES in P," 2002. http://www.cse.iitk.ac.in/news/primality.html.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithm*. Addison-Wesley, 1974.

[3] R. Anderson and S. Vaudenay, "Minding your p's and q's," in *Advances in Cryptology—ASIACRYPT '96*, vol. 1163 of *LNCS*, pp. 26–35, Springer-Verlag, 1996.

[4] G. Ateniese, M. Joye, and G. Tsudik, "On the difficulty of coalition-resistant in group signature schemes," in *SCN'99, Second Workshop on Security in Communication Networks*, 1999.

[5] G. Ateniese and G. Tsudik, "Some open issues and new directions in group signature schemes," in *Financial Cryptography, FC'99*, vol. 1648 of *LNCS*, pp. 196–211, Springer-Verlag, 1999.

[6] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Advances in Cryptology—CRYPTO 2000*, vol. 1880 of *LNCS*, pp. 255–270, Springer-Verlag, 2000.

[7] B. Barak, "How to go beyond the black-box simulation barrier," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, FOCS '01*, pp. 106–115, IEEE Computer Society, 2001.

[8] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Advances in Cryptology—CRYPTO '92*, vol. 740 of *LNCS*, pp. 390–420, Springer-Verlag, 1992.

[9] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 62–73, ACM, 1993.

[10] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway, "Everything provable, is provable in zero-knowledge," in *Advances in Cryptology—CRYPTO '88*, vol. 403 of *LNCS*, pp. 37–56, Springer-Verlag, 1990.

[11] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the 20th Annual Symposium on Theory of Computing (STOC '88)*, pp. 1–10, ACM Press, 1988.

[12] S. Berkovits, "How to broadcast A secret," in *Advances in Cryptology—EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 535–541, Springer-Verlag, 1991.

[13] D. Bleichenbacher, "Generating ElGamal signatures without knowing the secret key," in *Advances in Cryptology—EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 10–18, Springer-Verlag, 1996.

[14] M. Blum, "Coin flipping by telephone: A protocol for solving impossible problems," in *Proceedings of the 24th IEEE Computer Conference, IEEE COMPCON*, pp. 133–137, 1982.

[15] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proceedings of the 20th Annual Symposium on Theory of Computing (STOC)*, pp. 103–112, ACM Press, 1988.

[16] C. Blundo and A. Cresti, "Space requirements for broadcast encryption," in *Advances in Cryptology—EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 287–298, Springer-Verlag, 1994.

[17] C. Blundo, L. A. F. Mattos, and D. R. Stinson, "Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution," in *Advances in Cryptology—CRYPTO '96*, vol. 1109 of *LNCS*, pp. 387–400, Springer-Verlag, 1996.

[18] B. D. Boer, "Diffie-Hellman is as strong as discrete log for certain primes," in *Advances in Cryptology—CRYPTO '88*, vol. 403 of *LNCS*, pp. 520–539, Springer-Verlag, 1990.

[19] D. Boneh and M. Franklin, "An efficient public key traitor tracing scheme," in *Advances in Cryptology—CRYPTO '99*, vol. 1666 of *LNCS*, pp. 338–353, Springer-Verlag, 1999.

[20] D. Boneh and M. Franklin, "Anonymous authentication with subset queries," in *CCS '99, Proceedings of the 6th ACM Conference on Computer and Communications Security*, pp. 113–119, ACM, 1999.

[21] A. Bosselaers, H. Dobbertin, and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD," in *Fast Software Encryption: Third International Workshop*, vol. 1039 of *LNCS*, pp. 71–82, Springer-Verlag, 1996.

[22] J. Boyar, D. Chaum, I. Damgård, and T. Pederson, "Convertible undeniable signatures," in *Advances in Cryptology—CRYPTO '90*, vol. 537 of *LNCS*, pp. 189–205, Springer-Verlag, 1991.

[23] C. Boyd, "Digital multisignatures," in *Cryptography and Coding*, pp. 241–246, Oxford University Press, 1989.

[24] B. O. Brachtl, D. Coppersmith, M. M. Hyden, S. M. Matyas, C. H. Meyer, J. Oseas, S. Pilpel, and M. Schilling, "Data authentication using modification detection codes based on a public one-way encryption function." U.S. Patent # 4,908,861, 13, Mar 1990, 1990.

[25] S. A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates.* MIT Press, Cambridge, Massachusetts, 2000.

[26] S. Brands, "An efficient off-line electronic cash system based on the representation problem," Tech. Rep. CS-R9323, CWI, 1993.

[27] G. Brassard, C. Chaum, and C. Crepéau, "Minimum disclosure proofs of knowledge," *Journal of Computer and System Sciences*, vol. 37, no. 2, pp. 156–189, 1988.

[28] G. Brassard, C. Crépeau, R. Jozsa, and D. Langlois, "A quantum bit commitment scheme provably unbreakable by both parties," in *Proceedings of the 34th Annual Symposium on Foundations of Computer Science, FOCS '93*, pp. 362–371, IEEE Computer Society, 1993.

[29] G. Brassard, C. Crépeau, and J. M. Robert, "Information theoretic reductions among disclosure problems," in *Proceedings of the 27th Annual Symposium on Foundations of Computer Science, FOCS '88*, pp. 168–173, IEEE Computer Society, 1986.

[30] G. Brassard, C. Crépeau, and M. Yung, "Constant-round perfect zero-knowledge computationally convincing protocols," *Theoretical computer science*, vol. 84, no. 1, pp. 23–52, 1991.

[31] G. Brassard and M. Yung, "One-way group actions," in *Advances in Cryptology—CRYPTO '90*, vol. 537 of *LNCS*, pp. 94–107, Springer-Verlag, 1991.

[32] E. F. Brickell, "A fast modular multiplication algorithm with application to two key cryptography," in *Advances in Cryptology: Proceedings of Crypto 82*, pp. 51–60, Plenum Press, New York and London, 1983.

[33] J. A. Buchmann, *Introduction to Cryptography*. Springer-Verlag, 2000.

[34] J. Camenisch, *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zurich, 1998.

[35] J. Camenisch and M. Michels, "Separability and efficiency for generic group signature schemes," in *Advances in Cryptology—CRYPTO '99*, vol. 1666 of *LNCS*, pp. 413–430, Springer-Verlag, 1999.

[36] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," Report 2001/019, Cryptology ePrint Archive, Mar. 2001.

[37] J. Camenisch and A. Lysyanskaya, "An identity escrow scheme with appointed verifiers," in *Advances in Cryptology—CRYPTO 2001*, vol. 2139 of *LNCS*, pp. 388–407, Springer-Verlag, 2001.

[38] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups (extended abstract)," in *Advances in Cryptology—CRYPTO '97*, vol. 1294 of *LNCS*, pp. 410–424, Springer-Verlag, 1997.

[39] J. Camenish, "Efficient and generalized group signatures," in *Advances in Cryptology—EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 465–479, Springer-Verlag, 1997.

[40] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Adaptive security for threshold cryptosystems," in *Advances in Cryptology—CRYPTO '99*, vol. 1666 of *LNCS*, pp. 98–115, Springer-Verlag, 1999.

[41] R. Canetti, J. Kilian, E. Petrank, and A. Rosen, "Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 1–47, 2002. Preliminary version in *STOC '01*.

[42] D. Chaum, C. Crépeau, and I. Damgård, "Multi-party unconditionally secure protocols," in *Proceedings of the 20th Annual Symposium on Theory of Computing (STOC)*, pp. 11–19, ACM, 1988.

[43] D. Chaum, J.-H. Evertse, and J. van der Graaf, "An improved protocol for demonstrating possession of a discrete logarithm and some generalizations," in *Advances in Cryptology—EUROCRYPT '87*, vol. 304 of *LNCS*, pp. 127–141, Springer-Verlag, 1987.

[44] D. Chaum and E. v. Heyst, "Group signatures," in *Advances in Cryptology—EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 257–265, Springer-Verlag, 1991.

[45] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology—CRYPTO '92*, vol. 740 of *LNCS*, pp. 89–105, Springer-Verlag, 1993.

[46] D. Chaum, E. van Heijst, and B. Pfitzmann, "Cryptographically strong undeniable signatures, unconditionally secure for the signer," in *Advances in Cryptology—CRYPTO '91*, vol. 576 of *LNCS*, pp. 470–484, Springer-Verlag, 1991.

[47] D. Chaum, E. van Heijst, and B. Pfitzmann, "Cryptographically strong undeniable signatures, unconditionally secure for the signer," in *Advances in Cryptology—CRYPTO '91*, vol. 576 of *LNCS*, pp. 470–484, Springer-Verlag, 1992.

[48] D. Chaum, "Zero-knowledge undeniable signatures," in *Advances in Cryptology—EUROCRYPT '90*, vol. 473 of *LNCS*, pp. 458–464, Springer-Verlag, 1991.

[49] D. Chaum, "Designated confirmer signatures," in *Advances in Cryptology—EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 86–91, Springer-Verlag, 1994.

[50] D. Chaum and J.-H. Evertse, "A secure and privacy-protecting protocol for transmitting personal information between organizations," in *Advances in Cryptology—CRYPTO '86*, vol. 263 of *LNCS*, pp. 118–167, Springer-Verlag, 1987.

[51] D. Chaum and H. van Antwerpen, "Undeniable signatures," in *Advances in Cryptology—CRYPTO '89*, vol. 435 of *LNCS*, pp. 212–216, Springer-Verlag, 1990.

[52] D. L. Chaum, "Security without identification: transaction systems to make big brother obsolete," *CACM*, vol. 28, pp. 1030–1044, Oct. 1985.

[53] L. Chen, "Access with pseudonyms," in *Cryptography: Policy and Algorithms*, vol. 1029 of *LNCS*, pp. 232–243, Springer-Verlag, 1995.

[54] L. Chen and T. Pesersen, "New group signature schemes," in *Advances in Cryptology—EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 171–181, Springer-Verlag, 1995.

[55] L. Chen and T. P. Pedersen, "On the efficiency of group signatures providing information-theoretic anonymity," in *Advances in Cryptology—EUROCRYPT '95*, vol. 921 of *LNCS*, pp. 39–49, Springer-Verlag, 1995.

[56] C. H. Chiou and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Transactions on Software Engineering*, vol. 15, no. 8, pp. 929–934, 1989.

[57] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *Advances in Cryptology—CRYPTO '94*, vol. 839 of *LNCS*, pp. 257–270, Springer-Verlag, 1994.

[58] B. Chor, A. Fiat, M. Naor, and B. Pinkas, "Tracing traitors," *IEEE Transactions on Information Theory*, vol. 46, pp. 893–910, May 2000.

[59] H. Cohen, *A Course in Computational Algebraic Number Theory*, vol. 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993.

[60] C. Crépeau, "Equivalence between two flavors of oblivious transfer," in *Advances in Cryptology—CRYPTO '87*, vol. 293 of *LNCS*, pp. 350–354, Springer-Verlag, 1987.

[61] C. Crépeau and J. Kilian, "Achieving oblivious transfer using weakened security assumptions," in *Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FOCS '88*, pp. 42–52, IEEE Computer Society, 1988.

[62] C. Crépeau, "Efficient cryptographic protocols based on noisy channels," in *Advances in Cryptology—EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 306–317, Springer-Verlag, 1997.

[63] I. Damgård, *The Application of Claw Free Functions in Cryptography*. PhD thesis, Aarhus University, Mathematical Institute, 1988.

[64] I. Damgård, "A design principle for hash functions," in *Advances in Cryptology—CRYPTO '89*, vol. 435 of *LNCS*, pp. 416–427, Springer-Verlag, 1990.

[65] I. Damgård, J. Kilian, and L. Salvail, "On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions," in *Advances in Cryptology—CRYPTO '99*, vol. 1592 of *LNCS*, pp. 56–73, Springer-Verlag, 1999.

[66] I. Damgård, T. P. Pedersen, and B. Pfitzmann, "On the existence of statistically hiding bit commitment schemes and fail-stop signatures," *Journal of Cryptology*, vol. 10, no. 3, pp. 163–194, 1997.

[67] I. B. Damgård, "Collision free hash functions and public key signatureschemes," in *Advances in Cryptology—EUROCRYPT '87*, vol. 304 of *LNCS*, pp. 203–216, Springer-Verlag, 1988.

[68] I. Damgård, "Commitment schemes and zero-knowledge protocols," in *Lectures on Data Security*, vol. 1561 of *LNCS*, pp. 63–86, Springer-Verlag, 1999.

[69] I. Damgård and E. Fujisaki, "A statistically-hiding integer commitment scheme based on groups with hidden order," in *Advances in Cryptology—ASIACRYPT 2002*, vol. 2501 of *LNCS*, pp. 125–142, Springer-Verlag, 2002.

[70] I. Damgård and T. P. Pedersen, "New convertible undeniable signature schemes," in *Advances in Cryptology—EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 372–386, Springer-Verlag, 1996.

[71] I. B. Damgård, "Payment systems and credential mechanisms with provable security against abuse by individuals," in *Advances in Cryptology—CRYPTO '88*, vol. 403 of *LNCS*, pp. 328–335, Springer-Verlag, 1990.

[72] R. W. Davies and W. L. Price, "Digital signature—an update," in *Proc. International Conference on Computer Communication*, pp. 843–847, Elsevier, 1985.

[73] H. Delfs and H. Knebl, *Introduction to Cryptography: Principles and Applications*. Springer-Verlag, 2002.

[74] B. den Boer and A. Bosselaers, "Collisions for the compression function of MD5," in *Advances in Cryptology—EUROCRYPT '93*, vol. 765 of *LNCS*, pp. 293–304, Springer-Verlag, 1994.

[75] Y. Desmedt and M. Yung, "Weakness of undeniable signature schemes," in *Advances in Cryptology—CRYPTO '91*, vol. 576 of *LNCS*, pp. 205–220, Springer-Verlag, 1991.

[76] Y. Desmedt, "Society and group oriented cryptography: A new concept," in *Advances in Cryptology—CRYPTO '87*, vol. 293 of *LNCS*, pp. 120–127, Springer-Verlag, 1988.

[77] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Advances in Cryptology—CRYPTO '89*, vol. 435 of *LNCS*, pp. 307–315, Springer-Verlag, 1990.

[78] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[79] H. Dobbertin, "Cryptanalysis of MD4," *Journal of Cryptology*, vol. 11, pp. 253–271, 1998.

[80] H. Dobbertin, "Cryptanalysis of MD5 compress." Presented at the Rumpsession of *EUROCRYPT '96*, 1996.

[81] D. Dolev, C. Dwork, and M. Naor, "Non-malleable cryptography," in *Proceedings of the 23rd Annual Symposium on Theory of Computing (STOC '91)*, pp. 542–552, ACM Press, 1991.

[82] D. Dolev, C. Dwork, and M. Naor, "Nonmalleable cryptography," *SIAM Journal on Computing*, vol. 30, no. 2, pp. 391–437, 2000.

[83] C. Dwork, M. Naor, and A. Sahai, "Concurrent zero knowledge," in *Proceedings of the 30th Annual Symposium on Theory Of Computing (STOC '98)*, pp. 409–418, ACM Press, 1998.

[84] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology — CRYPTO '88*, vol. 196 of *LNCS*, pp. 10–18, Springer-Verlag, 1985.

[85] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[86] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *CACM*, vol. 28, pp. 637–647, 1985.

[87] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of Cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[88] U. Feige and A. Shamir, "Witness indistinguishability and witness hiding protocols," in *Proceedings of the 22nd Annual Symposium on Theory of Computing (STOC '90)*, pp. 416–426, ACM Press, 1990.

[89] A. Fiat and M. Naor, "Broadcast encryption," in *Advances in Cryptology—CRYPTO '93*, vol. 773 of *LNCS*, pp. 480–491, Springer-Verlag, 1993.

[90] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO '86*, vol. 263 of *LNCS*, pp. 186–194, Springer-Verlag, 1987.

[91] FIPS 180-1, "Secure hash standard." Federal Information Processing Standard Publication 180-1, NIST, US department of commerce, 1995.

[92] FIPS 180-2, "Secure hash standard." Federal Information Processing Standard Publication 180-2 (Draft), NIST, US department of commerce, 2001.

[93] FIPS 186, "Digital signature standard." Federal Information Processing Standard Publication 186, NIST, US department of commerce, 1994.

[94] A. Fujioka, T. Okamoto, and K. Ohta, "Interactive bi-proof systems and undeniable signature schemes," in *Advances in Cryptology—EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 243–256, Springer-Verlag, 1991.

[95] E. Fujisaki and T. Okamoto, "Statistical zero-knowledge protocols to prove modular polynomial relations," *IEICE Transaction of Fundamentals of Electronic Communications and Computer Science*, vol. E82-A(1):, no. 1, pp. 81–92, 1999. Previous version in *Crypto '97*.

[96] E. Gafni, J. Staddon, and Y. L. Yin, "Efficient methods for integrating traceability and broadcast encryption," in *Advances in Cryptology—CRYPTO '99*, vol. 1666 of *LNCS*, pp. 372–387, Springer-Verlag, 1999.

[97] J. A. Garay, J. Staddon, and A. Wool, "Long-lived broadcast encryption," in *Advances in Cryptology—CRYPTO 2000*, vol. 1880 of *LNCS*, pp. 333–352, Springer-Verlag, 2000.

[98] R. Gennaro, T. Rabin, and H. Krawczyk, "RSA-based undeniable signatures," *Journal of Cryptology*, vol. 13, no. 4, pp. 397–416, 2000.

[99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust and efficient sharing of RSA functions," in *Advances in Cryptology—CRYPTO '96*, vol. 1109 of *LNCS*, pp. 157–172, Springer-Verlag, 1996.

[100] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signatures," in *Advances in Cryptology—EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 354–371, Springer-Verlag, 1996.

[101] H. Ghodosi, J. Pieprzyk, C. Charnes, and R. Safavi-Naini, "Cryptosystems for hierarchical groups," in *Australasian Conference on Information Security and Privacy*, pp. 275–286, 1996.

[102] J. K. Gibson, "Some comments on damgård's hashing principle," *Electronics Letters*, vol. 26, no. 15, pp. 1178–1179, 1990.

[103] J. K. Gibson, "Discrete logarithm hash functionthat is collision free and one way," *IEE Proceedings-E*, vol. 138, no. 6, pp. 407–410, 1991.

[104] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocol with honest majority," in *Proceedings of the*

19th Annual Symposium on Theory of Computing (STOC '87), pp. 218–229, ACM Press, 1987.

[105] O. Goldreich, S. Micali, and A. Wigderson, "How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design," in *Advances in Cryptology—CRYPTO '86*, vol. 263 of *LNCS*, pp. 171–185, Springer-Verlag, 1987.

[106] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems," *Journal of the ACM*, vol. 38, no. 1, pp. 691–729, 1991.

[107] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge," *Journal of Cryptology*, vol. 7, pp. 1–32, 1994.

[108] O. Goldreich, *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[109] O. Goldreich and H. Krawczyk, "On the composition of zero-knowledge proof systems," *SIAM Journal on Computing*, vol. 25, no. 1, pp. 169–192, 1996. Preliminary version appeared in *ICALP '90*.

[110] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, pp. 186–208, 1989. Preliminary version in *STOC '85*.

[111] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, pp. 270–299, 1984.

[112] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, pp. 281–308, Apr. 1988.

[113] D. Halevi and A. Shamir, "The LSD broadcast encryption scheme," in *Advances in Cryptology—CRYPTO 2002*, vol. 2442 of *LNCS*, pp. 47–60, Springer-Verlag, 2002.

[114] S. Halevi, "Efficient commitment schemes with bounded sender and unbounded receiver," in *Advances in Cryptology—CRYPTO '95*, vol. 963 of *LNCS*, pp. 84–96, Springer-Verlag, 1995.

[115] S. Halevi and S. Micali, "Practical and provably-secure commitment schemes from collision-free hashing," in *Advances in Cryptology—CRYPTO '96*, vol. 1109 of *LNCS*, pp. 201–215, Springer-Verlag, 1996.

[116] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*. Oxford University Press, 5th ed., 1979.

[117] L. Harn, "Group-oriented (t, n) threshold digital signature scheme and digital multisignature," *IEE Proc. Computers and Digital Techniques*, vol. 141, pp. 307–313, September 1994.

[118] I. N. Herstein, *Topics in Algebra*. Xerox College Publishing, 1975.

[119] E. V. Heyst and T. P. Pedersen, "How to make efficient fail-stop signatures," in *Advances in Cryptology—EUROCRYPT '92*, vol. 658 of *LNCS*, pp. 366–377, Springer-Verlag, 1993.

[120] P. Horster, M. Michels, and H. Petersen, "Meta-multisignatures schemes based on the discrete logarithm problem," in *Information Security:The Next Decade. Proceedings of the IFIP TC11 Eleventh International Conference on Information Security, IFIP/SEC'95*, pp. 128–141, Chapman & Hall, 1995.

[121] R. Impagliazzo and M. Luby, "One-way functions are essential for complexity-based cryptography," in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS '89*, pp. 230–235, IEEE Computer Society, 1989.

[122] ISO/IEC 10118-4, "Information technology—Security techniques—Hash-functions—Part 4: Hash-functions using modular arithmetic," 1998.

[123] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," *NEC Research & Development*, vol. 71, pp. 1–8, Oct. 1983.

[124] M. Jakobsson, "Blackmailing using undeniable signatures," in *Advances in Cryptology—EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 425–427, Springer-Verlag, 1994.

[125] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," in *Advances in Cryptology—EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 143–154, Springer-Verlag, 1996.

[126] M. Jakobsson and M. Yung, "Proving without knowing: On oblivious, agnostic and blindfolded provers," in *Advances in Cryptology—CRYPTO '96*, vol. 1109 of *LNCS*, pp. 201–215, Springer-Verlag, 1996.

[127] M. Just, E. Kranakis, D. Krizanc, and P. C. van Oorschot, "On key distribution via true broadcasting," in *CCS '94, Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pp. 81–88, ACM Press, Nov. 1994.

[128] A. Kiayias and M. Yung, "Traitor tracing with constant transmission rate," in *Advances in Cryptology—EUROCRYPT 2002*, vol. 2332 of *LNCS*, pp. 450–465, Springer-Verlag, 2002.

[129] J. Kilian, "Founding cryptography on oblivious transfer," in *Proceedings of the 20th Annual Symposium on Theory of Computing (STOC '88)*, pp. 20–31, IEEE Computer Society, 1988.

[130] J. Kilian and E. Petrank, "Identity escrow," in *Advances in Cryptology—CRYPTO '98*, vol. 1462 of *LNCS*, pp. 169–185, Springer-Verlag, 1998.

[131] D. E. Knuth, *The Art of Computer Programming: Volume 2/Seminumerical Algorithms*. Addison-Wesley, 2nd ed., 1981.

[132] N. Koblitz, *A Course in Number Theory and Cryptography*, vol. 114 of *Graduate Texts in Mathematics*. Springer-Verlag, 2nd ed., 1994.

[133] C. Koç, "RSA hardware implementation," Tech. Rep. TR-801, RSA Laboratories, 1996.

[134] K. Kurosawa and Y. Desmedt, "Optimum traitor tracing and asymmetric schemes," in *Advances in Cryptology—EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 145–157, Springer-Verlag, 1998.

[135] K. Kurosawa and T. Yoshida, "Linear code implies public-key traitor tracing," in *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002*, vol. 2274 of *LNCS*, pp. 172–187, Springer-Verlag, 2002.

[136] X. Lai and J. L. Massey, "Hash functions based on block ciphers," in *Advances in Cryptology—EUROCRYPT '92*, vol. 658 of *LNCS*, pp. 55–70, Springer-Verlag, 1993.

[137] C. S. Laih and L. Harn, "Generalized threshold cryptosystems," in *Advances in Cryptology—ASIACRYPT '91*, vol. 739 of *LNCS*, pp. 159–166, Springer-Verlag, 1993.

[138] C. H. Lee, X. Deng, and H. Zhu, "Design and security analysis of anonymous group identification protocols," in *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002*, vol. 2274 of *LNCS*, pp. 188–198, Springer-Verlag, 2002.

[139] C.-M. Li, T. Hwang, and N.-Y. Lee, "Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders," in *Advances in Cryptology—EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 194–204, Springer-Verlag, 1995.

[140] C.-M. Li, T. Hwang, N.-Y. Lee, and J.-J. Tsai, "(t, n) threshold-multisignature schemes and generalized-multisignature scheme where suspected forgery implies traceability of adversarial shareholders," *Cryptologia*, vol. 24, no. 3, pp. 250–268, 2000.

[141] C. H. Lim and P. J. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup," in *Advances in Cryptology—CRYPTO '97*, vol. 1296 of *LNCS*, pp. 249–263, Springer-Verlag, 1997.

[142] H. K. Lo and H. F. Chau, "Is quantum bit commitment really possible?," *Physical Review Letters*, vol. 78, pp. 3410–3413, 1997. Preliminary version in Los Alamos preprint archive, xxx.lanl.gov/abs/quant-ph/9603004 (1996).

[143] M. Luby and J. Staddon, "Combinatorial bounds for broadcast encryption," in *Advances in Cryptology—EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 512–526, Springer-Verlag, 1998.

[144] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf, "Pseudonym systems," in *Selected Areas in Cryptography, 6th Annual International Workshop, SAC '99*, vol. 1758 of *LNCS*, pp. 184–214, Springer-Verlag, 1999.

[145] Y.-D. Lyuu and M.-L. Wu, "A fully public-key traitor-tracing scheme," *WSEAS Transactions on Circuits*, vol. 1, no. 1, pp. 88–93, 2002.

[146] Y.-D. Lyuu and M.-L. Wu, "A fully public-key traitor-tracing scheme," in *Proceedings of the 6th WSEAS International Multiconference on Circuits, Systems, Communications, and Computers (CSCC 2002)*, July 2002.

[147] Y.-D. Lyuu and M.-L. Wu, "Group undeniable signatures," in *Proceedings of the 2002 International Conference On Information Security, ICIS 2002*, (Rio De Janeiro, Brazil), Oct. 2002.

[148] Y.-D. Lyuu and M.-L. Wu, "Convertible group undeniable signatures," in *Information Security and Cryptology— ICISC 2002*, vol. 2587 of *LNCS*, pp. 49–62, Springer Verlag, 2003.

[149] Y.-D. Lyuu and M.-L. Wu, "Group undeniable signatures," *International Journal of Computer Research*, vol. 12, no. 2, 2003.

[150] Y.-D. Lyuu and M.-L. Wu, "Group undeniable signatures with convertibility." submitted, 2003.

[151] S. M. Matyas, "Key processing with control vectors," *Journal of Cryptology*, vol. 3, pp. 113–136, 1991.

[152] S. M. Matyas, C. H. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Technical Disclosure Bulletin*, vol. 27, pp. 5658–5659, 1985.

[153] U. Maurer, "Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms," in *Advances in Cryptology—CRYPTO '94*, vol. 839 of *LNCS*, pp. 271–281, Springer-Verlag, 1994.

[154] U. Maurer and S. Wolf, "The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms," *SIAM Journal on Computing*, vol. 28, no. 5, pp. 1689–1721, 1999.

[155] U. Maurer and S. Wolf, "The Diffie-Hellman protocol," *Designs, Codes, and Cryptography*, vol. 19, pp. 147–171, 2000.

[156] D. Mayers, "Unconditionally secure quantum bit commitment is impossible," *Physical review letters*, vol. 78, pp. 3414–3417, 1997.

[157] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1997.

[158] R. C. Merkle, "A fast software one-way hash function," *Journal of Cryptology*, vol. 3, pp. 43–58, 1990.

[159] R. C. Merkle, "One way hash functions and DES," in *Advances in Cryptology—CRYPTO '89*, vol. 435 of *LNCS*, pp. 428–446, Springer-Verlag, 1990.

[160] C. H. Meyer and M. Schilling, "Secure program load with manipulation detection code," in *Proceedings of the 6th Worldwide Congress on Computer and Communicatios Security and Protection (SECURICOM'88)*, pp. 111–130, 1988.

[161] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures: extended abstract," in *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 245–254, ACM Press, 2001.

[162] S. Micali, C. Rackoff, and B. Sloan, "The notion of security for probabilistic cryptosystems," *SIAM Journal on Computing*, vol. 17, pp. 412–426, Apr. 1988.

[163] M. Michel, "Breaking and repairing a convertible undeniable signature scheme," in *CCS '96, Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pp. 148–152, ACM Press, 1996.

[164] M. Michels and M. Stadler, "Efficient convertible undeniable signature schemes," in *Selected Areas in Cryptography, 4th Annual International Workshop, SAC '97*, pp. 231–244, 1997.

[165] G. L. Miller, "Riemann's hypothesis and tests for primality," *Journal of Computer and Systems Science*, vol. 13, pp. 300–317, 1976.

[166] S. Miyaguchi, K. Ohta, and M. Iwata, "128-bit hash function (n-hash)," *NTT Review*, vol. 2, pp. 128–132, 1990.

[167] M. Nao and M. Yung, "Universal one-way hash function and their cryptographic application," in *Proceedings of the 21th Annual Symposium on Theory of Computing (STOC)*, pp. 33–43, ACM Press, 1989.

[168] M. Nao, "Bit commitment using pseudorandomness," *Journal of Cryptology*, vol. 2, no. 2, pp. 151–158, 1991.

[169] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Advances in Cryptology—CRYPTO 2001*, vol. 2139 of *LNCS*, pp. 41–62, Springer-Verlag, 2001.

[170] M. Naor and B. Pinkas, "Threshold traitor tracing," in *Advances in Cryptology—CRYPTO '98*, vol. 1462 of *LNCS*, pp. 502–517, Springer-Verlag, 1998.

[171] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attack," in *Proceedings of the 22nd Annual Symposium on Theory of Computing (STOC)*, pp. 427–437, ACM, 1990.

[172] M. J. Norris and G. J. Simmons, "Algorithms for high-speed modular arithmetic," *Congressus Numerantium*, vol. 31, pp. 153–163, 1981.

[173] K. Ohta, T. Okamoto, and A. Fujioka, "Secure bit commitment function against divertibility," in *Advances in Cryptology—EUROCRYPT '92*, vol. 658 of *LNCS*, pp. 324–340, Springer-Verlag, 1992.

[174] T. Okamoto, "Provable secure and practical identification schemes and corresponding signature schemes," in *Advances in Cryptology—CRYPTO '92*, vol. 740 of *LNCS*, pp. 31–53, Springer-Verlag, 1993.

[175] T. Okamoto, "Designated confirmer signatures and public-key encryption are equivalent," in *Advances in Cryptology—CRYPTO '94*, vol. 839 of *LNCS*, pp. 61–74, Springer-Verlag, 1994.

[176] R. Ostrovsky, R. Venkatesan, and M. Yung, "Secure commitment against a powerful adversary," in *Proceedings of STACS 92*, vol. 577 of *LNCS*, pp. 439–448, Springer-Verlag, 1992.

[177] C. H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.

[178] R. Pass, "On deniability in the common reference string and random oracle model," in *Advances in Cryptology—CRYPTO 2003*, LNCS, Springer-Verlag, 2003.

[179] T. Pedersen, "Distributed provers with applications to undeniable signatures," in *Advances in Cryptology—EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 221–242, Springer-Verlag, 1991.

[180] H. Petersen, "How to convert any digital signature scheme into a group signature scheme," in *Security Protocols Workshop'97*, vol. 1361 of *LNCS*, pp. 67–78, Springer-Verlag, 1991.

[181] B. Pfitzmann, "Trials of traced traitors," in *Information Hiding, First International Workshop, IH '96*, vol. 1174 of *LNCS*, pp. 49–64, Springer-Verlag, 1996.

[182] J. Pieprzyk, J. Seberry, and Y. Zheng, "A one-way hashing algorithm with variable length of output," in *AUSCRYPT '92*, vol. 921 of *LNCS*, pp. 83–104, Springer-Verlag, 1993.

[183] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Advances in Cryptology—EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 387–398, Springer-Verlag, 1996.

[184] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, pp. 361–396, Mar. 2000.

[185] B. Preneel, *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.

[186] B. Preneel, "The state of cryptographic hash functions," in *Lectures on Data Security*, vol. 1561 of *LNCS*, pp. 158–182, Springer-Verlag, 1999.

[187] J. J. Quisquater and M. Girault, "2n-bit hash functios using n-bit symmetric block cipher algorithms," in *Advances in Cryptology—EUROCRYPT '89*, vol. 434 of *LNCS*, pp. 102–109, Springer-Verlag, 1990.

[188] M. Rabin, "How to exchange secrets by oblivious transfer," Tech. Rep. TR-81, Harvard Aiken Computation Laboratory, 1981.

[189] M. O. Rabin, "Digital signatures and public-key functions as intractible as factorization," Tech. Rep. MIT/LCS/TR-212, MIT Laboratory for Computer Science, Jan. 1979.

[190] M. O. Rabin, "Probabilistic algorithms for testing primality," *Journal of Number Theory*, vol. 12, pp. 128–138, 1980.

[191] C. Rackoff and D. R. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," in *Advances in Cryptology—CRYPTO '91*, vol. 576 of *LNCS*, pp. 433–444, Springer-Verlag, 1992.

[192] RIPE, *Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-Race 1040)*, vol. 1007 of *LNCS*. Springer-Verlag, 1995.

[193] R. Rivest, "The MD4 message-digest algorithm," 1992. RFC 1320, the Internet Engineering Task Force.

[194] R. Rivest, "The MD5 message-digest algorithm," 1992. RFC 1321, the Internet Engineering Task Force.

[195] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *CACM*, vol. 21, pp. 120–126, Feb. 1978.

[196] K. H. Rosen, *Elementary Number Theory and Its Applications.* Addison Wesley, third ed., 1993.

[197] J. B. Rosser and L. Schoenfeld, "Approximate formulas for some functions of prime numbers," *Illinois Journal of Mathematics*, vol. 6, pp. 64–94, 1962.

[198] L. Salvail, "Quantum bit commitment from a physical assumption," in *Advances in Cryptology—CRYPTO '98*, vol. 1462 of *LNCS*, pp. 338–353, Springer-Verlag, 1998.

[199] L. Salvail, "The search for the holy grail in quantum cryptography," in *Lectures on Data Security*, vol. 1561 of *LNCS*, pp. 183–216, Springer-Verlag, 1998.

[200] A. D. Santis, G. D. Crescenzo, G. Persiano, and M. Yung, "On monotone formula closure of SZK," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, FOCS '94*, pp. 454–465, IEEE Computer Society, 1994.

[201] A. D. Santis, G. D. Crescenzo, and G. Persiano, "Communication-efficient anonymous group identification," in *CCS '98, Proceedings of the 5th ACM Conference on Computer and Communications Security*, pp. 73–82, ACM, Nov. 1998.

[202] B. Schneier, *Applied Cryptography.* John Wiley & Sons, second ed., 1996.

[203] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[204] A. Shamir, "How to share a secret," *CACM*, vol. 22, no. 11, pp. 612–613, 1979.

[205] D. Shanks, *Solved and Unsolved Problems in Number Theory.* Chelsea, 1985.

[206] H. N. Shapiro, *Introduction to the Theory of Numbers*. John Wiley & Sons, 1983.

[207] V. Shoup and R. Gennaro, "Securing threshold cryptosystems against chosen ciphertext attack," in *Advances in Cryptology—EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 1–16, Springer-Verlag, 1998.

[208] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Advances in Cryptology—EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 256–266, Springer-Verlag, 1997.

[209] R. Solovay and V. Strassen, "A fast monte carlo test for primality," *SIAM Journal on Computing*, vol. 6, pp. 84–85, 1977.

[210] D. R. Stinson, *Cryptography: Theory and Practice*. CRC Press, 1995.

[211] D. R. Stinson and R. Wei, "Combinatorial properties and constructions of traceability schemes and frameproof codes," *SIAM Journal on Discrete Math*, vol. 11, no. 1, pp. 41–53, 1998.

[212] D. R. Stinson, "On some methods for unconditionally secure key distribution and broadcast encryption," *Designs, Codes, and Cryptography*, vol. 12, pp. 215–243, Nov. 1997.

[213] D. R. Stinson and R. Wei, "Key preassigned traceability schemes for broadcast encryption," in *Selected Areas in Cryptography '98 (SAC '98)*, vol. 1556 of *LNCS*, pp. 144–156, Springer-Verlag, 1998.

[214] H. C. A. V. Tilborg, *Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial*. Kluwer Academic Publishers, 2000.

[215] M. Tompa and H. Woll, "Random self-reducibility and zero knowledge interactive proofs of possession of information," in *Proceedings of the 28th Annual Symposium on Foundations of Computer Science, FOCS '87*, pp. 472–482, IEEE Computer Society, 1987.

[216] Y. Tsiounis and M. Yung, "On the security of ElGamal-based encryption," in *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98*, vol. 1431 of *LNCS*, pp. 117–134, Springer-Verlag, 1998.

[217] M. Waidner and B. Pfitzmann, "The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability," in *Advances in Cryptology—EUROCRYPT '89*, vol. 434 of *LNCS*, p. 690, Springer-Verlag, 1990.

[218] C. D. Walter, "Faster modular multiplication by operand scaling," in *Advances in Cryptology—CRYPTO '91*, vol. 576 of *LNCS*, pp. 313–323, Springer-Verlag, 1992.

[219] G. Wang, "On the security of the Li-Hwang-Lee-Tsai threshold group signature scheme," in *Information Security and Cryptology—ICISC 2002*, vol. 2587 of *LNCS*, pp. 76–90, Springer-Verlag, 2002.

[220] R. S. Winternitz, "Producing one-way hash function from DES," in *Advances in Cryptology: Proceedings of Crypto 83*, LNCS, pp. 202–207, Springer-Verlag, 1984.

[221] R. S. Winternitz, "A secure one-way hash function built from DES," in *Proc. 1984 IEEE Symposium on Security and Privacy*, pp. 88–90, IEEE Computer Society, 1984.

[222] Y. Zeng, T. Matsumoto, and H. Imai, "Connections between several version of one-way hash functions," in *Proc. SCIS90, the 1990 Symposium on Cryptography and Information Security*, (Nihondaira, Japan), 1990.

Index